Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 33

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

DISTRIBUTED LARGE-SCALE MIXED-INTEGER OPTIMIZATION WITH APPLICATION TO ENERGY AND MULTI-ROBOT NETWORKS

Presentata da: Andrea Camisa

Coordinatore Dottorato	**Supervisore**
Michele Monaci	Giuseppe Notarstefano

Esame finale anno 2021

Abstract

Several decision and control tasks in cyber-physical networks can be formulated as large-scale optimization problems with coupling constraints. In these "constraint-coupled" problems, each agent is associated to a local decision variable, subject to individual constraints. This thesis explores the use of primal decomposition techniques to develop tailored distributed algorithms for this challenging set-up over graphs. We first develop a distributed scheme for convex problems over random time-varying graphs with non-uniform edge probabilities. The approach is then extended to unknown cost functions estimated online. Subsequently, we consider Mixed-Integer Linear Programs (MILPs), which are of great interest in smart grid control and cooperative robotics. We propose a distributed methodological framework to compute a feasible solution to the original MILP, with guaranteed suboptimality bounds, and extend it to general nonconvex problems. Monte Carlo simulations highlight that the approach represents a substantial breakthrough with respect to the state of the art, thus representing a valuable solution for new toolboxes addressing large-scale MILPs. We then propose a distributed Benders decomposition algorithm for asynchronous unreliable networks. The framework has been then used as starting point to develop distributed methodologies for a microgrid optimal control scenario. We develop an ad-hoc distributed strategy for a stochastic set-up with renewable energy sources, and show a case study with samples generated using Generative Adversarial Networks (GANs). We then introduce a software toolbox named ChoiRbot, based on the novel Robot Operating System 2, and show how it facilitates simulations and experiments in distributed multi-robot scenarios. Finally, we consider a Pickup-and-Delivery Vehicle Routing Problem for which we design a distributed method inspired to the approach of general MILPs, and show the efficacy through simulations and experiments in ChoiRbot with ground and aerial robots.

Keywords: Distributed Optimization, Primal Decomposition, Constraint-coupled Optimization, Time-varying Networks, Mixed-integer Linear Programming (MILP), Distributed Microgrid Control, Generative Adversarial Networks (GANs), Cooperative Robotics, Robot Operating System 2, Distributed Pickup and Delivery.

Acknowledgment

This thesis work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638992 – OPT4SMART).

Contents

Introduction

Motivation and Challenges

The modern world is characterized by an increasingly pervasive connectivity that ranges from smartphones, drones, cars to smart factories/buildings, power systems, robots and autonomous systems in general. From the point of view of control theory, all these units (or *agents*) represent independent control systems that interact with the surrounding environment and are endowed with their own computation and communication capabilities. In recent times, there is a paradigm shift in the coordination and the control of these networked systems. Up to now, the main idea was to gather all the relevant information at a unique center that takes the decisions and communicates them to all the involved systems. We term this paradigm the *centralized* approach. With the dramatic increase of the size of these networks, the centralized approach is giving way to the novel *distributed* paradigm, in which there is no decision-making center and the solution to control problems is found in a cooperative way by the interconnected units with local computation and peer-to-peer communication. This transition ought to be facilitated by extending all the existing centralized decision-making algorithms to the distributed setting. Namely, although the units may be of heterogeneous nature (e.g. the dynamical systems may have different dynamics), from an abstract point of view they are treated as being equal to each other and the distributed algorithm itself must be responsible for achieving self-coordination of all the systems. Optimization techniques play an important role in the considered scenarios, since several important control tasks that arise in these networked systems (as e.g. stochastic energy network control, cooperative model predictive control, vehicle routing in robotic networks) can be formulated as optimization problems with a network-induced structure. The term *distributed optimization* refers to a branch of optimization literature that develops distributed algorithms to solve optimization problems over networks.

We concentrate on one of the most challenging scenarios arising in distributed optimization, namely problems that are *constraint-coupled* and *mixed-integer*. With the term constraint coupled we refer to optimization problems emerging in so-called large-scale networks (i.e. networks with a large number of nodes) in which each agent in the network is associated to a decision variable that is subject to individual constraints and aims to minimize its own cost function. However, all the decision variables must also satisfy global coupling constraints, therefore the agents must cooperate in order to find a solution by negotiating the usage of the coupling constraint. This optimization scenario is general and captures important distributed control applications (such as the ones mentioned above) and should be contrasted with the so-called cost-coupled optimization (typically resulting from distributed estimation and learning problems), in which all the agents share the *same* decision variable. Only recently is the constraint-coupled optimization receiving significant attention, while the cost-coupled set-up has been widely investigated in the literature, see the recent surveys [53, 79, 86]. One main difficulty of this optimization scenario is that the optimization variable can be extremely large. For this reason, in the distributed optimization literature the majority of works has concentrated on convex problems, which enjoy a number of useful mathematical properties and, as such, are easier to solve if compared to nonconvex ones. Mixed-integer optimization, on the other hand, refers to a branch of the optimization literature that studies problems in which part of the decision variables take on integer values only. Mixed-integer programs are fundamental building blocks for several applications in smart networks (as, e.g., distributed optimal control of hybrid and nonlinear systems) and typically appear as dynamic instances that need to be solved in real-time by fast algorithms. An important consequence of the mixed-integer nature of the optimization variable is that the resulting problem is nonconvex and NP-hard. The solution process becomes extremely more involved (even in the well-established centralized setting) and requires computation-intensive enumeration techniques. This calls for novel solution strategies that must simultaneously take into account both the distributed nature of problem and its combinatorial complexity. Moreover, in several contexts of interest the communication topology cannot be considered as fixed, but it changes over time. Indeed, the communication links among the agents may be unavailable in certain moments, e.g. due to a temporary failure.

Summary of the Contributions

This thesis contributes to the field of large-scale convex, mixed-integer and nonconvex distributed optimization with local and coupling constraints over static and time-varying networks by proposing novel solution methodologies based on the primal decomposition technique. Moreover, we apply the developed methodologies to two concrete application domains, namely distributed stochastic microgrid control and distributed multi-robot pickup and delivery, and develop a software toolbox for simulations and experiments in cooperative robotics.

First, we study large-scale convex optimization problems that have to be solved over peer-to-peer networks of agents with a possibly time-varying communication topology. Specifically, we consider optimization problems where agents aim to minimize the sum of local objective functions, each one depending on a local variable in an individual constraint set, subject to global coupling constraints. For this class of constraint-coupled problems, we propose and analyze a new distributed algorithmic framework for random time-varying networks where edges in the graph have certain (non-uniform) probabilities of being active at each time step. In the developed approach, agents perform their updates by using only the information coming from the currently active communication links and protect their private information throughout all the process. Notably, the amount of local computation does not depend on the size of the network, therefore the algorithm can be also implemented in very large networks of agents. We prove that almost surely the objective value converges to the optimal cost, and any limit point of the local solution estimates is an optimal solution to the constraint-coupled problem. We also derive convergence rates in objective value for various step-size choices. We then extend this approach to a scenario in which the exact form of the cost functions is a-priori unknown and propose a new distributed scheme with online estimated costs that maintains the same convergence properties of the original algorithm.

Second, we consider the challenge of designing fast, distributed algorithms to compute feasible solutions with guaranteed suboptimality bounds to large-scale Mixed-Integer Linear Programs (MILPs) with a constraint-coupled structure. MILPs are known to be NP-hard and, as such, finding a solution can be extremely expensive from a computational point of view. By relying on a suitably constructed convex approximation of the problem with an appropriate tightening of the coupling constraints, we propose a novel distributed methodology that exploits the primal decomposition approach to find a feasible (possibly suboptimal) solution to the original MILP. We prove asymptotic feasibility of the solution computed by the algorithm and demonstrate guaranteed a-priori and a-posteriori suboptimality bounds. Then, by adapting the constraint tightening approach, we prove that feasibility can be achieved in finite time and determine associated suboptimality bounds. We show through Monte Carlo simulations that our use of the primal decomposition idea outperforms the state of the art and pushes the optimality of the computed solutions to the limits. We also propose an extension of this framework to general nonconvex programs. In particular, we show how the distributed methodology for MILPs can be also applied to constraint-coupled problems with nonconvex constraint sets, with potential applications e.g. to distributed nonlinear Model Predictive Control. Inspired by the previous derivations, we then propose and analyze a new distributed scheme for constraint-coupled MILPs based on similar ideas but using the so-called Benders decomposition. The resulting distributed algorithm has improved finite-time properties and can be also employed over asynchronous and unreliable communication networks, possibly subject to packet losses.

Finally, we apply the studied approaches to two relevant control scenarios. The first one consists of an energy network application, in which the units of a microgrid aim at a cooperative control without a coordinator. We apply our distributed mixed-integer approaches to a deterministic control scenario, which can be modeled as a constraint-coupled MILP. Then, we consider a more challenging stochastic scenario with renewable energy sources. To this end, we extend the previously developed distributed algorithm for MILPs to the stochastic setting and provide a formal analysis. Specifically, we derive a bound that allows the microgrid units to quantify the worst-case power balance constraint violation based on the generated scenarios in the stochastic problem. Notably, the bound can be computed in a completely distributed way by the units. We then show simulation results performed with the disropt package [48] on a problem instance generated by using Generative Adversarial Neural Networks to synthesize realistic energy production profiles.

In the second application scenario, we consider decision and control tasks for co-operative robotics. To this end, we first develop a distributed robotic platform named ChoiRbot and based on the novel Robot Operating System (ROS) 2. This package is a first attempt to bridge the gap between the theory on distributed optimization-based control and multi-robot networks that can benefit from its application. The ChoiRbot framework, written in Python, exploits the new functionalities of ROS 2 and provides a comprehensive set of libraries to facilitate multi-robot simulations and experiments. ChoiRbot does not require a central coordinating unit and, as such, allows for the implementation on fully distributed control schemes. Thanks to its modular structure, we demonstrate that several applications of interest, possibly involving distributed optimization schemes, are easy to implement and discuss in detail a few use cases that have been tested either in simulation or experimentally. We finally employ the ChoiRbot platform and the algorithms developed in the thesis in a distributed Pickup-and-Delivery Vehicle Routing Problem. Specifically, we tailor our mixed-integer methods to this particular problem and exploit its structure to simplify the implementation of the distributed algorithm. We then show simulations performed in ChoiRbot to highlight the efficacy of the proposed solution approach. Finally, we perform a laboratory experiment with ground and aerial robots running the proposed algorithm on the ChoiRbot platform to highlight practical feasibility of the described technique.

Organization and Chapter Contributions

The thesis organization follows the contribution scheme outlined in the previous section. In the first three chapters we provide all the theoretical contributions, while in the final two chapters we apply the developed algorithms to practical scenarios with simulations and experiments (some theoretical results are provided as well) and describe the developed software toolbox.

In Chapter 1, we provide an introduction to distributed optimization and graph theory. Then, we formalize the constraint-coupled optimization scenario with a brief survey of example applications.

In Chapter 2, we study distributed algorithms for convex constraint-coupled problems. First, we review the primal decomposition technique and show how existing distributed optimization schemes for convex problems over static graphs can be reinterpreted using the primal decomposition formalism. As a side contribution, we propose a parallel algorithm to be used over master-worker architectures. Then, using the primal decomposition approach, we propose a distributed algorithm for random time-varying graphs with non-uniform edge activation probabilities, in which the agents cooperatively negotiate the utilization of the coupling constraint until optimality. The algorithm consists of a two-step iterative procedure in which each agent first solves a local version of the original optimization problem and then exchanges dual variables with neighbors. Importantly, the algorithm preserves privacy of local sensible information. We prove almost sure cost convergence and an almost sure primal recovery property without employing the averaging mechanisms typically used in duality-based algorithms. This result is obtained by reformulating the distributed algorithm as a (centralized) block subgradient method, where each block is associated to an edge of the graph. It is worth noting that almost sure cost convergence of the block subgradient method, which is proved as an intermediate step, represents a contribution per se. Then, for the distributed algorithm, we prove sublinear convergence rates in objective value for constant and diminishing step sizes, which apply both to time-varying networks and to static networks. Moreover, we discuss an extension of the distributed algorithm that can handle equality coupling constraints and prove formal results to provide guidelines for the choice of the algorithm parameters. Subsequently, we extend the results to a more complex scenario in which the exact form of the cost function is not known a-priori. We propose a distributed algorithm with a form similar to the previous one, in which the true cost functions are replaced with the online estimated versions. We then study the impact of this change on the primal decomposition framework and prove that, instead of standard subgradients, this algorithm relies on approximate subgradients that progressively approach the true ones. We then use ϵ-subgradient arguments to conclude that this extended algorithm inherits all the convergence properties of the previous one. The results of this chapter are based on [20–22, 24].

In Chapter 3, we focus on Mixed-Integer Linear Programs (MILPs) with a constraint-coupled structure. Specifically, the considered problems have linear cost, polyhedral local constraints and linear coupling constraints, with the further challenge that some of the variables are integer. We propose an innovative distributed methodology in which we combine the appealing properties of a convex relaxation of the MILP with the primal decomposition approach. Using a restriction-based approach, we show that the asymptotic mixed-integer solution provided by this algorithm is feasible for the original MILP and provide a-priori and a-posteriori suboptimality bounds. By slightly adapting the restriction approach we show that feasibility can be also achieved in a finite number of iterations with similar suboptimality bounds. We show the effectiveness of the proposed method through Monte Carlo simulations on randomly generated large-scale MILPs, which highlight that the solutions computed by the algorithm have extremely low suboptimality levels and also that the assumptions required by our scheme are much relaxed if compared to the state of the art. Then, we propose and analyze a new distributed algorithm for MILPs, which also builds on the solution of the convex relaxation and on the restriction approach, but is based on the so-called Benders decomposition. The distributed algorithm is inspired by the centralized version of this decomposition scheme and consists of an iterative procedure in which agents solve a local version of the problem, generates possibly violated constraints and communicates the active constraints to the neighbors. Eventually, each agent solves a final local MILP to compute its block of solution. We prove that the distributed algorithm converges in finite time to a reconstructed mixed-integer solution that enjoys the same asymptotic properties of the first algorithm studied in this chapter. Remarkably, this new algorithm can work under very general assumptions on the communication topology, which can be time-varying, asynchronous and even unreliable. To perform the local computation step, we provide a routine that the agents can run and prove finite-time convergence of this routine. Finally, we study an extension of the developed methodology for general nonconvex problems with constraint-coupled structure. In particular, we consider convex cost functions and convex coupling constraints, but allow the local constraint sets to be nonconvex. We show finite-time feasibility of the solutions provided by the proposed algorithm, which are particularly useful for receding-horizon distributed optimal control schemes (i.e. Model Predictive Control), where recursive feasibility is required. The results of this chapter are based on [23–26].

In Chapter 4, we focus on a energy system application. In particular, we consider a distributed microgrid control problem consisting of several interconnected power units, namely generators, storages and loads. We first recall the microgrid model and the associated mixed-integer optimal control problem and show that it can be cast as a constraint-coupled MILP. We then apply the method developed in Chapter 3 on a sample instance and provide simulation results. Then, we consider an extended microgrid scenario in which renewable energy sources are also present. Since the power produced by the additional units must be treated as stochastic, we recall a two-stage stochastic formulation of the microgrid control problem. We show that this problem can be also cast as a constraint-coupled MILP and we adapt the distributed algorithm of Chapter 3 to deal with the stochastic scenario. The new algorithm does not require the restriction approach and provides a feasible solution to the two-stage stochastic problem since the first iteration. Then, for the asymptotic solution provided by the algorithm, we prove an upper bound on the violation of the power balance constraint. We show the behavior of the algorithm through simulations, where energy profiles of the renewables are generated using Generative Adversarial Neural Networks trained on real data on the energy production in South Italy. The results of this chapter are based on [27].

In Chapter 5, we finally consider applications of the developed distributed methodologies to cooperative robotics. To this end, we develop the ChoiRbot platform, which is a toolbox for distributed cooperative robotics written in Python and based on the Robot Operating System (ROS) 2. In the first part of the chapter we describe its features. Specifically, the ChoiRbot platform provides a toolset to facilitate the implementation of simulations and experiments of distributed cooperative robotics. ChoiRbot has a tight integration with the disropt package [48] and, for this reason, it is particularly able to handle optimization-based distributed control algorithms. We provide a set of simulations and real experiments on several well-known cooperative robotics scenarios with the ChoiRbot platform. We also perform an experiment to show how constraint-coupled optimization can be applied to a multi-robot scenario. Namely, a set of ground robots have to decide the target position of each of them while keeping the barycenter within certain bounds. This problem is formulated as a constraint-coupled convex problem and solved over ChoiRbot with the distributed algorithms of Chapter 2. In the second part of the chapter, we consider a distributed Pickup-and-Delivery Vehicle Routing Problem, where a set of vehicles aim to cooperatively determine minimal-length paths to satisfy all the pickup and delivery requests. The problem is recast as a constraint-coupled MILP in order to apply the methods of Chapter 3. We show that in this scenario the distributed algorithm can be simplified by suitably exploiting the problem structure. Then, we perform extensive simulations to show how the algorithm behaves under different circumstances. Finally, we perform a demonstrative laboratory experiment on the ChoiRbot platform with ground and aerial robots executing the proposed distributed algorithm. The results of this chapter are based on [28, 48, 115]

Appendix A provides an overview of some basic concepts on optimization.

Chapter 1

Distributed Constraint-Coupled Optimization

In this chapter, we introduce the distributed optimization framework studied throughout the thesis. We first review some basic concepts of graph theory and introduce the distributed computation model. Then, we formalize the constraint-coupled optimization set-up and discuss some example applications.

1.1 Graph Theory and Distributed Computation Model

In a distributed scenario, we consider N units, called *agents* or *processors*, that have both communication and computation capabilities. We assume a message-passing communication paradigm, i.e. agents can pass information to each other by sending packets of information.

The main tool used to model communication among agents is graph theory. A graph consists of a set of nodes, which represent the agents, and a set of edges, which represent the communication links among the agents. Formally, we define a graph $\mathcal{G}$ as the ordered pair $(V, \mathcal{E})$, where $V = \{1, \ldots, N\}$ and $\mathcal{E} \subseteq V \times V$. The elements in V are called vertices (or nodes), while the elements in $\mathcal{E}$ are called edges and are of the type (i, j) with $i, j \in V$. It is possible to distinguish between *undirected* graphs, in which edges are not oriented (i.e. graphs such that $(i, j) \in \mathcal{E}$ iff $(j, i) \in \mathcal{E}$), and *directed* graphs (also called *digraphs*), in which edges are oriented (thus an edge (i, j) can belong to $\mathcal{E}$ even though the reverse edge (j, i) does not). An example of a directed and of an undirected graph is depicted in Figure 1.1. Given an edge $(i, j) \in \mathcal{E}$, i is called *in-neighbor* of j and j is an *out-neighbor* of i. For each agent i, we define the in-neighbor set as $\mathcal{N}_i^{\text{IN}} = \{j \in V : (j, i) \in \mathcal{E}\}$ and the out-neighbor set as $\mathcal{N}_i^{\text{OUT}} = \{j \in V : (i, j) \in \mathcal{E}\}$. If the graph is undirected, we simply say that i is a *neighbor* of j (and viceversa), and denote $\mathcal{N}_i = \mathcal{N}_i^{\text{IN}} = \mathcal{N}_i^{\text{OUT}}$ as the *neighbor* set. A (di)graph $\mathcal{G}$ can be entirely represented in terms of its *adjacency matrix* $A \in \mathbb{R}^{N \times N}$,

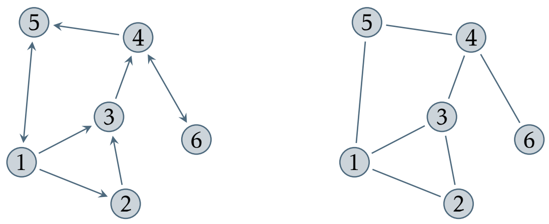


Figure 1.1: A directed (left) and an undirected (right) graph of $N = 6$ nodes.

where each entry (i, j) is 1 if $(i, j) \in \mathcal{E}$ and 0 otherwise. Given an undirected graph $\mathcal{G}$, we define also the *Laplacian matrix* $L \in \mathbb{R}^{N\times N}$, where each entry (i, j) is equal to

$$L_{ij} = \begin{cases} |\mathcal{N}_i| & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{1.1}$$

where $|\mathcal{N}_i|$ is the cardinality of $\mathcal{N}_i$.

The previous definitions hold true for static (time-invariant) graphs. In a similar way, it is possible to define time-dependent graphs. We say that $\mathcal{G}^t = (V, \mathcal{E}^t)$ is a (directed) time-varying graph, where $t \in \mathbb{N}$ is the time index, V is the set of vertices and $\mathcal{E}^t \subseteq V \times V$ is the set of edges at time t for all $t \geq 0$. For each agent i, we define the time-varying in-neighbor set as $\mathcal{N}_i^{\text{IN},t} = \{j \in V : (j,i) \in \mathcal{E}^t\}$ for all $t \geq 0$ and similarly for out-neighbors. As in the case of static graphs, for time-varying graphs we can define a time-varying adjacency matrix and a time-varying Laplacian matrix. A graphical representation of a time-varying network is given in Figure 1.2.

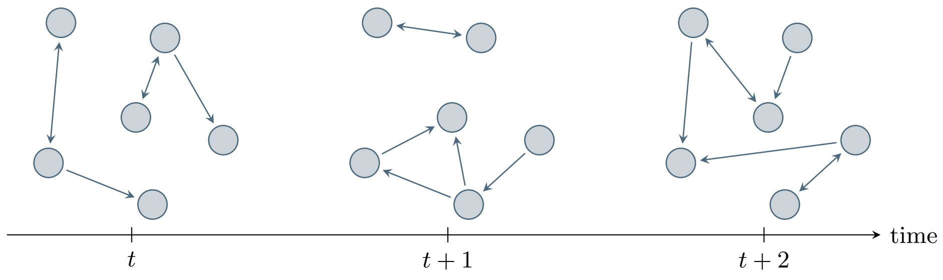


Figure 1.2: A directed time-varying graph of $N = 6$ nodes.

In the distributed scenario, we assign to each agent in the network a fixed identifier $i \in V$ from the set of vertices and we say that the edge (i, j) belongs to the communication graph at time t if $(i, j) \in \mathcal{E}^t$, which means that at time t agent i can send information to agent j. If the graph is undirected and $(i, j) \in \mathcal{E}^t$, then agents i and j can both exchange information with each other at time t. In a distributed algorithm, agents initialize their local states and then start an iterative procedure by interleaving computation and communication with neighboring units, with all the nodes performing the same actions. In particular, local states are updated by using only information received by in-neighbors.

Given a fixed graph $\mathcal{G}$, the following connectivity properties can be stated.

Definition 1.1. *A fixed directed graph $\mathcal{G}$ is said to be* strongly connected *if for every pair of nodes (i, j) there exists a path of directed edges that goes from i to j. If $\mathcal{G}$ is undirected, we say that $\mathcal{G}$ is* connected. $\triangle$

Connectivity properties can be also stated for time-varying topologies (we only consider directed graphs).

Definition 1.2. *A time-varying directed graph $\mathcal{G}^t = (V, \mathcal{E}^t)$, $t \in \mathbb{N}$, is said to be*

- jointly strongly connected *if the graph $\mathcal{G}^t_\infty \triangleq (V, \mathcal{E}^t_\infty)$, with $\mathcal{E}^t_\infty = \bigcup_{\tau=t}^{\infty} \mathcal{E}^\tau$, is strongly connected for all $t \geq 0$.*
- T-strongly connected *(or* uniformly jointly strongly connected*) if there exists a scalar $T > 0$ such that the graph $\mathcal{G}^t_T \triangleq (V, \mathcal{E}^t_T)$ with $\mathcal{E}^t_T = \bigcup_{\tau=0}^{T-1} \mathcal{E}^{t+\tau}$, is strongly connected for every $t \geq 0$.* $\triangle$

Given a network topology, agents can run distributed algorithms according to several communication protocols. When the steps of the algorithm explicitly depend on the value of t, we say that the algorithm is *synchronous*, i.e. agents must be aware of the current value of t and, thus, their local operations must be synchronized to a global clock. We will also consider a communication protocol in which agents are not aware of any global time information, i.e. their updates do not depend on t, and we term these algorithms *asynchronous*. In fact, if a distributed algorithm is designed to run over a jointly strongly connected graph and the local computation steps do not depend on t, then the algorithm can be also implemented in an asynchronous network.

1.2 Constraint-coupled Optimization Set-up

In this thesis we consider a distributed framework in which agents cooperatively aim to solve an optimization problem. Let us now formalize the distributed optimization set-up. We deal with a network of N agents that must solve a *constraint-coupled* optimization problem, which can be stated as

$$\begin{aligned} \min_{x_1,\ldots,x_N} \;& \sum_{i=1}^{N} f_i(x_i) \\ \text{subj. to} \;& \sum_{i=1}^{N} g_i(x_i) \leq b, \\ & x_i \in X_i, \qquad i = 1, \ldots, N, \end{aligned} \tag{1.2}$$

where $x_1, \ldots, x_N$ are the decision variables with each $x_i \in \mathbb{R}^{n_i}$, $n_i \in \mathbb{N}$. Moreover, for all $i \in \{1, \ldots, N\}$, the function $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ depends only on x_i, $X_i \subset \mathbb{R}^{n_i}$ is the constraint set associated to x_i and $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^S$ is the i-th contribution to the (vector-valued) *coupling* constraint, with $b \in \mathbb{R}^S$. Throughout the thesis, we stick to the convention that inequalities between vectors (e.g. $a \leq b$ with $a, b \in \mathbb{R}^n$) mean that the inequality is satisfied for each component of the vector (i.e. $a_i \leq b_i$ for all $i \in \{1, \ldots, n\}$). We term Problem (1.2) "constraint coupled" because, if the coupling constraint were not present, the problem would trivially split into N independent problems of the form

$$\min_{x_i \in X_i} \; f_i(x_i),$$

therefore the only coupling among the agents is given by the constraint $\sum_{i=1}^N g_i(x_i) \leq b$.

In the considered distributed computation framework, the problem data are assumed to be scattered throughout the network. Agents have only a partial knowledge of Problem (1.2) and must cooperate with each other in order to find a solution. Specifically, each agent i is assumed to know only its local constraint X_i, its local cost f_i and its own contribution g_i to the coupling constraint, together with the right-hand side b, and is only interested in computing its own block $x_i^\star$ of an optimal solution $(x_1^\star, \ldots, x_N^\star)$ of Problem (1.2).

Throughout the dissertation we will consider several variants of Problem (1.2), however we will always stick to the described constraint-coupled structure. In Chapter 2, we start by considering convex instances of Problem (1.2), where both the cost functions f_i and the constraints X_i and g_i are assumed to be convex. Then, we will focus on more complex scenarios where the convexity assumption is dropped. A particular emphasis will be given to a *Mixed-Integer Linear Programming* (MILP) version of Problem (1.2), which takes the form

$$\begin{aligned} \min_{x_1, \ldots, x_N} \; & \sum_{i=1}^N c_i^\top x_i \\ \text{subj. to} \; & \sum_{i=1}^N A_i x_i \leq b \\ & x_i \in X_i^{\text{MILP}}, \qquad i = 1, \ldots, N, \end{aligned} \tag{1.3}$$

where, for all $i \in \{1, \ldots, N\}$, the decision vector x_i has $p_i + q_i$ components (thus $c_i \in \mathbb{R}^{p_i + q_i}$) with $p_i, q_i \in \mathbb{N}$ and the local constraint set is of the form

$$X_i^{\text{MILP}} = P_i \cap (\mathbb{Z}^{p_i} \times \mathbb{R}^{q_i}),$$

for some nonempty compact polyhedron $P_i \subset \mathbb{R}^{p_i + q_i}$. Note that Problem (1.3) has the constraint-coupled structure as Problem (1.2), however in Problem (1.3) the constraint sets X_i^{MILP} are assumed to be mixed-integer. As such, Problem (1.3) is NP-hard and hence much more challenging to solve than convex instanes of Problem (1.2).

1.3 Application Frameworks

Let us show some application frameworks of the optimization set-up (1.2) for control problems over networks, namely optimization-based distributed control, distributed task negotiation and scheduling over networks, distributed planning for multi vehicles.

1.3.1 Cooperative Distributed Model Predictive Control

Model Predictive Control (MPC) is a widely studied technique in the control community, and is also used in distributed contexts. The goal is to design an optimization-based feedback control law for a (spatially distributed) network of dynamical systems. The leading idea is the principle of *receding horizon* control, which informally speaking consists of solving at each time step an optimization problem (usually termed *optimal control problem*), in which the system model is used to predict the system trajectory over a fixed time window and to determine an optimal input trajectory. After an optimal solution of the optimal control problem is found, the input associated to the current time instant is applied and the process is repeated (for a survey on MPC methods, see, e.g. [96]). A pictorial representation of this method is given in Figure 1.3.

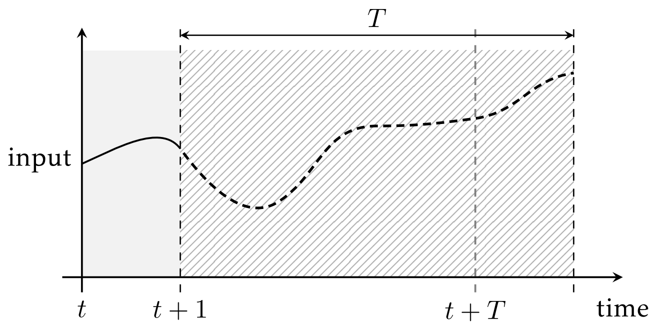


Figure 1.3: Illustration of the receding horizon principle. At time t, an optimal control problem over the time interval $[t, t+T]$ is solved. After applying the first sample of the resulting input trajectory, the horizon is shifted of one time unit and the process is repeated.

Now, we describe a typical distributed MPC framework applied to a network of linear systems with linear coupling constraints. Formally, assume we have N discrete-time linear dynamical systems with independent dynamics of the form $z_i(t+1) = A_i z_i(t) + B_i u_i(t)$, where $t \in \mathbb{Z}$ represents time, $z_i(t) \in \mathbb{R}^{q_i}$ is the system state at time t, $u_i(t) \in \mathbb{R}^{m_i}$ is the input fed to the system at time t and A_i, B_i are given matrices of appropriate dimensions, for all $i \in \{1, \dots, N\}$. We suppose that the states and the inputs must satisfy local constraints $z_i(t) \in Z_i$ and $u_i(t) \in U_i$ for all $i \in \{1, \ldots, N\}$ and $t \geq 0$, and that the agents' states are coupled to each other by means of coupling constraints of the form $\sum_{i=1}^{N} H_i z_i(t) \leq h$, for given matrices $H_i \in \mathbb{R}^{P \times q_i}$ and a given $h \in \mathbb{R}^P$. At time $t = 0$, given the initial conditions of the systems $z_1^0, \ldots, z_N^0$, the optimal control problem to be solved at the current time instant can be formulated as

$$\begin{aligned}
\min_{\substack{z_1,\ldots,z_N \\ u_1,\ldots,u_N}} \quad & \sum_{i=1}^{N} \bigg(\sum_{\tau=0}^{T-1} \ell_i(z_i(\tau), u_i(\tau)) + V_i(z_i(T)) \bigg) \\
\text{subj. to} \quad & z_i(\tau+1) = A_i z_i(\tau) + B_i u_i(\tau), && \tau = 0, \ldots, T-1, \quad \forall\, i, \\
& z_i(\tau+1) \in Z_i, u_i(\tau) \in U_i, && \tau = 0, \ldots, T-1, \quad \forall\, i, \\
& z_i(0) = z_i^0, && \qquad\qquad\qquad\quad\; \forall\, i, \\
& \sum_{i=1}^{N} H_i z_i(\tau) \leq h, && \tau = 1, \ldots, T,
\end{aligned} \tag{1.4}$$

where τ represent future time instants and T is the *prediction horizon*, $z_i = (z_i(0), \ldots, z_i(T))$ and $u_i = (u_i(0), \ldots, u_i(T-1))$ are the optimization vectors, $\ell_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$ is the *stage cost* and $V_i : \mathbb{R}^{q_i} \to \mathbb{R}$ is the *terminal cost*, for all $i \in \{1, \ldots, N\}$. Problem (1.4) can be fit into the constraint-coupled set-up (1.2) by setting

$$f_i(x_i) = \sum_{\tau=0}^{T-1} \ell_i(z_i(\tau), u_i(\tau)) + V_i(z_i(T)), \qquad g_i(x_i) = \begin{bmatrix} H_i z_i(1) \\ \vdots \\ H_i z_i(T) \end{bmatrix},$$

for all $i \in \{1, \ldots, N\}$, and $b = \mathbf{1} \otimes h$, where $\mathbf{1} \in \mathbb{R}^T$ is the vector of ones and $\otimes$ denotes the Kronecker product. Moreover, we define the local optimization vector as $x_i = (z_i, u_i) \in \mathbb{R}^{(T+1)q_i + Tm_i}$ and the local constraint set as

$$X_i \triangleq \Big\{ (z_i, u_i) \mid \; z_i(\tau+1) = A_i z_i(\tau) + B_i u_i(\tau), \; z_i(\tau+1) \in Z_i, \; u_i(\tau) \in U_i, \; \forall\, \tau \Big\},$$

for all $i \in \{1, \ldots, N\}$. In a distributed context, the goal for each agent i is to compute its block $(z_i^\star, u_i^\star)$ of optimal solution and to apply the first input $u_i^\star(0)$.

A practical application taking the form of Problem (1.4) will be considered in Chapter 4. Specifically, we will consider a more complex instance of Problem (1.4) where the goal is to control a microgrid modeled using stochastic mixed-integer approaches.

1.3.2 Distributed Task Assignment over Networks

Another important application is the so-called task assignment, a building block for decision making problems in which a certain number of agents must be assigned given tasks. The goal is to find the best matching of agents and tasks according to a given performance criterion.

Assume that N agents must be assigned N tasks with a one-to-one assignment. Define the variable $x_{i\kappa} \in \mathbb{R}$, which is 1 if agent i is assigned to task κ and 0 otherwise. Also, define the set E_A, which contains the tuple (i,κ) if agent i can be assigned to task κ. Finally, let $c_{i\kappa}$ be the cost occurring if agent i is assigned to task κ. In Figure 1.4, we show an illustrative example of the set-up. The task assignment problem can be formulated as the linear program

$$\begin{aligned} \min_{x} \quad & \sum_{(i,\kappa)\in E_A} c_{i\kappa} x_{i\kappa} \\ \text{subj. to} \quad & \mathbf{0} \le x \le \mathbf{1}, \\ & \sum_{\{\kappa \mid (i,\kappa)\in E_A\}} x_{i\kappa} = 1, \qquad i = 1,\dots,N, \\ & \sum_{\{i \mid (i,\kappa)\in E_A\}} x_{i\kappa} = 1, \qquad \kappa = 1,\dots,N, \end{aligned} \tag{1.5}$$

where x is the variable stacking all $x_{i\kappa}$ and $\mathbf{0}$ and $\mathbf{1}$ are the vectors of zeros and ones of appropriate dimensions. If Problem (1.5) is feasible, it can be shown that it admits an optimal solution such that $x_{i\kappa} \in \{0,1\}$ for all $(i,\kappa) \in E_A$ (this is referred to the so-called unimodularity property, see e.g. [9]).

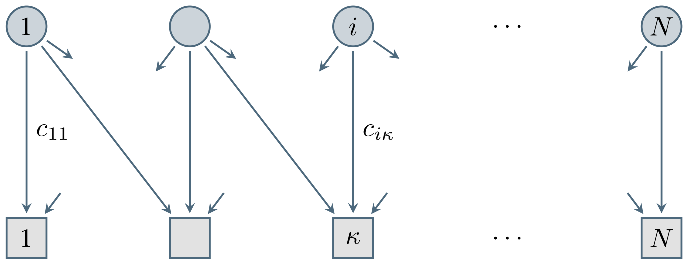


Figure 1.4: Graphical representation of the task assignment problem. Agents are represented by circles, while tasks are represented by squares. An arrow from agent i to task κ means that agent i can perform task κ (i.e. $(i,\kappa) \in E_A$). If agent i is assigned to task κ, the incurred cost is $c_{i\kappa}$.

Problem (1.5) can be cast in the constraint-coupled form (1.2). To see this, let us define $K_i \triangleq \big|\{\kappa : (i,\kappa) \in E_A\}\big|$ as the number of tasks that agent i can perform. We assume that agent i deals with the decision vector $x_i \in \mathbb{R}^{K_i}$, stacking the variables $x_{i\kappa}$ for all κ such that $(i,\kappa) \in E_A$. Then, the local sets X_i can be written as

$$X_i = \big\{ x_i \in \mathbb{R}^{K_i} \mid \mathbf{0} \le x_i \le \mathbf{1} \text{ and } x_i^\top \mathbf{1} = 1 \big\}, \qquad i = 1,\dots,N.$$

The coupling constraints can be written by defining, for all $i \in \{1,\dots,N\}$, the matrix $H_i \in \mathbb{R}^{N\times K_i}$, obtained by extracting from the $N \times N$ identity matrix the subset of columns corresponding to the tasks that agent i can perform. Problem (1.5) becomes

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^{N} c_i^\top x_i \\ \text{subj. to} \quad & x_i \in X_i, \qquad i = 1,\ldots,N, \\ & \sum_{i=1}^{N} H_i x_i = \mathbf{1}, \end{aligned}$$

where each c_i stacks the costs $c_{i\kappa}$ for all κ such that $(i,\kappa) \in E_A$. In a distributed context, the goal for each agent i is to compute its block $x_i^\star$ of optimal solution, which contains only one entry $x_{i\kappa} = 1$ corresponding to the task κ that agent i is eventually assigned.

A practical application taking the form of Problem (1.5) will be considered in Chapter 5. Specifically, we will a network of agents that must self-assign a set of pickup and delivery tasks. The problem will be then formulated using a more complex formalism arising in vehicle routing problems.

1.3.3 Distributed planning for multi vehicles

Finally, let us consider an application framework arising in cooperative robotics, where N robotic agents aim to cooperatively determine target positions to be reached. Specifically, assume the robots aim to compute optimal target positions $x_i^\star \in \mathbb{R}^n$ satisfying local constraints $X_i \subseteq \mathbb{R}^n$ while also satisfying constraints on some aggregate functions of the robot positions (as e.g. the center of mass), i.e., $\frac{1}{N}\sum_{i=1}^{N} g_i(x_i) \le b$. This task can be modeled as the convex constraint-coupled problem

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^{N} \|x_i - r_i\|_{Q_i}^2 \\ \text{subj. to} \quad & x_i \in X_i, \qquad i = 1,\ldots,N, \\ & \frac{1}{N}\sum_{i=1}^{N} g_i(x_i) \le b, \end{aligned} \tag{1.6}$$

where each $x_i \in \mathbb{R}^2$ is the local decision variable, $r_i \in \mathbb{R}^2$ is the ideal target position and $Q_i \in \mathbb{R}^{2\times 2}$ is a symmetric positive definite cost matrix.

We will consider a special instance of Problem (1.6) in Chapter 5, where we will consider a network of robots aiming to keep the barycenter of their positions within certain bounds.

1.4 Tour of Thesis Contributions

Throughout the thesis, we propose distributed algorithms for the constraint-coupled set-up (1.2) declined in a number of versions, with applications to several different scenarios. To ease the reading, in this section, we provide an overview of the considered challenges and proposed solutions, together with the corresponding chapters and sections. The scheme of the thesis contributions is reported in Figure 1.5, while in Table 1.1 we summarize the acronyms used in the following.

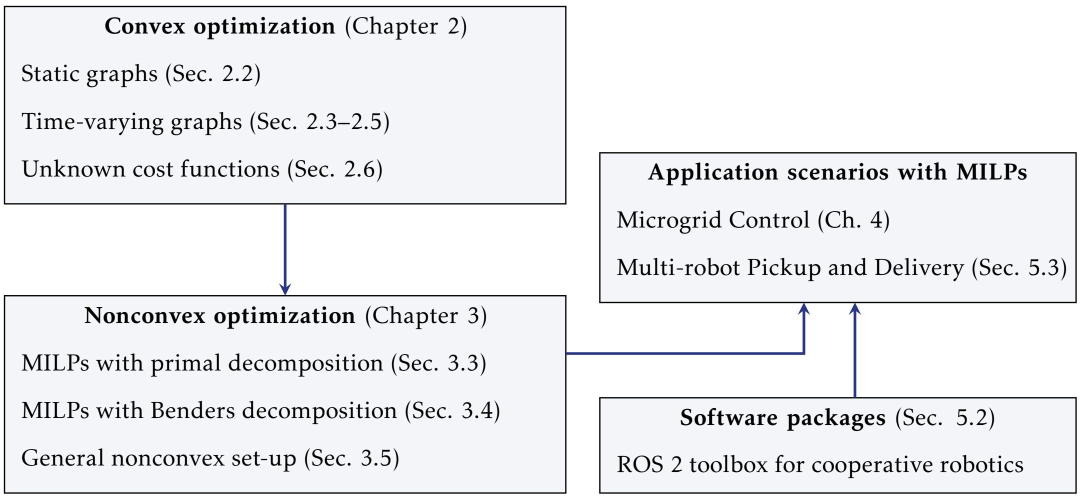

Convex optimization (Chapter 2)
Static graphs (Sec. 2.2)
Time-varying graphs (Sec. 2.3–2.5)
Unknown cost functions (Sec. 2.6)
Application scenarios with MILPs
Microgrid Control (Ch. 4)
Multi-robot Pickup and Delivery (Sec. 5.3)
Nonconvex optimization (Chapter 3)
MILPs with primal decomposition (Sec. 3.3)
MILPs with Benders decomposition (Sec. 3.4)
General nonconvex set-up (Sec. 3.5)
Software packages (Sec. 5.2)
ROS 2 toolbox for cooperative robotics

Figure 1.5: Overview of the thesis contributions

Table 1.1: List of the acronyms used in the thesis

Acronyms	
LP	Linear Program
MILP	Mixed-Integer Linear Program
MPC	Model Predictive Control
PDVRP	Pickup-and-Delivery Vehicle Routing Problem
ROS	Robot Operating System
GAN	Generative Adversarial (Neural) Network
MPI	Message Passing Interface
PEV	Plug-In Electric Vehicle

Chapter 2

Distributed Primal Decomposition for Convex Optimization

In this chapter, we focus on constraint-coupled convex problems and provide distributed algorithms based on the so-called *primal decomposition*. We begin by revisiting the basics of the primal decomposition approach and of a particular relaxation approach appeared in the literature. The application of these two methods to the constraint-coupled scenario will allow us to derive a parallel and a distributed algorithm for static communication graphs. We then extend the algorithm to the time-varying setting and provide a convergence analysis and a convergence rate analysis. To demonstrate the flexibility of the proposed approach, we further show how to handle more general set-ups and comment on the choice of the algorithm parameters. Numerical examples corroborate the theoretical results. Finally, we provide an extension of the algorithm with the assumption of a-priori unknown cost functions and show that this new algorithm inherits the convergence properties of the former. The results of this chapter are based on [20–22, 24].

2.1 Literature Review

The majority of the literature on distributed optimization has focused on a framework in which, differently from the constraint-coupled set-up, cost functions and constraints depend on the same, common decision variable, and agents aim for *consensual* optimal solutions. An exemplary, non-exhaustive list of works for this optimization set-up is [41, 58, 75, 81, 105, 106, 127]. Only recently has the constraint-coupled set-up gathered more attention from our community, due to its applicability to control scenarios. In [110], consensus-based dual decomposition is combined with a primal recovery mechanism, whereas [44] considers a distributed dual algorithm based on proximal minimization. In [83] a distributed algorithm based on successive duality steps is proposed. Differently from [44, 110], which employ running averages for primal recovery, the algorithm proposed in [83] can guarantee feasibility of primal iterates without averaging schemes. In [34], a consensus-based primal-dual perturbation algorithm is proposed to solve smooth constraint-coupled optimization problems. A distributed saddle-point algorithm with Laplacian averaging is proposed in [71] for a class of min-max problems. In [18], a distributed algorithm based on cutting planes is formulated. Recently, in [66] a primal-dual algorithm with constant step size is proposed under smoothness assumption of both costs and constraints. The works in [3,78,104] consider a similar set-up, but the proposed algorithms strongly rely on the sparsity pattern of the coupling constraints. Linear constraint-coupled problem set-ups have been also tackled by means of distributed algorithms based on the Alternating Direction Method of Multipliers (ADMM). In [33] the so-called consensus-ADMM is applied to the dual problem formulation, which is then tailored for an application in Model Predictive Control by [123]. In [29] an ADMM-based algorithm is proposed and analyzed using an operator theory approach while in [47, 126] augmented Lagrangian approaches equipped with a tracking mechanism are proposed. The analysis of the algorithm for random time-varying graphs builds on randomized block subgradient methods, therefore let us recall some related works from the centralized literature. A survey on block coordinate methods is given in [6], while a unified framework for nonsmooth problems can be found [97]. In [100], a randomized block coordinate descent method is formulated, whereas [39] investigates a stochastic block mirror descent approach with random block updates. In [77], a distributed algorithm for a linearly constrained problem is analyzed with coordinate descent methods. This technique is also used in [76], which considers a constraint-coupled problem. However, the approach used in [76, 77] only allow for a single pair of agents updating at a time and requires smooth cost functions.

2.2 Primal Decomposition Paradigm for Constraint-coupled Optimization

In this section, we review the primal decomposition method and a so-called relaxation approach for constraint-coupled convex optimization. We then provide a reinterpretation of existing distributed schemes in terms of primal decomposition. Part of the material of this section has appeared as [23, 24].

2.2.1 Distributed Convex Optimization Set-up

Let us consider a network of N agents that must solve the constraint-coupled problem

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subj. to} \quad & \sum_{i=1}^{N} g_i(x_i) \leq b, \\ & x_i \in X_i, \qquad i = 1,\ldots,N, \end{aligned} \tag{2.1}$$

where $x_1 \in \mathbb{R}^{n_1}, \ldots, x_N \in \mathbb{R}^{n_N}$ are the decision variables with each $n_i \in \mathbb{N}$. Moreover, for all $i \in \{1,\ldots,N\}$, $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is the objective function associated to x_i, $X_i \subset \mathbb{R}^{n_i}$ is the constraint set associated to x_i and $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^S$ is the i-th contribution to the coupling constraint, with $b \in \mathbb{R}^S$. We stick to the convention that inequalities for vectors are intended component wise.

We assume Problem (2.1) is feasible. Moreover, the following two assumptions guarantee that *(i)* the optimal cost of Problem (2.1) is finite and at least one optimal solution exists, *(ii)* duality arguments are applicable.

Assumption 2.1 (Convexity and compactness)**.** *For all $i \in \{1,\ldots,N\}$, the set X_i is convex and compact, the function f_i is convex and each component of g_i is a convex function.* △

Assumption 2.2 (Slater's constraint qualification)**.** *There exist $\bar{x}_1 \in X_1, \ldots, \bar{x}_N \in X_N$ such that $\sum_{i=1}^{N} g_i(\bar{x}_i) < b$.* △

If Problem (2.1) is a linear program, strong duality holds immediately and Assumption 2.2 is not necessary [10, Proposition 5.2.2].

We recall from Chapter 1 that in the considered distributed optimization framework the problem data are assumed to be scattered throughout the network. Specifically, each agent i is assumed to know only its local constraint X_i, its local cost f_i and its own contribution g_i to the coupling constraints, together with the right-hand side b, and is only interested in computing its own block $x_i^\star$ of an optimal solution $(x_1^\star, \ldots, x_N^\star)$ of Problem (2.1). In this section, agents are assumed to communicate according to a connected and undirected graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1,\ldots,N\}$ is the set of agent identifiers and $\mathcal{E} \subseteq V \times V$ is the set of edges. If $(i,j) \in \mathcal{E}$, then also $(j,i) \in \mathcal{E}$ and nodes i and j can exchange information. The set of neighbors of agent i in $\mathcal{G}$ is denoted as $\mathcal{N}_i = \{j \in \{1,\ldots,N\} \mid (i,j) \in \mathcal{E}\}$. In Section 2.3, we will consider a more general scenario with a randomized time-varying graph.

2.2.2 Review of Primal Decomposition

Let us now introduce the primal decomposition approach (or right-hand side allocation) [10,108]. This method is a powerful tool to recast constraint-coupled convex programs of the form (2.1) into a master-subproblem architecture.

The main idea of primal decomposition is to interpret the right-hand side vector b of the coupling constraint $\sum_{i=1}^{N} g_i(x_i) \leq b$ as a given, limited resource that must be shared among the agents. The right-hand side vector b represents the total amount of resource. Thus, for all $i \in \{1, \ldots, N\}$ we introduce local *allocation vectors*, denoted $y_i \in \mathbb{R}^S$, adding up to the total resource, i.e. satisfying $\sum_{i=1}^{N} y_i = b$. By introducing these auxiliary variables, Problem (2.1) can be equivalently restated as

$$\begin{aligned}
\min_{\substack{x_1,\ldots,x_N \\ y_1,\ldots,y_N}} \; & \sum_{i=1}^{N} f_i(x_i) \\
\text{subj. to } \; & g_i(x_i) \leq y_i, \qquad i = 1, \ldots, N, \\
& x_i \in X_i, \qquad i = 1, \ldots, N, \\
& \sum_{i=1}^{N} y_i = b.
\end{aligned}$$

The latter problem can be restructured into a hierarchical formulation[1], with the following *master problem*

$$\begin{aligned}
\min_{y_1,\ldots,y_N} \; & \sum_{i=1}^{N} p_i(y_i) \\
\text{subj. to } \; & \sum_{i=1}^{N} y_i = b \\
& y_i \in Y_i, \qquad i = 1, \ldots, N,
\end{aligned} \tag{2.2}$$

where, for each $i \in \{1, \ldots, N\}$, the function $p_i : Y_i \to \mathbb{R}$ assigns to each y_i the optimal cost of the i-th (parametric) *subproblem*

$$\begin{aligned}
p_i(y_i) \triangleq \min_{x_i} \; & f_i(x_i) \\
\text{subj. to } \; & g_i(x_i) \leq y_i, \\
& x_i \in X_i.
\end{aligned} \tag{2.3}$$

Note that the parameter y_i only appears in the constraints. The functions p_i are typically called *primal functions* (in contrast to the dual functions, see Appendix A.1) and are

[1] This is a consequence of the basic optimization property $\min_{x \in X, y \in Y} f(x,y) = \min_{y \in Y} \Big(\min_{x \in X} f(x,y) \Big)$.

used to study the effect of perturbations on the constraint $g_i(x_i) \leq y_i$. Under suitable assumptions, the primal functions can be shown to be convex (although not differentiable in general), from which it follows that the master problem is convex. In Problem (2.2), the additional constraint $Y_i \subseteq \mathbb{R}^S$, which is also the domain of the primal function, is the set of local allocations for which Problem (2.3) admits a feasible solution, i.e.

$$Y_i = \big\{ y_i \in \mathbb{R}^S \mid \exists\, x_i \in X_i \text{ such that } g_i(x_i) \leq y_i \big\}. \tag{2.4}$$

We do not discuss in detail the properties of the sets Y_i, which can be found e.g. in [108]. In Figure 2.1, we show a graphical representation of the primal decomposition scheme.

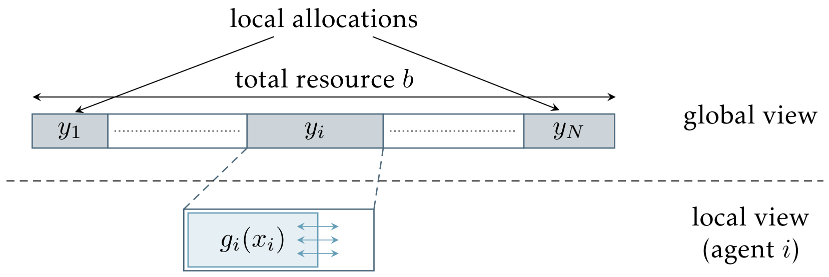


Figure 2.1: Illustration of the primal decomposition technique. Agents are assigned a local allocation vector y_i such that $\sum_{i=1}^N y_i = b$. Each agent i is then free to choose any x_i satisfying $g_i(x_i) \leq y_i$. For any such local solutions, the coupling constraint $\sum_{i=1}^N g_i(x_i) \leq b$ will be satisfied.

The following lemma formalizes the equivalence between Problem (2.1) and Problems (2.2)–(2.3).

Lemma 2.1 ([108, Lemma 1]). *Let Problem* (2.1) *be feasible. Then,*

(i) the optimal costs of Problems (2.1) *and* (2.2) *are equal;*

(ii) if $(y_1^\star, \ldots, y_N^\star)$ *is an optimal solution of* (2.2) *and, for all* i, $x_i^\star$ *is an optimal solution of* (2.3) *with* $y_i = y_i^\star$, *then* $(x_1^\star, \ldots, x_N^\star)$ *is an optimal solution of Problem* (2.1). △

For convenience, from now on we use the shorthand *optimal allocation* to refer to any optimal solution $(y_1^\star, \ldots, y_N^\star)$ of the master Problem (2.2). Note that, within the described framework, the local allocation $y_i^\star$ is the only information required for each agent i to compute its block $x_i^\star$ of an optimal solution of Problem (2.1). Thus, in the following we aim to derive parallel and distributed algorithms that compute an optimal allocation. The proposed algorithms will be mainly based on (sub)gradient methods applied to various reformulations of the master Problem (2.2). In order to compute subgradients, we will use the following result.

Lemma 2.2 ([10, Section 5.4.4]). *Let* $y_i \in Y_i$ *be given and let Problem* (2.3) *be feasible and have finite optimal cost. Moreover, let strong duality holds for Problem* (2.3). *Then, a* *subgradient of p_i at y_i, denoted $\widetilde{\nabla} p_i(y_i)$, can be computed as*

$$\widetilde{\nabla} p_i(y_i) = -\mu_i(y_i), \tag{2.5}$$

where $\mu_i(y_i)$ denotes a Lagrange multiplier associated to the constraint $g_i(x_i) \leq y_i$. $\triangle$

Note that Lagrange multipliers can be computed as dual optimal solutions of Problem (2.3) (cf. Appendix A.1). In order for the strong duality assumption to hold, we may assume Slater's constraint qualification for Problem (2.3). However, for certain values of the allocation vector $y_i \in Y_i$, there might not exist a $x_i \in X_i$ such that $g_i(x_i) < y_i$ (intuitively, this happens if y_i is on the boundary of Y_i). In such a case, we cannot guarantee strong duality and compute a subgradient of the primal function at y_i using Lemma 2.2. Nevertheless, in the next subsection we recall a method that efficiently handles this issue.

2.2.3 Review of Relaxation Approach

Although primal decomposition is a well-established framework, [10, 108], *fully distributed* approaches to address the general set-up (2.1) are missing. The presence of the constraints $y_i \in Y_i$ in Problem (2.2) (whose description in terms of inequalities is not available) makes its solution nontrivial. A distributed method to solve convex smooth versions of Problem (2.1) without the local constraints $x_i \in X_i$ (i.e. such that $Y_i \equiv \mathbb{R}^S$) has been investigated in [65]. In the recent work [83], a distributed methodology has been proposed to solve Problem (2.1). It consists in a relaxation approach applied to the original Problem (2.1) and a double duality step to apply dual decomposition. In the remainder of this section, we show that the algorithm proposed in [83] can be reinterpreted as a distributed scheme, based on primal decomposition, that circumvents the the constraints $y_i \in Y_i$. As this method represents an interesting alternative also for parallel (master-worker) architectures, in Section 2.2.4 we discuss a parallel algorithm.

Now, let us recall the relaxation approach of [83]. Formally, let $\rho_1, \ldots, \rho_N \geq 0$ be auxiliary scalar variables and consider the following *relaxed* version of Problem (2.1),

$$\begin{aligned}
\min_{\substack{x_1,\ldots,x_N,\\ \rho_1,\ldots,\rho_N}} \quad & \sum_{i=1}^{N} (f_i(x_i) + M\rho_i) \\
\text{subj. to} \quad & \sum_{i=1}^{N} g_i(x_i) \leq b + \sum_{i=1}^{N} \rho_i \mathbf{1}, \\
& x_i \in X_i, \;\; \rho_i \geq 0, \qquad i = 1, \ldots, N,
\end{aligned} \tag{2.6}$$

where $\mathbf{1} \in \mathbb{R}^S$ is the vector of ones and $M > 0$ is a parameter. The additional term $\sum_{i=1}^{N} \rho_i \mathbf{1}$ appearing in the right-hand side of the coupling constraint can be regarded as a (non-negative) violation, which is penalized in the cost by the term $M \sum_{i=1}^N \rho_i$. Indeed, for any locally feasible $x_1 \in X_1, \ldots, x_N \in X_N$, it is always possible to choose $\rho_1, \ldots, \rho_N \geq 0$ sufficiently large such that the relaxed version of the coupling constraint is satisfied. However, it turns out that by tuning appropriately the parameter M, such violations are zero at an optimal solution. We recall this fact in the next lemma, which uses an exact penalty function argument to establish the relationship between the optimal solutions of Problems (2.1) and (2.6).

Lemma 2.3 ([83, Proposition III.3]). *Let Problem* (2.1) *be feasible and let Assumptions 2.1 and 2.2 hold. Denote by $\mu^\star$ a Lagrange multiplier of Problem* (2.1) *associated to the constraint $\sum_{i=1}^N g_i(x_i) \leq b$ and assume $M > \|\mu^\star\|_1$. Then, a vector $(x_1^\star, \ldots, x_N^\star)$ is and optimal solution of* (2.1) *if an only if $(x_1^\star, \ldots, x_N^\star, 0, \ldots, 0)$ is an optimal solution of Problem* (2.6)*. Hence, the optimal solutions of* (2.6) *must have $\rho_i^\star = 0$ for all $i \in \{1, \ldots, N\}$.* △

For our purposes, from now on we can simply assume that M is large enough. In Section 2.3.4, we will discuss a mechanism to determine a suitable value of this parameter. To see the benefit of reformulating Problem (2.1) as (2.6), let us now apply the primal decomposition technique to Problem (2.6). The master problem becomes

$$\begin{aligned} \min_{y_1,\ldots,y_N} \; & \sum_{i=1}^N p_i(y_i) \\ \text{subj. to} \; & \sum_{i=1}^N y_i = b, \end{aligned} \tag{2.7}$$

where, for all $i \in \{1, \ldots, N\}$, the primal function $p_i : \mathbb{R}^S \to \mathbb{R}$ assigns each $y_i \in \mathbb{R}^S$ to the optimal cost of the i-th subproblem,

$$\begin{aligned} p_i(y_i) \triangleq \min_{x_i, \rho_i} \; & f_i(x_i) + M\rho_i \\ \text{subj. to} \; & g_i(x_i) \leq y_i + \rho_i \mathbf{1} \\ & x_i \in X_i, \;\; \rho_i \geq 0. \end{aligned} \tag{2.8}$$

Note that here the function $p_i(y_i)$ is formally different from the one defined in (2.3). However, to keep the notation light, we prefer not to introduce a new symbol. In the remainder of this chapter, we will always refer to the definition of $p_i(y_i)$ in (2.8).

The formulation (2.7) of the master problem is more convenient than the former one (2.2). Indeed, the new subproblem (2.8) is feasible for all $y_i \in \mathbb{R}^S$. Therefore, the constraints $y_i \in Y_i$ are not needed in Problem (2.7). Moreover, differently from Problem (2.3), Problem (2.8) enjoys strong duality[2] for all $y_i \in \mathbb{R}^S$ (indeed, the Slater assumption can be always satisfied by choosing a sufficiently large $\rho_i > 0$). Thus, we can

[2]Note that this does not mean that Assumption 2.2 can be dropped. Indeed, it is required for Lemma 2.3.

use Lemma 2.2 to compute a subgradient of the function p_i at *any* y_i. The combination of these two facts allows for the design of a parallel and a distributed algorithm, which would be far more difficult if performed directly on Problem (2.2).

Due to the equivalence of Problems (2.1) and (2.6) assessed by Lemma 2.3, it is natural to expect that the associated master problems have the same optimal solutions. This is formally proved next.

Lemma 2.4. *Let Problem* (2.1) *be feasible and let Assumptions 2.1 and 2.2 hold. Then, if* $M > \|\mu^\star\|_1$, *Problem* (2.7) *has the same optimal solutions and the same optimal cost of Problem* (2.2).

Proof. We only prove one direction since similar arguments can be used to prove the converse. Let $(\bar{y}_1, \dots, \bar{y}_N)$ be an optimal solution of Problem (2.7). We must show that this vector is feasible and cost-optimal for Problem (2.2). Clearly, $\sum_{i=1}^N \bar{y}_i = b$. In order to show that $\bar{y}_i \in Y_i$ for all i, recall that $(x_1^\star, \dots, x_N^\star)$ denotes an optimal solution of Problem (2.1). By Lemma 2.3, $(x_1^\star, \dots, x_N^\star, 0, \dots, 0)$ is an optimal solution of (2.6). Therefore, by Lemma 2.1, for all $i \in \{1, \dots, N\}$ the vector $(x_i^\star, 0)$ is an optimal solution of (2.8) with $y_i = \bar{y}_i$. Thus, by construction, $g_i(x_i^\star) \leq \bar{y}_i$, which means that $\bar{y}_i \in Y_i$ (cf. (2.4)). This proves that $(\bar{y}_1, \dots, \bar{y}_N)$ is feasible for Problem (2.2). To prove that $(\bar{y}_1, \dots, \bar{y}_N)$ is cost-optimal, simply notice that, by the preceding arguments, it holds $\sum_{i=1}^N p_i(\bar{y}_i) = \sum_{i=1}^N f_i(x_i^\star)$, which is the optimal cost of (2.1). The proof follows by applying Lemma 2.1 *(i)*. ■

Operatively, in order to have a valid value of M and to apply the algorithmic approaches to be described shortly, one can easily obtain a conservative estimate by following the approach detailed in Section 2.3.4. Next, we describe a parallel and a distributed algorithm to solve Problem (2.1) that rely on (2.7)–(2.8).

2.2.4 Parallel Primal Decomposition Algorithm

To derive a parallel algorithm to solve Problem (2.1) via primal decomposition, let us consider a subgradient method applied to Problem (2.7). Formally, let us denote by $y \in \mathbb{R}^{NS}$ the vector stacking all the vectors y_i and by $p(y) = \sum_{i=1}^N p_i(y_i)$ the cost function of Problem (2.7). A projected subgradient method applied to Problem (2.7) reads

$$y^{t+1} = \mathcal{P}_{\mathcal{Y}}\big(y^t - \alpha^t \widetilde{\nabla} p(y^t)\big), \tag{2.9}$$

where $t \in \mathbb{N}$ is the iteration index, α^t is the step size, $\widetilde{\nabla} p(y^t)$ denotes a subgradient of p at y^t and $\mathcal{P}_{\mathcal{Y}}(\cdot)$ denotes the Euclidean projection onto $\mathcal{Y}$, the feasible set of Problem (2.7),

$$\mathcal{Y} \triangleq \left\{ (y_1, \dots, y_N) \;\Big|\; \sum_{i=1}^N y_i = b \right\}.$$

The projection step admits a closed-form solution. Given $y \in \mathbb{R}^{NS}$, it can be shown using the optimality conditions of the projection problem that

$$\mathcal{P}_{\mathcal{Y}}(y) = \begin{bmatrix} y_1 - (\sum_{j=1}^{N} y_j - b)/N \\ \vdots \\ y_N - (\sum_{j=1}^{N} y_j - b)/N \end{bmatrix}.$$

Also, notice that $\widetilde{\nabla} p(y^t)$ is obtained as the stack of $\widetilde{\nabla} p_i(y_i^t)$. According to Lemma 2.2, a subgradient of p_i at a given y_i^t can be computed as

$$\widetilde{\nabla} p_i(y_i^t) = -\mu_i^t, \tag{2.10}$$

where μ_i^t is a Lagrange multiplier of Problem (2.8), with $y_i = y_i^t$, associated to the allocation constraint $g_i(x_i) \leq y_i^t + \rho_i \mathbf{1}$.

Therefore, the parallel primal decomposition algorithm reads as follows. Each agent i maintains a variable y_i^t. At each iteration t, each agent i computes a Lagrange multiplier μ_i^t of Problem (2.8) corresponding to y_i^t. The master node receives μ_i^t for $i \in \{1, \dots, N\}$ and sends back to each agent the average $\mu_{\text{avg}}^t = 1/N \sum_{j=1}^{N} \mu_j^t$. Then, each agent i updates its local allocation vector y_i^t according to (2.9), or, equivalently,

$$\begin{aligned} y_i^{t+1} &= y_i^t + \alpha^t \mu_i^t - \frac{1}{N}\Big(\sum_{j=1}^{N} (y_j^t + \alpha^t \mu_j^t) - b \Big) \\ &\overset{(a)}{=} y_i^t + \alpha^t \big(\mu_i^t - \mu_{\text{avg}}^t \big), \end{aligned} \tag{2.11}$$

where (a) follows by $\sum_{j=1}^{N} y_j^t = b$ and the definition of μ_{avg}^t. A pictorial representation of the parallel algorithm is provided in Figure 2.2. Note that, in Figure 2.2, the communication of information between agents and the master node is achieved through an ideal communication network, whereas in a distributed setting the network topology is given. We do not provide an explicit convergence result of the parallel algorithm (2.11) as it is analogous to the result for the forthcoming distributed algorithm given in Proposition 2.1.

2.2.5 Distributed Primal Decomposition for fixed graphs

In order to solve the master problem (2.7) in a distributed way, we draw inspiration from the parallel update (2.11). With a simple manipulation, the update (2.11) can be rewritten as

$$y_i^{t+1} = y_i^t + \frac{\alpha^t}{N} \sum_{j=1}^{N} \big(\mu_i^t - \mu_j^t \big).$$

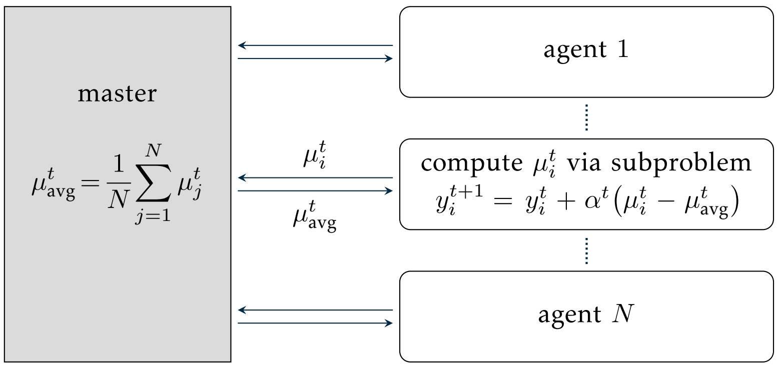

master
$\mu_{\text{avg}}^t = \frac{1}{N}\sum_{j=1}^{N} \mu_j^t$
agent 1
μ_i^t
μ_{avg}^t
compute μ_i^t via subproblem
$y_i^{t+1} = y_i^t + \alpha^t(\mu_i^t - \mu_{\text{avg}}^t)$
agent N

Figure 2.2: Illustration of the parallel primal decomposition algorithm. Each agent i computes μ_i^t as a Lagrange multiplier of Problem (2.8) and sends it to the master node. The master computes the average and sends back μ_{avg}^t, which is used by agents to compute the new allocation y_i^{t+1}.

To obtain a distributed protocol we let each agent i iteratively perform a modified version of the update, where the sum of the terms $(\mu_i^t - \mu_j^t)$ is limited to the neighbors $j \in \mathcal{N}_i$ instead of all the network $j \in \{1, \dots, N\}$ (which corresponds to all-to-all communication), and we absorb the scaling factor $1/N$ in the definition of the step size.

The Distributed Primal Decomposition algorithm that we propose reads as follows. Each agent i maintains a variable y_i^t. At each iteration t, agents compute μ_i^t as a Lagrange multiplier of the optimization problem

$$\begin{aligned} \min_{x_i, \rho_i} \quad & f_i(x_i) + M\rho_i \\ \text{subj. to} \quad \mu_i : \; & g_i(x_i) \le y_i^t + \rho_i \mathbf{1} \\ & x_i \in X_i, \;\; \rho_i \ge 0, \end{aligned} \tag{2.12}$$

where the notation "μ_i :" means that the Lagrange multiplier to be computed is associated with the constraint $g_i(x_i) \le y_i^t + \rho_i \mathbf{1}$. Then, agents exchange μ_j^t with neighbors $j \in \mathcal{N}_i$ and update their local allocation with

$$y_i^{t+1} = y_i^t + \alpha^t \sum_{j \in \mathcal{N}_i} \big(\mu_i^t - \mu_j^t\big), \qquad i = 1, \dots, N, \tag{2.13}$$

where α^t is the step size. The distributed algorithm architecture is depicted in Figure 2.3.

We note that the parallel update (2.11) and the distributed update (2.13) preserve the sum of y_i, i.e. $\sum_{i=1}^N y_i^{t+1} = \sum_{i=1}^N y_i^t$ for all t. This fact maintains feasibility of the iterates for Problem (2.7) provided that the allocation vectors at initialization are feasible, i.e. $\sum_{i=1}^N y_i^0 = b$. The update (2.13) has a sparse structure matching the communication graph, and reduces to (2.11) only in the particular case of complete graph.

We now formalize the convergence result of the Distributed Primal Decomposition

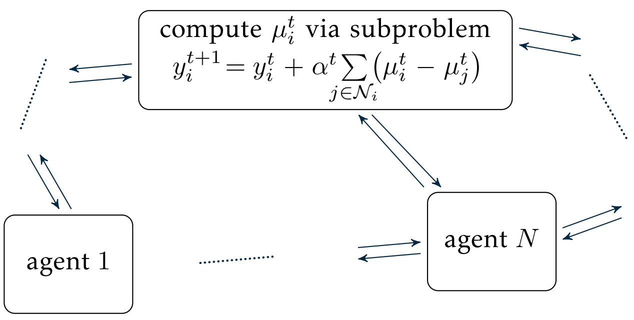

compute μ_i^t via subproblem
$y_i^{t+1} = y_i^t + \alpha^t \sum_{j \in \mathcal{N}_i} (\mu_i^t - \mu_j^t)$
agent 1
agent N

Figure 2.3: Architecture of the Distributed Primal Decomposition algorithm. Each agent i computes μ_i^t as a Lagrange multiplier of (2.12) and sends it to neighbors. Then, it updates y_i^{t+1} based on the received multipliers μ_j^t for $j \in \mathcal{N}_i$.

algorithm. As we already mentioned, we obtain this result by reinterpreting the algorithm proposed in [83]. We will consider the auxiliary sequence $\{(x_i^t, \rho_i^t)\}_{t \geq 0}$, where for all $i \in \{1, \dots, N\}$ and $t \geq 0$ the vector (x_i^t, ρ_i^t) is an optimal solution of Problem (2.12). We make the following assumption on the step-size sequence.

Assumption 2.3. *The step-size sequence $\{\alpha^t\}_{t \geq 0}$, with each $\alpha^t \geq 0$, satisfies $\sum_{t=0}^{\infty} \alpha^t = \infty$ and $\sum_{t=0}^{\infty} \big(\alpha^t\big)^2 < \infty$.* △

Assumption 2.3 is standard in the stochastic approximation literature, and can also be found more recently in the distributed optimization literature, see e.g. [44]. The following proposition summarizes the convergence properties of the distributed algorithm (2.12)–(2.13).

Proposition 2.1. *Let Problem* (2.1) *be feasible and let Assumptions 2.1, 2.2 and 2.3 hold. Moreover, let the local allocation vectors y_i^0 be initialized such that $\sum_{i=1}^N y_i^0 = b$ and let $M > \|\mu^\star\|_1$. Then, the distributed algorithm* (2.13) *generates an allocation vector sequence $\{y_1^t, \dots, y_N^t\}_{t \geq 0}$ and a primal sequence $\{x_1^t, \dots, x_N^t\}_{t \geq 0}$ such that*

(i) $\sum_{i=1}^N y_i^t = b$ *for all* $t \geq 0$;

(ii) $\lim_{t \to \infty} \|y_i^t - y_i^\star\| = 0$ *for all $i \in \{1, \dots, N\}$, where $(y_1^\star, \dots, y_N^\star)$ is an optimal solution of Problem* (2.2);

(iii) every limit point of $\{x_1^t, \dots, x_N^t\}_{t \geq 0}$ is an optimal solution of Problem (2.1).

Proof. We show that algorithm (2.13) can be recast to the algorithm in [83] and rely on the convergence result provided in [83]. Let us first recall the algorithm in [83], which reads as follows. Each agent i maintains variables λ_{ij}^t for $j \in \mathcal{N}_i$. At each iteration, each agent i gathers λ_{ji}^t from neighbors $j \in \mathcal{N}_i$ and compute $\big((x_i^t, \rho_i^t), \mu_i^t\big)$ as a primal-dual optimal solution pair of

$$\begin{aligned} \min_{x_i,\rho_i} \quad & f_i(x_i) + M\rho_i \\ \text{subj. to} \quad & \mu_i : g_i(x_i) - \frac{b}{N} + \sum_{j\in\mathcal{N}_i} \big(\lambda_{ij}^t - \lambda_{ji}^t\big) \le \rho_i \mathbf{1} \\ & x_i \in X_i, \;\; \rho_i \ge 0. \end{aligned} \tag{2.14}$$

Then, they gather μ_j^t from $j \in \mathcal{N}_i$ and update λ_{ij}^t with

$$\lambda_{ij}^{t+1} = \lambda_{ij}^t - \gamma^t\big(\mu_i^t - \mu_j^t\big) \qquad \forall\, j \in \mathcal{N}_i, \tag{2.15}$$

where γ^t is the step size. We now show that the update (2.15) is equivalent to (2.13) up to a change of coordinates. To this end, let us define for all $t \ge 0$

$$y_i^t \triangleq - \sum_{j\in\mathcal{N}_i} \big(\lambda_{ij}^t - \lambda_{ji}^t\big) + \frac{b}{N}, \qquad i = 1, \ldots, N. \tag{2.16}$$

Then, it holds

$$\sum_{i=1}^N y_i^t = \underbrace{\sum_{i=1}^N \sum_{j\in\mathcal{N}_i} \big(\lambda_{ji}^t - \lambda_{ij}^t\big)}_{=\mathbf{0}} + \sum_{i=1}^N \frac{b}{N} = b \qquad \text{for all } t \ge 0,$$

which follows since the graph is undirected. This motivates the assumption $\sum_{i=1}^N y_i^0 = b$ and proves *(i)*.

To prove *(ii)*, we simply note that the update of y_i^t, as defined in (2.16), reads

$$\begin{aligned} y_i^{t+1} &= \frac{b}{N} + \sum_{j\in\mathcal{N}_i} \big(\lambda_{ji}^{t+1} - \lambda_{ij}^{t+1}\big) \\ &= \frac{b}{N} + \sum_{j\in\mathcal{N}_i} \big(\lambda_{ji}^t - \lambda_{ij}^t\big) + 2\gamma^t \sum_{j\in\mathcal{N}_i} \big(\mu_i^t - \mu_j^t\big) \\ &= y_i^t + 2\gamma^t \sum_{j\in\mathcal{N}_i} \big(\mu_i^t - \mu_j^t\big), \qquad i = 1, \ldots, N, \end{aligned}$$

where we point out that each μ_i^t is a dual optimal solution of (2.14), or, equivalently, a Lagrange multiplier of Problem (2.8) with $y_i = y_i^t$. Then, by defining the step-size sequence $\alpha^t \triangleq 2\gamma^t$, we see that the update (2.15) coincides with (2.13). As proven in [83], the sequence $\{(\lambda_{ij}^t)_{(i,j)\in\mathcal{E}}\}_{t\ge 0}$ converges to an optimal solution $(\lambda_{ij}^\star)_{(i,j)\in\mathcal{E}}$ of a suitable dual reformulation of the original problem (2.1) with the same optimal cost. Therefore, the sequence $\{y_1^t, \ldots, y_N^t\}$ converges to some $(\bar{y}_1, \ldots, \bar{y}_N)$, which is feasible for Problem (2.7) (by point *(i)* of this proposition). Moreover, as proven in [83], the local problem (2.14) at $(\lambda_{ij}^\star)_{(i,j)\in\mathcal{E}}$ (which is the same as problem (2.12) at $\bar{y}_i$) have optimal cost $f_i(x_i^\star) = p_i(y_i^\star)$, from which it follows that $\bar{y}_i = y_i^\star$ for all i. This concludes the proof of *(ii)*. As Problem (2.14) is equivalent to (2.8) with $y_i = y_i^t$, part *(iii)* follows by [83, Theorem 2.4 (ii)]. ■

2.3 Distributed Primal Decomposition over Random Time-varying Networks

In this section, we propose a new distributed algorithm based on primal decomposition to solve Problem (2.1) over time-varying networks. First, we describe the time-varying network model. Then, we introduce the distributed algorithm and discuss some extensions and the choice of the parameters. The results of this section and of Sections 2.4 and 2.5 are based on [21, 22].

2.3.1 Random Time-Varying Communication Model

We assume agents communicate according to a time-varying communication graph, obtained from an *underlying* graph $\mathcal{G}_u = (V, \mathcal{E}_u)$ assumed to be undirected and connected, where $V = \{1, \ldots, N\}$ is the set of nodes and $\mathcal{E}_u \subseteq V \times V$ is the set of edges. An edge (i, j) belongs to $\mathcal{E}_u$ if and only if agents i and j can transmit information to each other, in which case also $(j, i) \in \mathcal{E}_u$. In many applications, the communication links are not always active (due e.g. to temporary unavailability). This is taken into account by considering that each undirected edge $(i, j) \in \mathcal{E}_u$ has a probability $\sigma_{ij} \in (0, 1]$ of being active. As a result, the actual communication network is a random time-varying graph $\mathcal{G}^t = (V, \mathcal{E}^t)$, where $t \in \mathbb{N}$ represents the time index and $\mathcal{E}^t \subseteq \mathcal{E}_u$ is the set of active edges at time t. The set of neighbors of agent i in $\mathcal{G}^t$ is denoted by $\mathcal{N}_i^t = \{j \in V \mid (i, j) \in \mathcal{E}^t\}$. Consistently, the set of neighbors of agent i in the underlying graph $\mathcal{G}_u$ is denoted by $\mathcal{N}_{i,u}$. A pictorial representation of the time-varying communication model is provided in Figure 2.4.

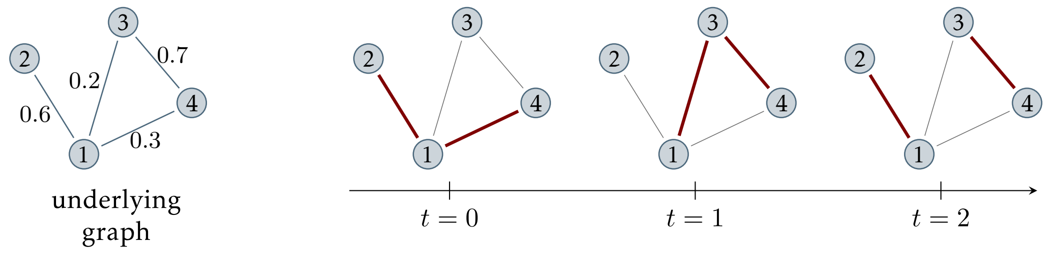


Figure 2.4: Example of random time-varying graph with $N = 4$ agents. Active edges are denoted with red lines, while inactive edges are depicted with thin gray lines. The (connected) underlying graph is shown on the left with the edge activation probabilities.

Let us define ν_{ij}^t as the Bernoulli random variable that is equal to 1 if $(i,j) \in \mathcal{E}^t$ and 0 otherwise, for all $(i,j) \in \mathcal{E}_u$ with $j > i$ and $t \geq 0$. The following assumption is made.

Assumption 2.4. *For all $(i,j) \in \mathcal{E}_u$ with $j > i$, the random variables $\{\nu_{ij}^t\}_{t\geq 0}$ are independent and identically distributed (i.i.d.). Moreover, for all $t \geq 0$, the random variables $\{\nu_{ij}^t\}_{(i,j)\in\mathcal{E}_u,\, j>i}$ are mutually independent.* $\triangle$

2.3.2 Distributed Algorithm Description

Let us now introduce the Distributed Primal Decomposition algorithm for time-varying graphs to solve Problem (2.1). Let $t \in \mathbb{N}$ be the iteration index, let $\alpha^t \geq 0$ denote the step size and let $M > 0$ be the tuning parameter of the relaxation approach (see also Section 2.3.4). Each agent i maintains an estimate of the local allocation vector $y_i^t \in \mathbb{R}^S$, initialized such that $\sum_{i=1}^N y_i^0 = b$ (e.g. $y_i^0 = b/N$ for all i). At each iteration $t \in \mathbb{N}$, agent i computes (x_i^t, ρ_i^t) as an optimal solution to a local optimization problem (cf. (2.17)) and μ_i^t as a Lagrange multiplier of the same problem. Then, the agent exchanges μ_i^t with the neighbors on the currently active communication links and updates y_i^t (cf. (2.18)). Algorithm 1 summarizes the distributed algorithm from the perspective of node i, where we recall that the notation "μ_i :" in (2.17) means that μ_i is the Lagrange multiplier associated to the constraint $g_i(x_i) \leq y_i^t + \rho_i \mathbf{1}$.

Algorithm 1 Distributed Primal Decomposition (Time-varying graphs)

```
**Initialization**: $y_i^0$ such that $\sum_{i=1}^N y_i^0 = b$
**For** $t = 0, 1, 2, \dots$
    **Compute** $((x_i^t, \rho_i^t),\ \mu_i^t)$ as a primal-dual solution of
        $$\begin{aligned} \min_{x_i,\rho_i} \quad & f_i(x_i) + M\rho_i \\ \text{subj. to} \quad \mu_i :\ & g_i(x_i) \leq y_i^t + \rho_i \mathbf{1} \\ & x_i \in X_i,\ \ \rho_i \geq 0 \end{aligned} \tag{2.17}$$
    **Receive** $\mu_j^t$ from neighbors $j \in \mathcal{N}_i^t$ and update
        $$y_i^{t+1} = y_i^t + \alpha^t \sum_{j\in\mathcal{N}_i^t} \big(\mu_i^t - \mu_j^t\big) \tag{2.18}$$
```

Some appealing features of Algorithm 1 are worth highlighting. The algorithm naturally preserves privacy of all the agents, in the sense they do not communicate any of their private information (such as the local cost function f_i, the local constraint set X_i or the local solution estimate x_i^t). Moreover, the algorithm has attractive scaling properties, indeed the amount of local computation stays constant as the size of the network is increased.

Remark 2.1. The Distributed Primal Decomposition algorithm can be formulated in aggregate form through the use of the graph Laplacian matrix (see Section 1.1). For a single coupling constraint ($S = 1$), the update (2.18) reads

$$y^{t+1} = y^t + \alpha^t L^t \mu^t = y^t - \alpha^t L^t \widetilde{\nabla} p(y^t),$$

where $L^t \in \mathbb{R}^{N \times N}$ is the time-varying Laplacian matrix of $\mathcal{G}^t$ and we recall that $\mu^t \in \mathbb{R}^N$ denotes the vector stacking all μ_i^t, while $\widetilde{\nabla} p(y^t)$ denotes a subgradient of $p(y) = \sum_{i=1}^N p_i(y_i)$ at y^t. Similarly, the parallel update (2.11) reads

$$\bar{y}^{t+1} = \bar{y}^t - \alpha^t L_{\text{co}} \widetilde{\nabla} p(\bar{y}^t).$$

where the constant $1/N$ is absorbed in the step size and L_{co} is the laplacian matrix of the complete graph. △

Next we provide the convergence properties of the Distributed Primal Decomposition algorithm for time-varying graphs. It should be noted that the only change with respect to the algorithm for static graphs in Section 2.2.5 is that the local allocation vector is updated by using only the currently active communication links. However, despite this simple change, the analysis of Algorithm 1 is quite complex and requires several technical tools that will be provided in the forthcoming subsections.

Theorem 2.1. *Let Problem* (2.1) *be feasible and let Assumptions 2.1, 2.2, 2.3 and 2.4 hold. Moreover, let $\mu^\star$ be an optimal Lagrange multiplier of Problem* (2.1) *associated to the constraint $\sum_{i=1}^N g_i(x_i) \leq b$ and assume $M > \|\mu^\star\|_1$. Consider a sequence $\{x_i^t, \rho_i^t\}_{t \geq 0}$, $i = 1, \ldots, N$, generated by Algorithm 1 with allocation vectors y_i^0 initialized such that $\sum_{i=1}^N y_i^0 = b$. Then, almost surely,*

(i) $\sum_{i=1}^N \big(f_i(x_i^t) + M \rho_i^t \big) \to f^\star$ *as $t \to \infty$, where $f^\star$ is the optimal cost of* (2.1)*;*

(ii) every limit point of $\{(x_1^t, \ldots x_N^t)\}_{t \geq 0}$ *is an optimal solution of* (2.1). △

By Theorem 2.1 (i) and the fact that $\rho_i^t \geq 0$ (by construction), it follows that $\{\rho_i^t\}_{t \geq 0}$ vanishes for all $i \in \{1, \ldots, N\}$ and thus $\sum_{i=1}^N f_i(x_i^t) \to f^\star$ as $t \to \infty$. In principle, in order to satisfy the assumption $M > \|\mu^\star\|_1$ in Theorem 2.1, knowledge is needed of the dual optimal solution $\mu^\star$. However, this is not necessary in practice, as a lower bound for M can be efficiently computed when a Slater point is known. In Section 2.3.4, we provide a sufficient condition to select valid values of M without any knowledge on $\mu^\star$.

Note also that the algorithm does not employ any averaging mechanism typically appearing in dual algorithms when the cost functions are not strictly convex. Thanks to the primal decomposition approach, we are still able to prove asymptotic feasibility and optimality of the sequence $\{(x_1^t, \ldots x_N^t)\}_{t \geq 0}$. As shown in Section 2.5.3, the absence of running averages allows for faster practical convergence, compared to existing methods.

Finally, let us highlight that, in the typical case in which $\|\mu^\star\|_1$ is not known a priori, the agents can easily obtain a conservative bound for the parameter M in a completely distributed way. As described in Section 2.3.4, it is sufficient to run a combination of $\max$-consensus and average consensus protocols to make the agents agree upon a common value of M before starting the execution of the distributed algorithm.

2.3.3 Handling Equality Coupling Constraints

The Distributed Primal Decomposition algorithm is also able to handle additional equality coupling constraints with minor modifications to the overall algorithmic structure. To see this, let us consider the constraint-coupled problem

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^N f_i(x_i) \\ \text{subj. to} \quad & \sum_{i=1}^N g_i(x_i) \le b, \quad \sum_{i=1}^N h_i(x_i) = \kappa, \\ & x_i \in X_i, \qquad i = 1,\ldots,N, \end{aligned} \tag{2.19}$$

which, compared to Problem (2.1), includes $Q \in \mathbb{N}$ additional equality coupling constraints with each $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^Q$ and $\kappa \in \mathbb{R}^Q$. We maintain all the previous assumptions, including convexity of the problem, therefore we must assume that each h_i is an affine function. These new constraints ought to be managed by adapting the relaxation approach of Section 2.2.3. Formally, let us define the exact penalty function

$$\begin{aligned} P(x) \triangleq \max\Bigg\{ 0, \sum_{i=1}^N g_{i,1}(x_i) - b_1, \ldots, \sum_{i=1}^N g_{i,S}(x_i) - b_S, \\ \Bigg|\sum_{i=1}^N h_{i,1}(x_i) - \kappa_1\Bigg|, \ldots, \Bigg|\sum_{i=1}^N h_{i,Q}(x_i) - \kappa_Q\Bigg|\Bigg\} \end{aligned}$$

and let us consider the problem

$$\min_{x_1 \in X_1,\ldots,x_N \in X_N} \sum_{i=1}^N f_i(x_i) + MP(x). \tag{2.20}$$

By standard results (see e.g. [10]), if we denote by $\mu^\star$ and $\lambda^\star$ the Lagrange multiplier associated to the inequality and the equality constraints of Problem (2.19), then Problems (2.19) and (2.20) have the same optimal solutions, provided that $M > \|\mu^\star\|_1 + \|\lambda^\star\|_1$. Upon rewriting Problem (2.20) in its epigraph form and applying the primal decomposition approach, we obtain a master problem identical to (2.7) (with b replaced by the column vector (b, κ)) and the following form of the subproblems,

$$\begin{aligned} \min_{x_i, \rho_i} \quad & f_i(x_i) + M\rho_i \\ \text{subj. to} \quad \mu_i : \; & g_i(x_i) \leq y_i^{\text{IN}} + \rho_i \mathbf{1} \\ \lambda_i : \; & \left| h_i(x_i) - y_i^{\text{EQ}} \right| \leq \rho_i \mathbf{1} \\ & x_i \in X_i, \;\; \rho_i \geq 0, \end{aligned} \tag{2.21}$$

where here we defined $y_i = (y_i^{\text{IN}}, y_i^{\text{EQ}})$ and $y_i^{\text{IN}} \in \mathbb{R}^S$, $y_i^{\text{EQ}} \in \mathbb{R}^Q$. The distributed algorithm maintains the form of Algorithm 1, except that the initialization must satisfy $\sum_{i=1}^N y_i^0 = (b, \kappa)$, Problem (2.17) is replaced by Problem (2.21) at $y_i = (y_i^{\text{IN},t}, y_i^{\text{EQ},t})$, and the update (2.18) is replaced by

$$y_i^{\text{IN},t+1} = y_i^{\text{IN},t} + \alpha^t \sum_{j \in \mathcal{N}_i^t} \big(\mu_i^t - \mu_j^t \big), \tag{2.22a}$$

$$y_i^{\text{EQ},t+1} = y_i^{\text{EQ},t} + \alpha^t \sum_{j \in \mathcal{N}_i^t} \big(\lambda_i^t - \lambda_j^t \big), \tag{2.22b}$$

with μ_i^t and λ_i^t being the Lagrange multipliers associated to the inequality constraints of Problem (2.21). The convergence properties of this modified version of the algorithm will be discussed in Remark 2.2, after the proof of Theorem 2.1.

2.3.4 Discussion on the Parameters

Let us discuss the choice of the parameter M in the local optimization problem (2.17) appearing in Algorithm 1.

As per Theorem 2.1, in order to have convergence to an optimum it must hold $M > \|\mu^\star\|_1$, where $\mu^\star$ is any dual optimal solution of the original problem (2.1). This assumption is needed for the relaxation approach to apply (cf. Section 2.2.3). In general, a dual optimal solution $\mu^\star$ of the original problem (2.1) may not be known in advance. However, if a Slater point is known (cf. Assumption 2.2), it is possible for the agents to compute a conservative lower bound on M. The next proposition provides a sufficient condition to satisfy $M > \|\mu^\star\|_1$.

Proposition 2.2. *Let Assumptions 2.1 and 2.2 hold. Moreover, let $(\bar{x}_1, \ldots, \bar{x}_N)$ be a Slater point, i.e. a feasible point for Problem* (2.1) *with $\sum_{i=1}^N g_i(\bar{x}_i) < b$. Then, a valid choice of M for Theorem 2.1 is any number satisfying*

$$M > \frac{1}{\gamma} \sum_{i=1}^N \Big(f_i(\bar{x}_i) - \min_{x_i \in X_i} f_i(x_i) \Big), \tag{2.23}$$

where $\gamma = \min_{1 \leq s \leq S} \{ b_s - \sum_{i=1}^N g_{is}(\bar{x}_i) \}$.

Proof. Let us consider the dual problem associated to (2.1) when only the constraint $\sum_{i=1}^{N} g_i(x_i) \leq b$ is dualized,

$$\begin{aligned} &\max_{\mu \in \mathbb{R}^S} \; q(\mu) \\ &\text{subj. to } \; \mu \geq 0, \end{aligned} \tag{2.24}$$

with $q(\mu)$ being the dual function, defined as

$$\begin{aligned} q(\mu) &= \inf_{x_1 \in X_1, \ldots, x_N \in X_N} \left\{ \sum_{i=1}^{N} \big(f_i(x_i) + \mu^\top g_i(x_i) \big) - \mu^\top b \right\} \\ &= -\mu^\top b + \sum_{i=1}^{N} \inf_{x_i \in X_i} \big(f_i(x_i) + \mu^\top g_i(x_i) \big), \\ &= -\mu^\top b + \sum_{i=1}^{N} \min_{x_i \in X_i} \big(f_i(x_i) + \mu^\top g_i(x_i) \big), \end{aligned}$$

where $\mu^\top b$ can be brought out of the $\inf$ since is constant for $x_1, \ldots, x_N$, the $\inf$ can be split because the summands depend on different variables and the operator $\inf$ can be replaced by $\min$ since the sets X_i are compact and f_i, g_i are continuous due to convexity (cf. Assumption 2.1). Let us denote by $\mu^\star$ an optimal solution of Problem (2.24). By Assumptions 2.1 and 2.2, strong duality holds, therefore $q(\mu^\star) = \sum_{i=1}^{N} f_i(x_i^\star)$, where $(x_1^\star, \ldots, x_N^\star)$ is an optimal solution of Problem (2.1). Also, note that $\mu^\star$ is also a Lagrange multiplier of Problem (2.1) (see [10, Proposition 5.1.4]). To upper bound $\|\mu^\star\|_1$, we invoke [80, Lemma 1],

$$\begin{aligned} \|\mu^\star\|_1 &\leq \frac{1}{\gamma} \left(\sum_{i=1}^{N} f_i(\bar{x}_i) - q(\mu^\star) \right) \\ &= \frac{1}{\gamma} \sum_{i=1}^{N} \big(f_i(\bar{x}_i) - f_i(x_i^\star) \big) \\ &\leq \frac{1}{\gamma} \sum_{i=1}^{N} \Big(f_i(\bar{x}_i) - \min_{x_i \in X_i} f_i(x_i) \Big), \end{aligned} \tag{2.25}$$

where the minimum in the right-hand side of (2.25) exists since X_i is compact. The proof follows by choosing M as any number strictly greater than the right-hand side of (2.25). ∎

From an operative point of view, if each agent knows its block $\bar{x}_i$ of a Slater vector $(\bar{x}_1, \ldots, \bar{x}_N)$, the network can run a combination of $\min$-consensus and average consensus protocols to determine the right-hand side of (2.23), because the quantities in the sum are locally computable. As such, the calculation of M can be completely distributed and can be performed as a preliminary step before starting the execution of Algorithm 1.

2.4 Convergence analysis and convergence rates

In this section, we provide the convergence analysis of Algorithm 1 and derive convergence rates under two different assumptions on the step-size.

The analysis is based on the primal decomposition scheme applied to the relaxed problem (cf. Section 2.2.3). First, we reformulate the master problem (2.7) by exploiting the graph structure. This reformulation is then used to show that our distributed algorithm is equivalent to a (centralized) randomized block subgradient method, whose analysis is a building block to prove Theorem 2.1.

Let us introduce some notation needed for the analysis. The $n \times n$ identity matrix is denoted by I_n. Where the size of the matrix is clear from the context, we drop the subscript n. Given a vector $x \in \mathbb{R}^n$ and a positive definite matrix $W \in \mathbb{R}^{n \times n}$, we denote by $\|x\|_W = \sqrt{x^\top W x}$ the norm of x weighted by W, which we term W-norm. The symbol $\otimes$ denotes the Kronecker product. Given a block vector $z = (z_1, \ldots, z_m)$, we denote its ℓ-th block by $[z]_\ell$ when the notation z_ℓ can be ambiguous. We recall that $\mathbf{1}$ denotes the vector of ones of appropriate dimensions. Similarly, we denote by $\mathbf{0}$ the vector of zeros.

2.4.1 Encoding the Coupling Constraints in Cost Function

As already mentioned in Section 2.2, a solution of Problem (2.1) can be indirectly obtained by solving Problem (2.7). In order to put Problem (2.7) into a form that is more convenient for distributed computation, let us now apply a graph-induced change of coordinates. Such a manipulation has a twofold benefit: *(i)* it allows for the suppression and implicit satisfaction of the constraint $\sum_{i=1}^{N} y_i = b$, *(ii)* it allows for the application of a randomized block subgradient method to take into account the random activation of edges.

Consider the underlying communication graph $\mathcal{G}_u$. Assuming an arbitrary ordering of the edges, let $\Gamma \in \mathbb{R}^{|\mathcal{E}_u| \times N}$ denote the incidence matrix of $\mathcal{G}_u$, where each row (corresponding to an edge of $\mathcal{G}_u$) contains all zero entries except for the column corresponding to the edge tail (equal to 1), and for the column corresponding to the edge head (equal to -1). Namely, if the k-th row of Γ corresponds to the edge (i, j), then the (k, ℓ)-th entry of Γ is

$$(\Gamma)_{k\ell} = \begin{cases} 1 & \text{if } \ell = i, \\ -1 & \text{if } \ell = j, \\ 0 & \text{otherwise,} \end{cases}$$

for all $\ell \in \{1, \ldots, N\}$. For all $(i, j) \in \mathcal{E}_u$, let $z_{ij} \in \mathbb{R}^S$ be a vector associated to the edge

(i,j) and denote by $z \in \mathbb{R}^{S|\mathcal{E}_u|}$ the vector stacking all z_{ij}, with the same ordering as in Γ. Consider the change of coordinates for Problem (2.7) defined through the following linear mapping

$$y = \Pi z + \frac{1}{N}(\mathbf{1} \otimes b), \tag{2.26}$$

where $\mathbf{1} \in \mathbb{R}^S$ and the matrix Π is defined as

$$\Pi \triangleq (\Gamma^\top \otimes I_S) \in \mathbb{R}^{SN \times S|\mathcal{E}_u|}. \tag{2.27}$$

By using the properties of the Kronecker product, the blocks of y can be written as

$$y_i = [\Pi z]_i + \frac{b}{N} = \sum_{j \in \mathcal{N}_{i,u}} (z_{ij} - z_{ji}) + \frac{b}{N}, \quad \forall\, i \in \{1,\ldots,N\}.$$

The next lemma formalizes the fact that the change of variable (2.26) implicitly encodes the constraint $\sum_{i=1}^N y_i = b$.

Lemma 2.5. *The matrix Π in* (2.27) *satisfies:*

(i) $\sum_{i=1}^N [\Pi z]_i = \mathbf{0}$ *for all* $z \in \mathbb{R}^{S|\mathcal{E}_u|}$*;*

(ii) for all $\tilde{y} \in \mathbb{R}^{SN}$ *satisfying* $\sum_{i=1}^N \tilde{y}_i = b$ *there exists* $\tilde{z} \in \mathbb{R}^{S|\mathcal{E}_u|}$ *such that* $\tilde{y} = \Pi\tilde{z} + \frac{1}{N}(\mathbf{1} \otimes b)$.

Proof. To show *(i)*, we see that

$$\begin{aligned}
\sum_{i=1}^N [\Pi z]_i &= (\mathbf{1}^\top \otimes I_S)\Pi z \\
&= (\mathbf{1}^\top \otimes I_S)(\Gamma^\top \otimes I_S) z \\
&= \big((\Gamma \otimes I_S)(\mathbf{1} \otimes I_S)\big)^\top z \\
&\overset{(a)}{=} \big((\Gamma\mathbf{1}) \otimes I_S\big)^\top z \\
&\overset{(b)}{=} \big(\mathbf{0} \otimes I_S\big)^\top z = \mathbf{0},
\end{aligned}$$

where in (a) we used the fact $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ since the matrix dimensions are compatible, and (b) follows by the property $\Gamma\mathbf{1} = \mathbf{0}$ of incidence matrices.

To prove *(ii)*, let $\tilde{y} \in \mathbb{R}^{SN}$ be such that $\sum_{i=1}^N \tilde{y}_i = b$, or, equivalently, $(\mathbf{1}^\top \otimes I_S)\tilde{y} = b$. Let us first show that $v^\top(\tilde{y} - \frac{1}{N}(\mathbf{1} \otimes b)) = 0$ for all $v \in \mathrm{Ker}(\Pi^\top)$. To this end, take $v \in \mathrm{Ker}(\Pi^\top)$. Since $\mathcal{G}_u$ is connected, then $\mathrm{rank}(\Gamma) = N - 1$. Thus, by the properties of the Kronecker product, it holds

$$\mathrm{rank}(\Pi^\top) = \mathrm{rank}(\Gamma \otimes I_S)$$

$$
\begin{aligned}
&= \operatorname{rank}(\Gamma)\operatorname{rank}(I_S)\\
&= (N-1)S.
\end{aligned}
$$

Moreover, by the Rank-Nullity Theorem, it holds

$$
\dim \operatorname{Ker}(\Pi^\top) = SN - \operatorname{rank}(\Pi^\top) = S.
$$

But since the columns of $(\mathbf{1} \otimes I_S) \in \mathbb{R}^{SN \times S}$ are linearly independent, and since the point *(i)* of the lemma implies that they belong to $\operatorname{Ker}(\Pi^\top)$, it follows that they are actually a basis of $\operatorname{Ker}(\Pi^\top)$, so that the vector v can be written as $v = (\mathbf{1} \otimes I_S)\lambda$, for some $\lambda \in \mathbb{R}^S$. Therefore, it holds

$$
\begin{aligned}
v^\top \Big(\tilde{y} - \frac{1}{N}(\mathbf{1} \otimes b)\Big) &= \lambda^\top (\mathbf{1}^\top \otimes I_S)\Big(\tilde{y} - \frac{1}{N}(\mathbf{1} \otimes b)\Big)\\
&= \lambda^\top \underbrace{(\mathbf{1}^\top \otimes I_S)\tilde{y}}_{b} - \frac{1}{N}\lambda^\top \underbrace{(\mathbf{1}^\top \otimes I_S)(\mathbf{1} \otimes b)}_{Nb} = 0.
\end{aligned}
$$

Thus, since v is arbitrary, it follows that $v^\top(\tilde{y} - \frac{1}{N}(\mathbf{1} \otimes b)) = 0$ for all $v \in \operatorname{Ker}(\Pi^\top)$. By definition of orthogonal complement, this means that $\tilde{y} - \frac{1}{N}(\mathbf{1} \otimes b) \in \operatorname{Ker}(\Pi^\top)^\perp = \operatorname{Im}(\Pi)$. Equivalently, there exists $\tilde{z}$ such that $\tilde{y} = \Pi\tilde{z} + \frac{1}{N}(\mathbf{1} \otimes b)$. ■

We now plug the change of coordinates (2.26) into Problem (2.7). Formally, for all $i \in \{1, \dots, N\}$, define the functions

$$
\tilde{p}_i\big(\{z_{ij}, z_{ji}\}_{j \in \mathcal{N}_{i,u}}\big) \triangleq p_i\bigg([\Pi z]_i + \frac{b}{N}\bigg), \qquad z \in \mathbb{R}^{S|\mathcal{E}_u|}.
$$

By Lemma 2.5, we directly obtain the following result.

Corollary 2.1. *Problem* (2.7) *is equivalent to the unconstrained optimization problem*

$$
\min_{z \in \mathbb{R}^{S|\mathcal{E}_u|}} \sum_{i=1}^{N} \tilde{p}_i\big(\{z_{ij}, z_{ji}\}_{j \in \mathcal{N}_{i,u}}\big), \tag{2.28}
$$

in the sense that (i) *the optimal costs are equal,* (ii) *if $z^\star$ is an optimal solution of* (2.28), *then $y^\star = \Pi z^\star + \frac{1}{N}(\mathbf{1} \otimes b)$ is an optimal solution of* (2.7). △

In the following, we denote the cost function of (2.28) as $\tilde{p}(z) = p\big(\Pi z + \frac{1}{N}(\mathbf{1} \otimes b)\big)$.

2.4.2 Randomized Block Subgradient Method

The analysis of the distributed algorithm requires results from the (centralized) block subgradient literature. To the best of our knowledge, no reference provides the needed results with the desired degree of refinement (i.e. almost sure cost convergence with multiple block updates and non-uniform block probabilities). Thus, in this subsection, we formulate and analyze a (centralized) randomized block subgradient method for convex problems, which is a side contribution of this chapter.

Let us consider the unconstrained convex problem

$$\min_{\theta \in \mathbb{R}^m} \ \varphi(\theta), \tag{2.29}$$

where θ is the optimization variable and $\varphi : \mathbb{R}^m \to \mathbb{R}$ is a convex function. We assume that Problem (2.29) has finite optimal cost, denoted by $\varphi^\star$, and that at least an optimal solution $\theta^\star \in \mathbb{R}^m$ exists, so that $\varphi^\star = \varphi(\theta^\star)$.

Let us consider a partition of $\mathbb{R}^m$ into $B \in \mathbb{N}$ parts, i.e. $\mathbb{R}^m = \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_B}$, such that $m = \sum_{\ell=1}^{B} m_\ell$. Therefore, the optimization variable is the stack of B blocks,

$$\theta = (\theta_1, \ldots, \theta_B),$$

where $\theta_\ell \in \mathbb{R}^{m_\ell}$ for all $\ell \in \{1, \ldots, B\}$. Now, we develop a subgradient method with block-wise updates to solve Problem (2.29). At each iteration $t \in \mathbb{N}$, each block ℓ is updated with a probability $\sigma_\ell > 0$. We stress that according to the considered model, blocks can have different update probabilities and multiple blocks can be updated simultaneously.

For all t, we denote by $B^t \subseteq \{1, \ldots, B\}$ the index set of the blocks selected at time t. For all $\ell \in \{1, \ldots, B\}$ and $t \geq 0$, let us define ν_ℓ^t as the Bernoulli random variable that is equal to 1 if $\ell \in B^t$ and 0 otherwise. The following assumption is made (compare with Assumption 2.4).

Assumption 2.5. *For all $\ell \in \{1, \ldots, B\}$, the random variables $\{\nu_\ell^t\}_{t \geq 0}$ are independent and identically distributed (i.i.d.). Moreover, for all $t \geq 0$, the random variables $\{\nu_\ell^t\}_{\ell \in \{1,\ldots,B\}}$ are mutually independent.* △

The algorithm considered here is based on a subgradient method. However, at each iteration t, only the blocks in B^t are updated, i.e.

$$\theta_\ell^{t+1} = \begin{cases} \theta_\ell^t - \alpha^t [\widetilde{\nabla} \varphi(\theta^t)]_\ell, & \text{if } \ell \in B^t, \\ \theta_\ell^t, & \text{if } \ell \notin B^t, \end{cases} \tag{2.30}$$

where α^t is the step size. We stress again that algorithm (2.30) allows for multiple block updates at once and, furthermore, blocks have non-uniform update probabilities. We now provide the convergence proof for algorithm (2.30). To the best of our knowledge, for this general block-subgradient method no almost sure cost convergence results have been proven in the literature.

Theorem 2.2. *Let Assumption 2.5 hold and let the step-size sequence $\{\alpha^t\}_{t \geq 0}$ satisfy As-*

sumption 2.3. Moreover, assume the subgradients of φ are block-wise bounded, i.e. assume for all $\ell \in \{1, \dots, B\}$ there exists $C_\ell > 0$ such that $\|[\widetilde{\nabla}\varphi(\theta)]_\ell\| \leq C_\ell$ for all $\theta \in \mathbb{R}^m$. Consider a sequence $\{\theta^t\}_{t \geq 0}$ generated by algorithm (2.30), initialized at any $\theta^0 \in \mathbb{R}^m$. Then, almost surely, it holds

$$\lim_{t \to \infty} \varphi(\theta^t) = \varphi^\star.$$

Proof. To keep the notation light, let us denote the computed subgradients as $\beta^t \triangleq \widetilde{\nabla}\varphi(\theta^t)$. Each block ℓ is denoted by $\beta^t_\ell = [\widetilde{\nabla}\varphi(\theta^t)]_\ell$. Moreover, for all $\ell \in \{1, \dots, B\}$, let us define the matrix $U_\ell \in \mathbb{R}^{m \times m}$, obtained by setting to zero in the identity matrix all the blocks on the diagonal, except for the ℓ-th block. Thus, when applied to a vector $\theta \in \mathbb{R}^m$, all the blocks other than the ℓ-th one are set to zero, i.e.

$$[U_\ell \theta]_\kappa = \begin{cases} \theta_\ell & \text{if } \kappa = \ell, \\ \mathbf{0} & \text{otherwise}, \end{cases} \qquad \forall\, \kappa \in \{1, \dots, B\}.$$

Moreover, for the sake of analysis, let us define

$$W \triangleq \operatorname{diag}\Big(\frac{1}{\sigma_1} I_{m_1}, \; \dots, \; \frac{1}{\sigma_B} I_{m_B}\Big),$$

where $\operatorname{diag}(\cdot)$ is the (block) diagonal operator. Note that W is positive definite, thus we can consider the weighted norm $\|\theta\|_W$ for which by definition it holds

$$\|\theta\|_W^2 = \sum_{\ell=1}^{B} \frac{\|\theta_\ell\|^2}{\sigma_\ell}, \qquad \theta \in \mathbb{R}^m.$$

Next we analyze algorithm (2.30). Let us focus on an iteration t and consider any vector $\theta \in \mathbb{R}^m$. As for the activated blocks $\ell \in B^t$, it holds

$$\begin{aligned} \|\theta_\ell^{t+1} - \theta_\ell\|^2 &= \|\theta_\ell^t - \alpha^t \beta_\ell^t - \theta_\ell\|^2 \\ &= \|\theta_\ell^t - \theta_\ell\|^2 + (\alpha^t)^2 \|\beta_\ell^t\|^2 - 2\alpha^t (\beta_\ell^t)^\top \big(\theta_\ell^t - \theta_\ell\big), \\ &\leq \|\theta_\ell^t - \theta_\ell\|^2 + (\alpha^t)^2 C_\ell^2 - 2\alpha^t U_\ell (\beta^t)^\top \big(\theta^t - \theta\big), \qquad \forall\, \ell \in B^t, \end{aligned}$$

where $\|\beta_\ell^t\| \leq C_\ell$ holds by assumption. As for the other blocks $\ell \notin B^t$, we have

$$\|\theta_\ell^{t+1} - \theta_\ell\|^2 = \|\theta_\ell^t - \theta_\ell\|^2, \qquad \forall\, \ell \notin B^t.$$

Let us now write the overall evolution in W-norm,

$$\begin{aligned}\|\theta^{t+1} - \theta\|_W^2 &= \sum_{\ell \in B^t} \frac{\|\theta_\ell^{t+1} - \theta_\ell\|^2}{\sigma_\ell} + \sum_{\ell \notin B^t} \frac{\|\theta_\ell^{t+1} - \theta_\ell\|^2}{\sigma_\ell} \\ &\le \sum_{\ell=1}^{B} \frac{\|\theta_\ell^{t} - \theta_\ell\|^2}{\sigma_\ell} + (\alpha^t)^2 \sum_{\ell \in B^t} \frac{C_\ell^2}{\sigma_\ell} - 2\alpha^t \bigg(\sum_{\ell \in B^t} \frac{1}{\sigma_\ell} U_\ell \bigg) (\beta^t)^\top \big(\theta^t - \theta\big) \\ &\le \|\theta^t - \theta\|_W^2 + (\alpha^t)^2 C - 2\alpha^t \bigg(\sum_{\ell \in B^t} \frac{1}{\sigma_\ell} U_\ell \bigg) (\beta^t)^\top \big(\theta^t - \theta\big), \end{aligned} \tag{2.31}$$

where $C \triangleq \sum_{\ell=1}^{B} \frac{C_\ell^2}{\sigma_\ell} > 0$. Let us compute the expected value of the matrix $\sum_{\ell \in B^t} \frac{1}{\sigma_\ell} U_\ell$ appearing in (2.31),

$$\mathbb{E}\Bigg[\sum_{\ell \in B^t} \frac{1}{\sigma_\ell} U_\ell \Bigg] = \mathbb{E}\Bigg[\sum_{\ell=1}^{B} \frac{\nu_\ell^t}{\sigma_\ell} U_\ell \Bigg] = \sum_{\ell=1}^{B} U_\ell = I_m. \tag{2.32}$$

Now, by taking the conditional expectation of (2.31) with respect to $\mathcal{F}^t = \{\theta^0, \ldots, \theta^t\}$ (namely the sequence generated by algorithm (2.30) up to iteration t), we obtain for all $\theta \in \mathbb{R}^m$ and $t \ge 0$

$$\begin{aligned}\mathbb{E}\Big[\|\theta^{t+1} - \theta\|_W^2 \;\Big|\; \mathcal{F}^t \Big] &\overset{(a)}{\le} \|\theta^t - \theta\|_W^2 + (\alpha^t)^2 C - 2\alpha^t (\beta^t)^\top \big(\theta^t - \theta\big), \\ &\overset{(b)}{\le} \|\theta^t - \theta\|_W^2 + (\alpha^t)^2 C - 2\alpha^t \big(\varphi(\theta^t) - \varphi(\theta)\big), \end{aligned}$$

where in (a) we used (2.32) and the independence of the drawn blocks from the previous iterations (cf. Assumption 2.5), and (b) follows by definition of subgradient of the function φ. By restricting the above inequality to any optimal solution $\theta^\star$ of Problem (2.28), we obtain

$$\mathbb{E}\Big[\|\theta^{t+1} - \theta^\star\|_W^2 \;\Big|\; \mathcal{F}^t \Big] \le \|\theta^t - \theta^\star\|_W^2 + (\alpha^t)^2 C - 2\alpha^t \big(\varphi(\theta^t) - \varphi^\star\big). \tag{2.33}$$

Inequality (2.33) satisfies the assumptions of [12, Proposition 8.2.10]. By following the same arguments as in [12, Proposition 8.2.13], we conclude that, almost surely, it holds

$$\lim_{t \to \infty} \varphi(\theta^t) = \varphi^\star. \qquad \blacksquare$$

In the preceding proof, the subgradient boundedness assumption is required to derive the inequality (2.33) and thus to apply supermartingale convergence arguments. In the forthcoming discussion, we will show that this assumption is satisfied due to the relaxation approach, however in principle Theorem 2.2 may be strengthened by considering a more relaxed assumption in place of subgradient boundedness (see, e.g., [11, Eq. (3.19)]).

2.4.3 Equivalence of Algorithm 1 and Randomized Block Subgradient

Let us now continue with the analysis of Algorithm 1. Differently from Problem (2.7), its equivalent formulation (2.28) is unconstrained. Hence, it can be solved via subgradient methods without projections steps. It is possible to exploit the particular structure of Problem (2.28) to recast the random activation of edges as the random update of blocks within a block subgradient method (2.30) applied to Problem (2.28). We will use the following identifications,

$$\theta = z, \qquad \text{and} \qquad \varphi(\theta) = \sum_{i=1}^{N} \tilde{p}_i\big(\{z_{(ij)}, z_{(ji)}\}_{j\in\mathcal{N}_{i,u}}\big). \tag{2.34}$$

As for the block structure, the mapping is as follows. Each block $\ell \in \{1,\dots,B\}$ of z, i.e. $z_\ell \in \mathbb{R}^{2S}$, is associated to an undirected edge $(i,j) \in \mathcal{E}_u$, with $j > i$, and is defined as

$$z_\ell = \begin{bmatrix} z_{(ij)} \\ z_{(ji)} \end{bmatrix}. \tag{2.35}$$

Therefore, there is a total of $B = |\mathcal{E}_u|/2$ blocks. At each iteration t, each block z_ℓ is updated if the corresponding edge $(i,j) \in \mathcal{E}^t$, i.e. if $\nu_{ij}^t = 1$. A pictorial representation of the block structure of z is provided in Figure 2.5. Consistently with the notation

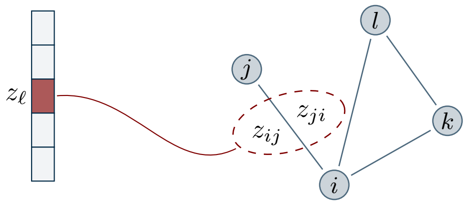


Figure 2.5: Block structure of the variable z. Each block, say ℓ, is associated to an undirected edge, say (i,j). The block is the stack of z_{ij}, associated to the edge (i,j), and z_{ji} associated to the edge (j,i).

of Section 2.4.2, we use the shorthands $\sigma_\ell = \sigma_{ij}$ and $\nu_\ell^t = \nu_{ij}^t$. At each iteration t of algorithm (2.30), the set B^t contains all and only the blocks associated to the edges in $\mathcal{E}^t$.

Next, we explicitly write the evolution of the sequences generated by Algorithm 1 as a function of the sequences generated by the block subgradient method (2.30). For this purpose, let us write a subgradient of $\tilde{p}$ at any $z \in \mathbb{R}^{S|\mathcal{E}_u|}$. By definition, it holds $\tilde{p}(z) = p\big(\Pi z + \frac{1}{N}(\mathbf{1}\otimes b)\big)$. Thus, by using the subgradient property for affine transformations of the domain[3], it holds

$$\widetilde{\nabla}\tilde{p}(z) = (\Gamma \otimes I_S)\widetilde{\nabla}p\Big(\Pi z + \frac{1}{N}(\mathbf{1}\otimes b)\Big). \tag{2.36}$$

By exploiting the structure of p, the i-th block of $\widetilde{\nabla}p(y)$ is equal to $\frac{\tilde{\partial}p(y)}{\partial y_i} = \widetilde{\nabla}p_i(y_i)$. Moreover, since Problem (2.8) enjoys strong duality, a subgradient of p_i at y_i can be computed as $\widetilde{\nabla}p_i(y_i) = -\mu_i$, where μ_i is an optimal Lagrange multiplier of Problem (2.8) (cf. Lemma 2.2). By collecting these facts together with (2.36), it follows that the blocks of $\widetilde{\nabla}\tilde{p}(z)$ can be computed as

$$\begin{aligned}\frac{\tilde{\partial}\tilde{p}(z)}{\partial z_{ij}} &= \widetilde{\nabla}p_i\Big([\Pi z]_i + \frac{b}{N}\Big) - \widetilde{\nabla}p_j\Big([\Pi z]_j + \frac{b}{N}\Big)\\ &= \mu_j - \mu_i, \qquad\qquad \forall\,(i,j)\in\mathcal{E}_u,\end{aligned} \tag{2.37}$$

where $\frac{\tilde{\partial}\tilde{p}(z)}{\partial z_{ij}}$ denotes the block of $\widetilde{\nabla}\tilde{p}(z)$ associated to z_{ij} and, for all $k\in\{1,\dots,N\}$, μ_k denotes an optimal Lagrange multiplier for the problem

$$\begin{aligned}\min_{x_k,\rho_k}\;\; & f_k(x_k) + M\rho_k\\ \text{subj. to }\; & g_k(x_k) \le [\Pi z]_k + \frac{b}{N} + \rho_k\mathbf{1}\\ & \rho_k \ge 0,\;\; x_k\in X_k.\end{aligned} \tag{2.38}$$

Using the positions (2.34) and (2.35) and using the formula for the computation of subgradients (2.37), the update (2.30) can be recast as

$$z_{ij}^{t+1} = \begin{cases} z_{ij}^t + \alpha^t\big(\mu_i^t - \mu_j^t\big), & \text{if } (i,j)\in\mathcal{E}^t,\\ z_{ij}^t, & \text{if } (i,j)\notin\mathcal{E}^t,\end{cases} \tag{2.39}$$

where μ_k^t denotes an optimal Lagrange multiplier of (2.38) with $z = z^t$ (with a slight abuse of notation[4]). Thus,

$$\begin{aligned}[\Pi z^{t+1}]_i + \frac{b}{N} &= \sum_{j\in\mathcal{N}_{i,u}} (z_{ij}^{t+1} - z_{ji}^{t+1}) + \frac{b}{N}\\ &\overset{(a)}{=} \underbrace{\sum_{j\in\mathcal{N}_{i,u}} (z_{ij}^t - z_{ji}^t) + \frac{b}{N}}_{y_i^t} + 2\alpha^t\sum_{j\in\mathcal{N}_i^t}\big(\mu_i^t - \mu_j^t\big)\\ &= y_i^{t+1}, \qquad i = 1,\dots,N,\end{aligned} \tag{2.40}$$

[3]This property of subgradients is the counterpart of the chain rule for differentiable functions.

[4] Indeed, the symbol μ_i^t was already defined in Algorithm 1. However, as per the equivalence of the two algorithms (which is being shown here), the two quantities coincide.

where (a) follows by (2.39). Therefore Algorithm 1 and the block subgradient method (2.30) are equivalent (up to a factor 2 in front of the step size α^t, which can be embedded in its definition). Before going on, let us state the following technical result.

Lemma 2.6. *For all $z \in \mathbb{R}^{S|\mathcal{E}_u|}$, the subgradients of $\tilde{p}$ at z are block-wise bounded, i.e.*

$$\|[\widetilde{\nabla}\tilde{p}(z)]_\ell\| \leq C_\ell, \qquad \forall\, \ell \in \{1, \ldots, B\},$$

where each $C_\ell > 0$ is a sufficiently large constant proportional to the parameter M.

Proof. Fix a block ℓ and suppose that it is associated to the edge (i, j). According to the previous discussion, the ℓ-th block of $\widetilde{\nabla}\tilde{p}(z)$ is equal to

$$[\widetilde{\nabla}\tilde{p}(z)]_\ell = \begin{bmatrix} \mu_j - \mu_i \\ \mu_i - \mu_j \end{bmatrix},$$

where each μ_k is a Lagrange multiplier of Problem (2.38). As shown in [83, Section III-B], it holds $\|\mu_k\|_1 \leq M$ for all $k \in \{1, \ldots, N\}$. Thus, the proof follows by using the equivalence of norms and by choosing a sufficiently large $C_\ell > 0$. ∎

2.4.4 Proof of Theorem 2.1

We are now able to provide the proof of Theorem 2.1. To show *(i)*, let us consider the block subgradient method (2.30) applied to Problem (2.28). Note that the function $\tilde{p}(z)$ is convex (because the functions p_i are convex, cf. [10, Section 5.4.4]) and its optimal cost is equal to $f^\star$, the optimal cost of Problem (2.1) (cf. Corollary 2.1, Lemma 2.4 and Lemma 2.3). By Lemma 2.6 and by the theorem's assumptions, we can apply Theorem 2.2 to conclude that, almost surely,

$$f^\star = \lim_{t\to\infty} \sum_{i=1}^{N} \tilde{p}_i\big(\{z_{ij}^t, z_{ji}^t\}_{j\in\mathcal{N}_{i,u}}\big) \overset{(a)}{=} \lim_{t\to\infty} \sum_{i=1}^{N} p_i(y_i^t) \overset{(b)}{=} \lim_{t\to\infty} \sum_{i=1}^{N} \big(f_i(x_i^t) + M\rho_i^t\big),$$

where (a) follows by definition of $\tilde{p}_i$ and by (2.40) and (b) follows by construction of (x_i^t, ρ_i^t).

To prove *(ii)*, it is possible to follow the same line of proof of [83]. However, as here we are considering a probabilistic setting in a primal decomposition framework, we report the proof for completeness. Let us consider the sample set $\bar{\Omega}$ for which point *(i)* of the theorem holds, and pick any sample path $\omega \in \bar{\Omega}$. Consider the primal sequence $\{(x_1^t, \ldots, x_N^t, \rho_1^t, \ldots, \rho_N^t)\}_{t\geq 0}$ generated by Algorithm 1 corresponding to ω. By summing over $i \in \{1, \ldots, N\}$ the inequality $g_i(x_i^t) \leq y_i^t + \rho_i^t\mathbf{1}$ (which holds by construction), it holds

$$\sum_{i=1}^{N} g_i(x_i^t) \leq \sum_{i=1}^{N} y_i^t + \sum_{i=1}^{N} \rho_i^t \mathbf{1} = \sum_{i=1}^{N} \rho_i^t \mathbf{1}. \tag{2.41}$$

Define $\rho^t = \sum_{i=1}^{N} \rho_i^t$. By construction, the sequence $\{(x_1^t, \ldots, x_N^t, \rho^t)\}_{t\geq 0}$ is bounded (as a consequence of point *(i)* and by continuity of the functions $f_i(x_i) + M\rho_i$), so that there exists a sub-sequence of indices $\{t_h\}_{h\geq 0} \subseteq \{t\}_{t\geq 0}$ such that the sequence $\{(x_1^{t_n}, \ldots, x_N^{t_h}, \rho^{t_h})\}_{h\geq 0}$ converges. Denote the limit point of such sequence as $(\bar{x}_1, \ldots, \bar{x}_N, \bar{\rho})$. From point *(i)* of the theorem, it follows that

$$\sum_{i=1}^{N} f_i(\bar{x}_i) + M\bar{\rho} = f^\star.$$

By Lemma 2.3, it must hold $\bar{\rho} = 0$. As the functions g_i are continuous, by taking the limit in (2.41) as $h \to \infty$, with $t = t_h$, it holds

$$\sum_{i=1}^{N} g_i(\bar{x}_i) \leq b + \bar{\rho}\mathbf{1} = b. \tag{2.42}$$

Therefore, the point $(\bar{x}_1, \ldots, \bar{x}_N)$ is an optimal solution of Problem (2.1). Since the sample path $\omega \in \bar{\Omega}$ is arbitrary, every limit point of $\{(x_1^t, \ldots, x_N^t)\}_{t\geq 0}$ is feasible and cost-optimal for Problem (2.1), almost surely. ■

Remark 2.2. As regards the modified algorithm for handling equality coupling constraints (cf. Section 2.3.3), we note that the analysis of (2.21)–(2.22) can be performed with almost the same line of proof used before. We do not discuss this in detail, however we point out that it is still possible to apply the change of coordinates (2.26) with obvious changes to the dimension of z and obtain the unconstrained master problem (2.28). The updates (2.22) are re-mapped to the z space in much the same way we did for the original update (2.18), resulting in the block subgradient method (2.39). Thus, the convergence properties of the modified algorithm remain unchanged. In particular, Theorem 2.1 holds with the assumption that $M > \|\mu^\star\|_1 + \|\lambda^\star\|_1$. △

2.4.5 Convergence Rates

The Distributed Primal Decomposition algorithm enjoys a sublinear rate for both constant and diminishing step-size rules. We refer to Appendix A.2 for the basic definitions regarding convergence rates. For constant step size, the cost sequence converges as $O(1/t)$, while for diminishing step size, the rate is $O(1/\log(t))$. The results provided here are expressed in terms of the quantity

$$f_{\text{best}}^t \triangleq \min_{\tau \le t} \sum_{i=1}^N \mathbb{E}[f_i(x_i^\tau) + M\rho_i^\tau],$$

where the expression in the expected value is the optimal cost of Problem (2.17) for agent i at time τ. Intuitively, f_{best}^t represents the best cost value obtained by the algorithm up to a certain iteration t, in an expected sense.

The following analysis is based on deriving convergence rates for our generalized block subgradient method and thus also complements the ones in, e.g. [39]. In the next lemma we derive a basic inequality.

Lemma 2.7. *Let Assumptions 2.1, 2.2 and 2.4 hold. Then, for all t* ≥ 0 *it holds*

$$2\left(\sum_{\tau=0}^t \alpha^\tau\right)(f_{\text{best}}^t - f^\star) \le \|z^0 - z^\star\|_W^2 + C\sum_{\tau=0}^t (\alpha^\tau)^2. \tag{2.43}$$

Proof. We consider the same line of proof of Theorem 2.2 up to (2.33), specialized for $\theta^t = z^t$, $\theta^\star = z^\star$ (an optimal solution of Problem (2.28)), with corresponding cost $\varphi^\star = \tilde{p}(z^\star) = f^\star$ (the optimal cost of Problem (2.1)). By taking the total expectation of (2.33) with respect to $\mathcal{F}^t$, it follows that, for all $t \geq 0$,

$$\begin{aligned}\mathbb{E}\Big[\|z^{t+1} - z^\star\|_W^2\Big] &= \mathbb{E}\Big\{\mathbb{E}\Big[\|z^{t+1} - z^\star\|_W^2 \big| \mathcal{F}^t\Big]\Big\} \\ &\le \mathbb{E}\Big[\|z^t - z^\star\|_W^2\Big] + (\alpha^t)^2 C - 2\alpha^t\Big(\mathbb{E}[\tilde{p}(z^t)] - f^\star\Big).\end{aligned}$$

Applying recursively the previous inequality yields

$$\mathbb{E}\Big[\|z^{t+1} - z^\star\|_W^2\Big] \le \|z^0 - z^\star\|_W^2 + C\sum_{\tau=0}^t (\alpha^\tau)^2 - 2\sum_{\tau=0}^t \alpha^\tau\Big(\mathbb{E}[\tilde{p}(z^\tau)] - f^\star\Big)$$

for all $t \geq 0$. By using the fact $\|z^{t+1} - z^\star\|_W^2 \geq 0$, we obtain

$$2\sum_{\tau=0}^t \alpha^\tau\Big(\mathbb{E}[\tilde{p}(z^\tau)] - f^\star\Big) \le \|z^0 - z^\star\|_W^2 + C\sum_{\tau=0}^t (\alpha^\tau)^2,$$

for all $t \geq 0$. The proof follows by combining the previous inequality with $\mathbb{E}[\tilde{p}(z^t)] \geq \min_{\tau \le t} \mathbb{E}[\tilde{p}(z^\tau)]$ and $\tilde{p}(z^\tau) = p(y^\tau) = \sum_{i=1}^N p_i(y_i^\tau) = \sum_{i=1}^N f_i(x_i^\tau) + M\rho_i^\tau$. ■

For constant step sizes, it immediately follows that the convergence rate is sublinear of the type $O(1/t)$, as formalized next.

Proposition 2.3 (Sublinear rate for constant step size). *Let the same assumptions of Theorem 2.1 hold (except for Assumption 2.3). Assume* $\alpha^t = \alpha > 0$ *for all t* ≥ 0. *Then, it* *holds*

$$f_{\text{best}}^t - f^\star \leq \frac{\|z^0 - z^\star\|_W^2}{2\alpha(t+1)} + \frac{C\alpha}{2}. \qquad \triangle$$

Note that the previous convergence rate has a term that goes to zero as t goes to infinity, plus a constant (positive) term. In general, without further assumptions, only convergence within a neighborhood of the optimum can be proved when a constant step size is used.

For the case of exact convergence with diminishing step size, we assume it has the form $\alpha^t = \frac{K}{t+1}$ with $K > 0$ (which satisfies Assumption 2.3). We can obtain a sublinear rate $\mathcal{O}(1/\log(t))$, as shown next.

Proposition 2.4 (Sublinear rate for diminishing step size). *Let the same assumptions of Theorem 2.1 hold. Assume $\alpha^t = \frac{K}{t+1}$ for all t ≥ 0, with $K > 0$. Then, it holds*

$$f_{\text{best}}^t - f^\star \leq \frac{\|z^0 - z^\star\|_W^2 + CK^2}{2K\log(t+2)}.$$

Proof. Let us set $\alpha^t = \frac{K}{t+1}$ in (2.43), then it holds

$$f_{\text{best}}^t - f^\star \leq \frac{\|z^0 - z^\star\|_W^2 + CK^2 \sum_{\tau=1}^{t+1} \frac{1}{\tau^2}}{2K \sum_{\tau=1}^{t+1} \frac{1}{\tau}}.$$

The proof follows by using the inequalities $\sum_{\tau=1}^{t} \frac{1}{\tau^2} \leq 1$ and $\sum_{\tau=1}^{t} \frac{1}{\tau} \geq \log(t+1)$. ■

We remark that the previous results can be also derived for the algorithm discussed in Section 2.2.5 with static communication graph. Essentially, the same arguments can be followed without block randomization in algorithm (2.30). For constant step sizes the rate can be proven to be

$$f_{\text{best}}^t - f^\star \leq \frac{\|z^0 - z^\star\|^2}{2\alpha(t+1)} + \frac{C\alpha}{2},$$

while for diminishing step sizes the rate is

$$f_{\text{best}}^t - f^\star \leq \frac{\|z^0 - z^\star\|^2 + CK^2}{2K\log(t+2)},$$

where here the quantities f_{best}^t and C are defined as $f_{\text{best}}^t \triangleq \min_{\tau \leq t} \sum_{i=1}^{N} f_i(x_i^\tau) + M\rho_i^\tau$ and $C \triangleq \sum_{\ell=1}^{B} C_\ell^2$.

2.5 Numerical Analysis

In this section, we show the efficacy of Distributed Primal Decomposition and validate the theoretical findings through a numerical study. We first concentrate on a simple example to show the main algorithm features. Then, we perform in-depth simulations on an electric vehicle charging scenario. All the simulations are performed with the disropt Python package (to be introduced in Chapter 5), with communication based on the Message Passing Interface (MPI).

2.5.1 Basic Nonsmooth Example

We begin by considering a network of $N = 100$ agents that must solve the constraint-coupled convex problem

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^{N} \|x_i - r_i\|_1 \\ \text{subj. to} \quad & \sum_{i=1}^{N} i \cdot x_i \leq \mathbf{0} \\ & -10 \cdot \mathbf{1} \leq x_i \leq 10 \cdot \mathbf{1}, \qquad i = 1, \ldots, N, \end{aligned} \tag{2.44}$$

where each $x_i \in \mathbb{R}^3$, and $r_i \in \mathbb{R}^3$ is a random vector with entries in the interval $[15, 20]$. Problem (2.44) is in the form (2.1) with the positions $f_i(x_i) = \|x_i - r_i\|_1$, $X_i = \{x_i \in \mathbb{R}^3 | -10 \cdot \mathbf{1} \leq x_i \leq 10 \cdot \mathbf{1}\}$ and $g_i(x_i) = i \cdot x_i$. Note that the objective function and the coupling constraint functions are convex but not smooth. As for the communication graph, we generate an Erdős-Rényi graph with edge probability 0.2. The edge activation probabilities σ_{ij} are randomly picked in $[0.3, 0.9]$. The generated graph is shown in Figure 2.6.

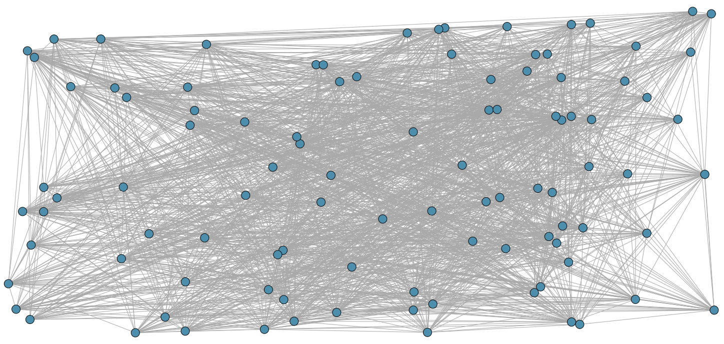


Figure 2.6: Underlying communication graph $\mathcal{G}_u$ for the basic nonsmooth example.

In order to apply the Distributed Primal Decomposition algorithm, we compute a valid value of the parameter M appearing in Problem (2.17) by using Proposition 2.2

with the Slater vector $(\bar{x}_1, \dots, \bar{x}_N)$ with each $\bar{x}_i = -10 \cdot \mathbf{1}$. After performing all the computations, we obtain the condition $M > 1$ and we finally choose $M = 6$. The Distributed Primal Decomposition algorithm is initialized at $y_i^0 = 0$ for all $i \in \{1, \dots, N\}$ and the step size $\alpha^t = 1/(t+1)^{0.6}$ is used (which satisfies Assumption 2.3). The simulation results are reported in Figures 2.7 and 2.8. The asymptotic behavior of Theorem 2.1 is confirmed.

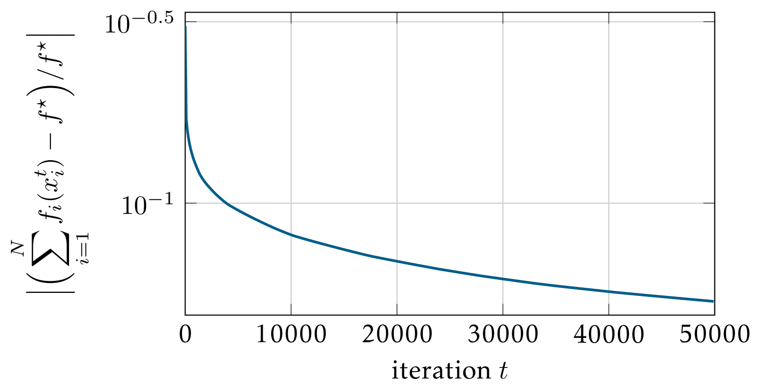


Figure 2.7: Evolution of the normalized cost error for the illustrative example with $N = 100$ agents.

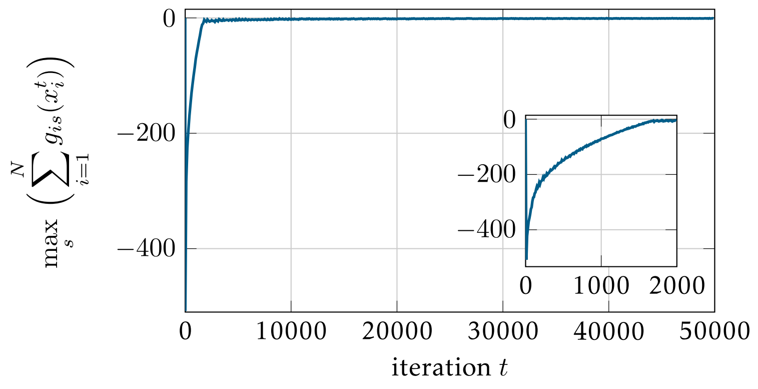


Figure 2.8: Evolution of the coupling constraints for the illustrative example. A value below zero means that the solution computed by the algorithm at that iteration is feasible. The inset figure shows the behavior of the algorithm in the early iterations.

2.5.2 Electric Vehicle Charging Problem

Let us now consider the charging of Plug-in Electric Vehicles (PEVs), which is formulated in detail in [121] and is slightly changed here in order to better highlight the algorithm behavior. The simulations reported in the remainder of this section are all referred to this application scenario.

The problem consists of determining an optimal charging schedule of N electric vehicles. Each vehicle i has an initial state of charge E_i^{init} and a target state of charge

E_i^{ref} that must be reached within a time horizon of 8 hours, divided into $T = 12$ time slots of $\Delta T = 40$ minutes. Vehicles must further satisfy a coupling constraint, which is given by the fact that the total power drawn from the (shared) electricity grid must not exceed $P^{\max} = N/2$. In this thesis, we consider the "charge-only" case. In order to make sure the local constraint set are convex (cf. Assumption 2.1), we drop the additional integer constraints considered in [121]. Thus, the vehicles optimize their charging rate rather than activating or de-activating the charging mode at each time slot. Formally, the resulting linear program is

$$\begin{aligned}
\min_{x_1,\dots,x_N} \quad & \sum_{i=1}^{N} c_i^\top x_i \\
\text{subj. to} \quad & \sum_{i=1}^{N} A_i x_i \le b, \\
& x_i \in X_i, \qquad i = 1, \dots, N,
\end{aligned}$$

where the local constraint sets X_i are compact polyhedra and a total of $S = 12$ coupling constraints are present. For a complete reference on the other quantities involved in the problem and not explicitly specified here, we refer the reader to the extended formulation in [121].

We consider a network of $N = 50$ agents where the underlying graph $\mathcal{E}_u$ is generated as an Erdős-Rényi graph with edge probability 0.2. The edge activation probabilities σ_{ij} are randomly picked in $[0.3, 0.9]$. In particular, in the next subsections we *(i)* compare our algorithm with the state of the art, *(ii)* discuss the parameter M and *(iii)* show the convergence rate.

2.5.3 Comparison with State of the Art

We compare Distributed Primal Decomposition with the Distributed Dual Subgradient algorithm [44]. As for the algorithm tuning (i.e. the step size α^t in the update (2.18) and the parameter M appearing in Problem (2.17)), we choose $M = 30$ and the diminishing step size $\alpha^t = \frac{1}{(t+1)^{0.6}}$. Our algorithm is initialized in $y_i^0 = b/N$ for all i and the Distributed Dual Subgradient algorithm is initialized in $\lambda_i^0 = \mathbf{0}$ for all i. In Figure 2.9, the cost error of both algorithms is shown, compared with the result of a centralized problem solver. For our algorithm, the symbol x_i^t represents the local solution of Problem (2.17) at time t, while for the Distributed Dual Subgradient, the same symbol represents the (unweighted) running average of the local solutions over the past iterations. The figure highlights that, in this simulation, Distributed Primal Decomposition reached almost exact cost convergence shortly after 10000 iterations with a sudden change of approximately 10 orders of magnitude. In principle, for the

Distributed dual subgradient, it is not possible to have such rapid changes because of the use of running averages.

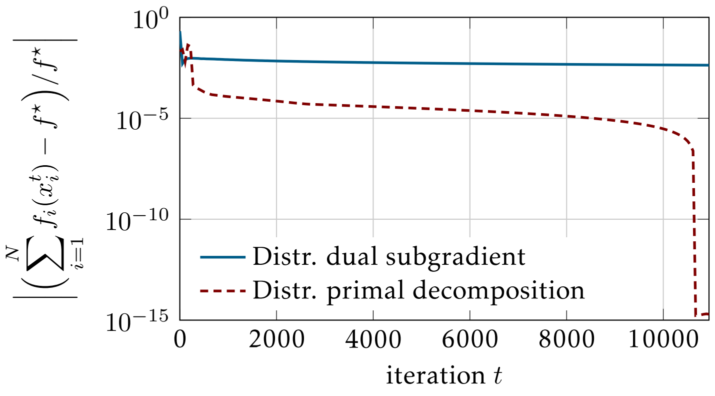


Figure 2.9: Evolution of the normalized cost error for the comparative study with the state of the art.

In Figure 2.10, we show the value of the coupling constraints. The picture highlights that both algorithms are able to provide feasible solutions within less than 500 iterations, confirming the primal recovery property.

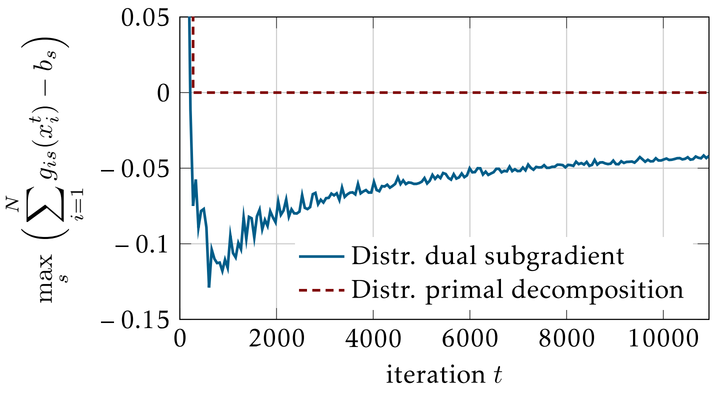


Figure 2.10: Evolution of the coupling constraint for the comparative study with the state of the art. A value below zero means that the solution computed by the algorithm at that iteration is feasible.

2.5.4 Impact of the Parameters

We also perform a numerical comparison of the algorithm behavior for different values of the parameter M (see also Section 2.3.4). Under the same set-up of the previous simulation, we use a different initialization to guarantee the requirements imposed by Theorem 2.1 and also to create some asymmetry among the initial allocations of the agents. Thus, in this simulation we consider the initialization rule $y_i^0 = 5(N-2i)\mathbf{1} + b/N$ for all i, which satisfies $\sum_{i=1}^{N} y_i^0 = b$.

In Figure 2.11 we plot the cost error, including the extra penalty term $\sum_{i=1}^{N} M\rho_i^t$, for three different values of M (all of which satisfy the assumption $M > \|\mu^\star\|_1$). It can be seen that the slope of the curve decreases as M increases, which agrees with the fact that the larger is M, the larger is the set in which subgradients can be found (Lemma 2.6).

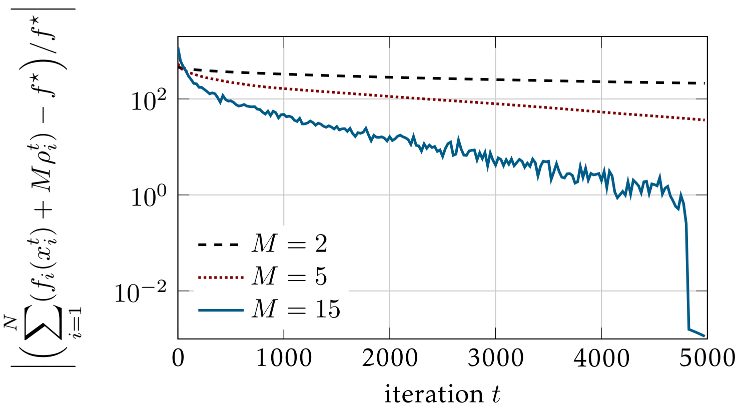


Figure 2.11: Evolution of the normalized cost error for different values of M, under diminishing step size.

Figure 2.12 shows the maximum value of ρ_i^t among agents. Recall that ρ_i^t is an upper bound on the violation of the local allocation y_i^t. The picture underlines that such a quantity is forced to zero faster as M gets bigger. This can be intuitively explained by the fact that larger values of the penalty $M\rho_i$ drive the algorithm more quickly towards feasibility of the coupling constraint.

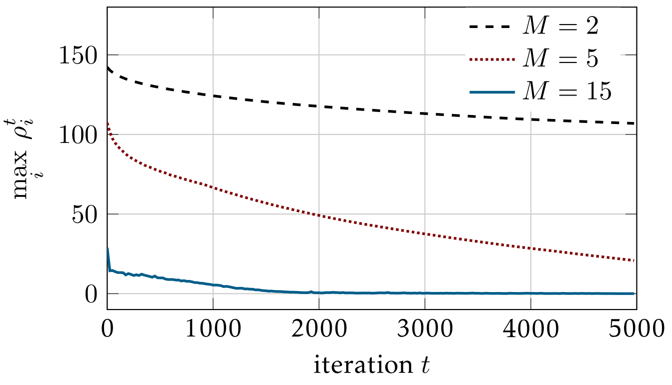


Figure 2.12: Evolution of the value of $\max_i \rho_i^t$ for varying values of M. The quantity represents an upper bound on the coupling constraint violation.

2.5.5 Comparative Study on Convergence Rates

We finally perform a simulation to point out the different behavior of the algorithm with constant and diminishing step sizes. Under the same set-up of the previous example, with $M = 10$, we run the algorithm with the diminishing step-size law $\alpha^t = \frac{0.5}{(t+1)^{0.6}}$ and with the constant step size $\alpha^t = 0.01$. As before, agents initialize their local allocation at $y_i^0 = 5(N-2i)\mathbf{1} + b/N$ for all i.

Figure 2.13 shows the different algorithm behavior under the two step size choices. For constant step size, the algorithm converges within a certain tolerance (which is seen in the picture at around iteration 6000), confirming the observations in Section 2.4.5. Moreover, the sublinear behavior with the diminishing step size is confirmed. Interestingly, in this example the constant step size behaved linearly up to iteration 4000 and superlinearly in the interval 4000–6000, therefore performing much better than the sublinear bound in Proposition 2.3.

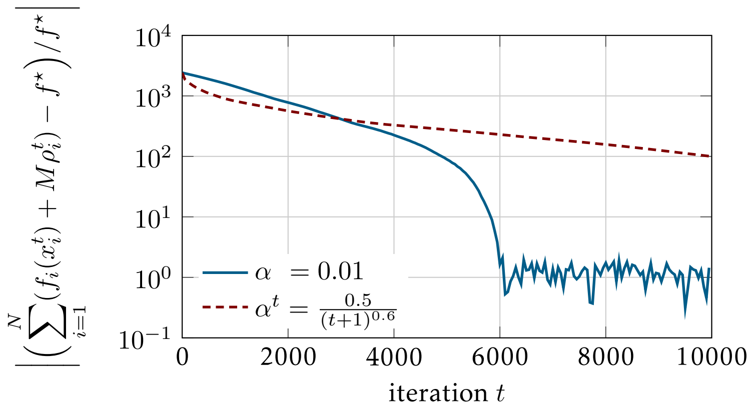


Figure 2.13: Evolution of the cost error for the comparative study on step sizes.

2.6 Extension to Unknown Cost Functions

In this section, we discuss an extension of the Distributed Primal Decomposition algorithm to the case of a-priori unknown cost functions. We first introduce the optimization set-up. Then, we describe the distributed algorithm and provide the convergence analysis, which is corroborated with a numerical example. The results of this section are based on [20].

2.6.1 Constraint-coupled Set-up with Unknown Costs

Let us consider again N agents that must solve Problem (2.1), recalled here

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^N f_i(x_i) \\ \text{subj. to} \quad & \sum_{i=1}^N g_i(x_i) \le b, \\ & x_i \in X_i, \qquad i = 1,\ldots,N, \end{aligned}$$

where all the symbols have been already defined in Section 2.2.1. We also maintain the same assumptions, namely Assumption 2.1 (convexity and compactness) and Assumption 2.2 (Slater's constraint qualification). For the time being, we assume that there is only one coupling constraint ($S = 1$), i.e. that $g_i : \mathbb{R}^{n_i} \to \mathbb{R}$ for all i and $b \in \mathbb{R}$.

The main novelty of this section is that we consider the cost functions are *not* known in advance and must be estimated online. This challenging assumption can model, for instance, situations in which evaluation of the cost function is computationally intensive and can be done only for a small number of points. For this reason, we assume that agents are equipped with an estimation mechanism that progressively refines their knowledge of the objective functions. We model the estimation mechanism as a black-box *oracle* that can be queried to provide estimations of the cost function. Since each agent i has its own cost function f_i, we assume that each agent has its own instance of the oracle providing estimated versions of the cost function over time. As these oracles will be embedded within the distributed algorithm, we denote by $f_i^t(\cdot) = \textsc{Oracle}(i, t)$ the output of oracle i at an iteration $t \in \mathbb{N}$. We do not impose a specific estimation mechanism, so that each agent i can use the most appropriate method depending on the cost function at hand. The only assumption we make on the oracles is formalized next.

Assumption 2.6 (Oracles). *For each agent $i \in \{1, \ldots, N\}$, the estimated functions $f_i^t(\cdot) = \textsc{Oracle}(i, t)$ converge uniformly to the true cost function $f_i(\cdot)$.* $\triangle$

As an example, if the objective function f_i is parametric with known form but unknown parameters, one can apply a recursive least squares approach to obtain iteratively refined approximations. In such a case, for the estimation mechanism one should choose a persistently exciting input to sample the function f_i. Assumption 2.6 is then satisfied.

We next discuss how the Distributed Primal Decomposition algorithm can be extended to handle the described scenario. To this end, the new distributed algorithm makes use of the estimated cost functions in place of the original ones, but nevertheless it will be able to solve Problem (2.1) exactly.

We assume the exchange of information among the agents occurs according to a fixed communication model as described in Section 2.2.1. The assumption of static graph can be relaxed to handle the more general case of random time-varying graphs using the techniques discussed in Section 2.3. However, since this is not the main focus of this section, we prefer to maintain the assumption of static network to keep the discussion simple.

2.6.2 Distributed Algorithm Description

The distributed algorithm that we propose has the same structure as the Distributed Primal Decomposition algorithm described in Section 2.2.5. However, instead of minimizing the true cost function f_i in the local problem, each agent i minimizes the estimated version f_i^t obtained by the oracle.

Formally, each agent i maintains a local allocation estimate $y_i^t \in \mathbb{R}$. At each iteration $t \in \mathbb{N}$, agent i queries the black-box oracle and obtains an updated estimate of its cost function, denoted by f_i^t. Then, it computes $\mu_i^t \in \mathbb{R}$ as a Lagrange multiplier of Problem (2.45) and exchanges it with neighbors. Finally, it updates y_i^t according to (2.46). The following table summarizes the algorithm from the perspective of node i, where we recall that α^t is the step size and the notation "μ_i :" means that μ_i is the Lagrange multiplier associated to the constraint $g_i(x_i) \leq y_i^t + \rho_i$.

Algorithm 2 Distributed Primal Decomposition with Estimated Costs

```
**Initialization**: $y_i^0$ such that $\sum_{i=1}^N y_i^0 = b$

**For** $t = 0, 1, 2, \dots$

  **Query oracle** and obtain local cost estimate $f_i^t(\cdot) = \textsc{Oracle}(i, t)$

  **Compute** $((x_i^t, \rho_i^t), \mu_i^t)$ as a primal-dual solution of

  $$\begin{aligned} \min_{x_i, \rho_i} \quad & f_i^t(x_i) + M\rho_i \\ \text{subj. to} \quad & \mu_i : g_i(x_i) \leq y_i^t + \rho_i \\ & x_i \in X_i \end{aligned} \tag{2.45}$$

  **Receive** $\mu_j^t$ from neighbors $j \in \mathcal{N}_i$ and update

  $$y_i^{t+1} = y_i^t + \alpha^t \sum_{j \in \mathcal{N}_i^t} \left(\mu_i^t - \mu_j^t\right) \tag{2.46}$$
```

Note that the true cost function $f_i(x_i)$ is never used by each agent i, indeed only its estimated version $f_i^t(x_i)$ appears in Problem (2.45). Using a surrogate function in place of the true one can significantly reduce the computational cost of solving Problem (2.45). By using the fact that the surrogate function f_i^t approaches the true one as t increases (cf. Assumption 2.6), we will show that the algorithm is still able to compute the optimal solution of Problem (2.1). Note also that all the privacy and scalability properties of the original algorithm are maintained. Indeed, no private data is exchanged with the neighbors and the amount of local computation remains bounded as N increases. In the following subsection, we provide a convergence analysis of the algorithm. As the analysis requires several intermediate steps, the reader interested in the main result can directly look at Theorem 2.3.

2.6.3 Algorithm Analysis

Intuitively, in order to analyze Algorithm 2 one should follow a line of proof similar to the one used in Section 2.4 but without block randomization (since the communication graph is static). However, we must appropriately take into account the fact that the local problems do not have the true cost function.

Properties of the Primal Functions

Central to the analysis is the structure of the primal functions $p_i(y_i)$. To see this, we recall that subgradients of the primal functions are Lagrange multipliers of the associated problem (cf. Lemma 2.2). Since we are using estimated cost functions in place of the original ones, we must expect that the computed Lagrange multipliers μ_i^t are erroneous (if compared to the "standard" algorithm with the true cost functions) and, thus, also the subgradients of $p_i(y_i)$. We will see that, as a consequence of Assumption 2.6, such errors on the subgradients asymptotically go to zero.

In what follows, we study in more detail the structure of the primal functions. Since we are considering a single coupling constraint ($S = 1$), the subgradients of p_i are actually subderivatives. As this will not impact much on our analysis, we will denote them with the symbol of derivatives, i.e. $p_i'(y_i)$. Whether the symbol denotes a derivative or a subderivative will always be clear from the context. In the forthcoming analysis, we will need the following result. The proof relies on duality-based arguments similar to the ones in [83] together with Lemma 2.2, however we report it for completeness.

Lemma 2.8. *For all $y_i \in \mathbb{R}$, the subderivatives of p_i satisfy $-M \leq p_i'(y_i) \leq 0$, for all $i \in \{1, \ldots, N\}$.*

Proof. Fix an agent i. By Lemma 2.2, it holds $p_i'(y_i) = -\mu_i(y_i)$, where $\mu_i(y_i)$ is a dual optimal solution of subproblem (2.8), i.e.

$$\begin{aligned} \min_{x_i \in X_i,\, \rho_i \geq 0} \; & f_i(x_i) + M\rho_i \\ \text{subj. to} \; & g_i(x_i) \leq y_i + \rho_i, \end{aligned}$$

associated to the constraint $g_i(x_i) \leq y_i + \rho_i$. Let us derive the dual function,

$$\begin{aligned} q_i(\mu_i) &= \inf_{x_i \in X_i,\, \rho_i \geq 0} \Big(f_i(x_i) + M\rho_i + \mu_i(g_i(x_i) - y_i - \rho_i) \Big) \\ &= \begin{cases} \min\limits_{x_i \in X_i} \Big(f_i(x_i) + \mu_i(g_i(x_i) - y_i) \Big) & \text{if } \mu_i \leq M, \\ -\infty & \text{otherwise.} \end{cases} \end{aligned}$$

Thus, the dual problem reads

$$\max_{0 \le \mu_i \le M} \; q_i(\mu_i),$$

from which we see that any optimal solution satisfies $0 \le \mu_i(y_i) \le M$. Therefore, it follows that $-M \le p_i'(y_i) \le 0$. ∎

Let us define for all $i \in \{1, \dots, N\}$ the scalars

$$y_i^{\text{MIN}} \triangleq \min_{x_i \in X_i} \; g_i(x_i), \tag{2.47a}$$

$$y_i^{\text{MAX}} \triangleq \max_{x_i \in X_i} \; g_i(x_i). \tag{2.47b}$$

from which it directly follows that any locally feasible solution $x_i \in X_i$ satisfies $y_i^{\text{MIN}} \le g_i(x_i) \le y_i^{\text{MAX}}$.

The next important lemma regards the structure of the primal functions, which is graphically represented in Figure 2.14. Intuitively, the numbers y_i^{MIN}, y_i^{MAX} represent the minimum and maximum resource that each agent i can use. As the allocation y_i ranges from y_i^{MIN} to y_i^{MAX}, the optimal cost of the subproblem (2.8) decreases since the constraint $g_i(x_i) \le y_i + \rho_i$ becomes less and less stringent. Eventually, for allocations greater than y_i^{MAX}, the cost cannot be further improved and $p_i(y_i)$ becomes constant. Instead, if $y_i \le y_i^{\text{MIN}}$, optimal solutions to Problem (2.8) must compensate for the gap $y_i^{\text{MIN}} - y_i$ with an appropriate choice of ρ_i. The cost penalty $M\rho_i$ gives rise to the linear behavior.

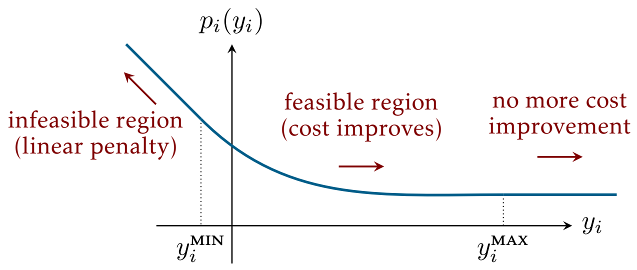


Figure 2.14: Illustration of Lemma 2.9. See the text for details.

Lemma 2.9. *For all $i \in \{1, \dots, N\}$, the primal function $p_i(y_i)$ satisfies:*

(i) $p_i(y_i) = p_i(y_i^{\text{MAX}})$ for all $y_i \ge y_i^{\text{MAX}}$;

(ii) $p(y_i) = -My_i + q_i$ for all $y_i \le y_i^{\text{MIN}}$,

with $q_i = p_i(y_i^{\text{MIN}}) + My_i^{\text{MIN}}$.

Proof. To ease the notation, we drop the index i. Let us show *(i)*. Let $\bar{y} \ge y^{\text{MAX}}$ and let $(x^{\text{MAX}}, \rho^{\text{MAX}})$ be an optimal solution of Problem (2.8) with $y = y^{\text{MAX}}$. To prove that $p(\bar{y}) = p(y^{\text{MAX}})$, we must demonstrate that $(x^{\text{MAX}}, \rho^{\text{MAX}})$ is an optimal solution of Problem (2.8) when $y = \bar{y}$. By construction, it holds

$$g(x^{\text{MAX}}) \leq y^{\text{MAX}} + \rho^{\text{MAX}} \leq \bar{y} + \rho^{\text{MAX}},$$

thus $(x^{\text{MAX}}, \rho^{\text{MAX}})$ is a feasible solution. Suppose that it is not optimal, then there exists $(\tilde{x}, \tilde{\rho})$ such that $\tilde{x} \in X$, $\tilde{\rho} \geq 0$, $g(\tilde{x}) \leq \bar{y} + \tilde{\rho}$ and

$$f(\tilde{x}) + M\tilde{\rho} < f(x^{\text{MAX}}) + M\rho^{\text{MAX}},$$

i.e. $(\tilde{x}, \tilde{\rho})$ has a lower cost than $(x^{\text{MAX}}, \rho^{\text{MAX}})$. However, by (2.47b) and $\tilde{\rho} \geq 0$ we have

$$g(\tilde{x}) \leq \max_{x_i \in X_i} g_i(x_i) = y^{\text{MAX}} \leq y^{\text{MAX}} + \tilde{\rho},$$

and $(\tilde{x}, \tilde{\rho})$ would be a feasible solution for Problem (2.8) with $y = y^{\text{MAX}}$ with a cost lower than $(x^{\text{MAX}}, \rho^{\text{MAX}})$, contradicting the assumption that $(x^{\text{MAX}}, \rho^{\text{MAX}})$ is optimal.

Now we prove *(ii)*. Let $\bar{y} \leq y_i^{\text{MIN}}$ and let $(x^{\text{MIN}}, \rho^{\text{MIN}})$ be optimal solution of Problem (2.8) with $y = y^{\text{MIN}}$. It holds $p(y^{\text{MIN}}) = f(x^{\text{MIN}}) + M\rho^{\text{MIN}}$ and $g(x^{\text{MIN}}) \leq y^{\text{MIN}} + \rho^{\text{MIN}}$. The goal is to show that

$$p(\bar{y}) = p(y^{\text{MIN}}) + M(y^{\text{MIN}} - \bar{y}) = f(x^{\text{MIN}}) + M(\rho^{\text{MIN}} + y^{\text{MIN}} - \bar{y}),$$

i.e. that $(x^{\text{MIN}}, \rho^{\text{MIN}} + y^{\text{MIN}} - \bar{y})$ is an optimal solution of Problem (2.8) with the given y. By using the assumption on $(x^{\text{MIN}}, \rho^{\text{MIN}})$ and the fact that $y^{\text{MIN}} - \bar{y} \geq 0$ (by (2.47a)), we can immediately show feasibility,

$$g(x^{\text{MIN}}) \leq y^{\text{MIN}} + \rho^{\text{MIN}} \leq y^{\text{MIN}} + \rho^{\text{MIN}} + y^{\text{MIN}} - \bar{y}.$$

Suppose that $(x^{\text{MIN}}, \rho^{\text{MIN}} + y^{\text{MIN}} - \bar{y})$ is not optimal for Problem (2.8) with the given y. Then, there exists $(\tilde{x}, \tilde{\rho})$ such that $\tilde{x} \in X$, $\tilde{\rho} \geq 0$, $g(\tilde{x}) \leq \bar{y} + \tilde{\rho}$ and

$$\begin{aligned} f(\tilde{x}) + M\tilde{\rho} &< f(x^{\text{MIN}}) + M(\tilde{\rho} + y^{\text{MIN}} - \bar{y}) \\ &= p(y^{\text{MIN}}) + M(y^{\text{MIN}} - \bar{y}), \end{aligned}$$

from which it follows that $f(\tilde{x}) + M(\tilde{\rho} + y^{\text{MIN}} - \bar{y}) < p(y^{\text{MIN}})$, i.e. the vector $(\tilde{x}, \tilde{\rho} + y^{\text{MIN}} - \bar{y})$ has a lower cost than $(x^{\text{MIN}}, \rho^{\text{MIN}})$. Moreover, using again $y^{\text{MIN}} - \bar{y} \geq 0$, we obtain

$$g(\tilde{x}) \leq \bar{y} + \tilde{\rho} \leq \bar{y} + \tilde{\rho} + y^{\text{MIN}} - \bar{y},$$

from which it follows that $(\tilde{x}, \tilde{\rho} + y^{\text{MIN}} - \bar{y})$ is a feasible solution for Problem (2.8) with the given y with a cost lower than $(x^{\text{MIN}}, \rho^{\text{MIN}})$, contradicting the assumption that $(x^{\text{MIN}}, \rho^{\text{MIN}})$

is optimal. ■

Uniform Convergence of Estimated Subgradients

Next we provide a sequence of results that show that the estimated subgradients of the primal functions converge to the true ones. This fact will be necessary in the proof of Theorem 2.3 to assess that Algorithm 2 asymptotically recovers the behavior of Algorithm 1.

Similarly to the definition of primal function (2.8), let us define $p_i^t(y_i)$ as the optimal cost of the subproblem with the surrogate function at time t with allocation $y_i \in \mathbb{R}$, i.e.

$$\begin{aligned} p_i^t(y_i) \triangleq \min_{x_i, \rho_i} \; & f_i^t(x_i) + M\rho_i \\ \text{subj. to } \; & g_i(x_i) \leq y_i + \rho_i \\ & x_i \in X_i, \;\; \rho_i \geq 0. \end{aligned} \tag{2.48}$$

We will work with the dual problems associated to (2.8) and (2.48). The dual problem of (2.8) was derived in the proof of Lemma 2.8. Similarly, one can derive the dual problem associated to (2.48), which is

$$\begin{aligned} \max_{\mu_i} \; & \underbrace{\left[\min_{x_i \in X_i} \big(f_i^t(x_i) + \mu_i (g_i(x_i) - y_i) \big) \right]}_{\triangleq q_i^t(\mu_i)} \\ \text{subj. to } \; & 0 \leq \mu_i \leq M, \end{aligned} \tag{2.49}$$

for all $i \in \{1, \ldots, N\}$. Note that for all $y_i \in \mathbb{R}$ the function q_i^t is continuous and thus the maximum in (2.49) exists finite.

Lemma 2.10. *Let Assumption 2.6 hold. Then, the dual function sequence $\{q_i^t\}_t$ converges to q_i, uniformly in $\mu_i \in D_i = \{\mu_i : \mu_i \leq M\}$ and $y_i \in \mathbb{R}$, where D_i is the domain of q_i and q_i^t.*

Proof. To ease the notation, we drop the index i. Since our aim is to prove uniformity with respect to both μ and y, in this proof we denote the functions as $q^t(\mu, y)$ and $q(\mu, y)$ to show explicitly the dependence of q^t and q on both μ and y. By definition, we have that, for any fixed $\mu \in D$ and $y \in \mathbb{R}$,

$$q(\mu, y) = \min_{x \in X} \big(f(x) + \mu(g(x) - y) \big) \leq f(x) + \mu(g(x) - y), \quad \text{for all } x \in X, \tag{2.50}$$

and also

$$q^t(\mu, y) = \min_{x \in X} \big(f^t(x) + \mu(g(x) - y) \big) \leq f^t(x) + \mu(g(x) - y) \quad \text{for all } x \in X. \tag{2.51}$$

By the uniform convergence of $\{f^t\}_t$ (cf. Assumption 2.6), we have that for all $\varepsilon > 0$ there exists $N > 0$ such that for all $t \geq N$ it holds

$$f^t(x) - f(x) < \varepsilon \quad \text{and} \quad f(x) - f^t(x) < \varepsilon$$

for all $x \in X$. Subtracting (2.50) from (2.51) and using the uniform convergence we obtain, for all $t \geq N$ and for any $\mu \in D$ and $y \in \mathbb{R}$,

$$\begin{aligned} q^t(\mu, y) - q(\mu, y) &\leq f^t(x) - f(x) \qquad \text{for all } x \in X \\ &< \varepsilon. \end{aligned}$$

Similarly, subtracting (2.51) from (2.50), we obtain, for all $t \geq N$ and for any $\mu \in D$ and $y \in \mathbb{R}$,

$$q(\mu, y) - q^t(\mu, y) < \varepsilon.$$

Since the previous results do not actually depend on the chosen μ or y, they are uniform in μ and y. Therefore we have proven that for all $\varepsilon > 0$ there exists $N \geq 0$ such that $|q^t(\mu) - q(\mu)| < \varepsilon$ for all $t \geq N$, uniformly in $y \in \mathbb{R}$ and $\mu \in D$. ■

Owing to Lemma 2.2, a subderivative of the time-varying primal function p_i^t at any $y_i \in \mathbb{R}$ is given by $p_i^{t\prime}(y_i) = -\mu_i^t(y_i)$, where $\mu_i^t(y_i)$ is a maximum of $q_i^t(\cdot, y_i)$ (with respect to μ_i) in the interval $0 \leq \mu_i \leq M$. Since there may be several maxima, we assume that there is a tie-break rule such that, for a given $y_i \in \mathbb{R}$, the same maximum is always selected. This tie-break rule allows us to define for all $t \geq 0$ a well-defined function $\mu_i^t : \mathbb{R} \to \mathbb{R}$ such that

$$\mu_i^t(y_i) \in \operatorname*{argmax}_{0 \leq \mu_i \leq M} q_i^t(\mu_i, y_i), \tag{2.52}$$

and similarly for the subderivative of p_i,

$$p_i'(y_i) = -\mu_i(y_i) \in \operatorname*{argmax}_{0 \leq \mu_i \leq M} q_i(\mu_i, y_i).$$

The assumption on the tie-break rule will be formalized next in Assumption 2.7.

Lemma 2.11. *Let Assumption 2.6 hold. Then, the subderivative function $p_i^{t\prime}(y_i)$ converges uniformly to $p_i'(y_i)$, i.e. for all $\eta > 0$ there exists $N > 0$ such that $|p_i^{t\prime}(y_i) - p_i'(y_i)| < \eta$ for all $t \geq N$ and $y_i \in \mathbb{R}$.*

Proof. To ease the notation, we drop the index i. Since $p^{t\prime}(y) = -\mu^t(y)$, we need to prove that the function $\mu^t(y)$ converges uniformly to $\mu(y)$. By definition (2.52), the function sequence $\{\mu^t(y)\}_{t \in \mathbb{N}}$ is uniformly bounded in $[0, M]$. Thus we can extract a convergent subsequence $\{\mu^{t_n}(y)\}_{n\in\mathbb{N}}$ and denote by $\bar{\mu}(y)$ its limit function. Let us first show that the limit function maps each y to a maximum of $q(\mu, y)$ over $\mu \in [0, M]$. For all $y \in \mathbb{R}$, by optimality of $\mu^{t_n}(y)$ for q^{t_n} it holds

$$q^{t_n}(\mu(y), y) \leq q^{t_n}(\mu^{t_n}(y), y), \qquad \forall n \in \mathbb{N}.$$

By taking the limit as $n \to \infty$ and by using Lemma 2.10, we obtain for all $y \in \mathbb{R}$

$$q(\mu(y), y) \leq q(\bar{\mu}(y), y).$$

However, by optimality of $\mu(y)$ for q it also holds $q(\mu(y), y) \geq q(\bar{\mu}(y), y)$ for all $y \in \mathbb{R}$. Thus, equality follows for all y and therefore

$$\bar{\mu}(y) \in \operatorname*{argmax}_{\mu\in[0,M]} \; q(\mu, y), \qquad \forall y \in \mathbb{R}. \tag{2.53}$$

We finally need to show that $\bar{\mu}(y)$ is also the smallest number in $\operatorname{argmax}_{\mu\in[0,M]} \; q(\mu, y)$. Let us denote

$$\begin{aligned} Q^\star(y) &= \operatorname*{argmax}_{0\leq\mu\leq M} \; q(\mu, y), \\ Q^t(y) &= \operatorname*{argmax}_{0\leq\mu\leq M} \; q^t(\mu, y). \end{aligned}$$

By (2.52), for all $y \in \mathbb{R}$ it holds

$$\mu^{t_n}(y) \leq \mu, \qquad \forall \mu \in Q^t t_n(y).$$

By taking the limit as n goes to infinity and by using (2.53), we obtain for all $y \in \mathbb{R}$

$$\bar{\mu}(y) \leq \mu, \qquad \forall \mu \in Q^\star(y),$$

and the proof follows. ■

Estimated Subgradients are Epsilon-Subgradients

The uniform convergence of the the estimated subgradients is not enough to prove that Algorithm 2 is able to asymptotically recover optimality. However, it turns out that the subgradients of the time-varying primal functions p_i^t are so-called ϵ-subgradients of the true primal function p_i. Formally, given a convex function $\varphi(\theta) : \mathbb{R}^n \to \mathbb{R}$, an ϵ-subgradient of φ at some $\theta_0 \in \mathbb{R}^n$, is a vector $\widetilde{\nabla}_\epsilon \varphi(\theta_0) \in \mathbb{R}^n$ satisfying

$$\varphi(\theta) \geq \varphi(\theta_0) + \widetilde{\nabla}_\epsilon \varphi(\theta_0)^\top (\theta - \theta_0) - \epsilon, \qquad \forall \theta \in \mathbb{R}^n,$$

In the following important proposition, we prove a central result for the analysis.

Proposition 2.5. *Let Assumptions 2.1, 2.2 and 2.6 hold. Then, there exists a sequence* $\{\epsilon_i^t\}_{t\in\mathbb{N}}$ *of non-negative scalars such that for all* $t \in \mathbb{N}$ *and* $y_i \in \mathbb{R}$ *it holds*

$$p_i(z) \geq p_i(y_i) + (z - y_i)^\top p_i^{t\prime}(y) - \epsilon_i^t, \qquad \forall\, z \in \mathbb{R}. \tag{2.54}$$

Moreover, $\lim\limits_{t\to\infty} \epsilon_i^t = 0$.

Proof. To ease the notation, we drop the index i. We begin by proving that, for each fixed t and $y \in \mathbb{R}$, there exists a finite number ϵ^t satisfying (2.54). We will then find an upper bound of ϵ^t, independent of y, that goes to zero as t goes to infinity, which yields the desired result.

Fix $t \in \mathbb{N}$ and $y_0 \in \mathbb{R}$. Let us define $\epsilon^t(y_0)$ as the smallest non-negative number satisfying (2.54) at y_0, i.e.

$$\begin{aligned} \epsilon^t(y_0) = \inf_{\epsilon}\ & \epsilon \\ \text{subj. to}\ & \epsilon \geq p(y_0) + (z - y_0)p^{t\prime}(y_0) - p(z), \qquad \forall\, z \in \mathbb{R}. \end{aligned} \tag{2.55}$$

By definition, $\epsilon^t(y_0) \geq 0$. We must prove that the infimum in (2.55) is attained at a real number (i.e. that $\epsilon^t(y_0) \neq +\infty$). The optimization problem (2.55) is in epigraph form and can be equivalently rewritten as

$$\epsilon^t(y_0) = \sup_{z\in\mathbb{R}} \big[p(y_0) + (z - y_0)p^{t\prime}(y_0) - p(z)\big]. \tag{2.56a}$$

Using the properties of the sup, we can rewrite $\epsilon^t(y_0)$ as

$$\epsilon^t(y_0) = -\inf_{z\in\mathbb{R}} r(z), \tag{2.56b}$$

with

$$r(z) = p(z) - zp^{t\prime}(y_0) - p(y_0) + y_0 p^{t\prime}(y_0). \tag{2.57}$$

Note that $r(z)$ is also a function of t and y_0, however we leave these arguments as implicit so as to keep the notation light. We now show that the minimum of $r(z)$ exists, which in turn implies that $\epsilon^t(y_0) \in \mathbb{R}$. To see this, first note that since $p(z)$ is convex then also $r(z)$ is convex. Let us study the subderivative of $r(z)$. By Lemma 2.9, $p(y)$ is linear for $y \leq y^{\text{MIN}}$ and for $y \geq y^{\text{MAX}}$. Thus, $r(z)$ is differentiable for all $z \in (-\infty, y^{\text{MIN}}) \cup (y^{\text{MAX}}, +\infty)$. For all $z \leq y^{\text{MIN}}$, it holds

$$r'(z) = p'(z) - p^{t\prime}(y_0) = -M - p^{t\prime}(y_0) \leq 0,$$

where the last equality follows by Lemma 2.9 and the inequality follows by Lemma 2.8. Analogously, for all $z \geq y^{\text{MAX}}$, it holds

$$r'(z) = p'(z) - p^{t'}(y_0) = -p^{t'}(y_0) \geq 0.$$

Thus $r(z)$ is non increasing for $z \leq y^{\text{MIN}}$ and non decreasing for $z \geq y^{\text{MAX}}$. Being the function convex (and thus continuous), there exists a (finite) minimum in the interval $[y^{\text{MIN}}, y^{\text{MAX}}]$. i.e.,

$$\min_{z \in \mathbb{R}} r(z) = \min_{z \in [y^{\text{MIN}}, y^{\text{MAX}}]} r(z).$$

Thus, $\epsilon^t(y_0) \in \mathbb{R}$ since the inf in (2.55) is finite.

Now we proceed to compute a vanishing overestimate of $\epsilon^t(y_0)$. Consider the sequence $\{p^{t'}(y_0)\}_{t \in \mathbb{N}}$ and fix $\beta > 0$. Define $\eta = \beta / |y^{\text{MIN}} - y^{\text{MAX}}| > 0$. By Lemma 2.11, there exists $N > 0$ such that $|p^{t'}(y_0) - p'(y_0)| < \eta$ for all $t \geq N$. To compute the overestimate of $\epsilon^t(y_0)$, we assume $p(z)$ is replaced with a convex, piece-wise linear surrogate function $p_{y_0}^{\text{AUX}}(z)$, defined as

$$p_{y_0}^{\text{AUX}}(z) = \max\Big\{ p(y^{\text{MIN}}) + M(y^{\text{MIN}} - y), \quad p(y_0) + p'(y_0)(y - y_0), \quad p(y^{\text{MAX}}) \Big\}.$$

The resulting function consists of three pieces. The left-most piece and the right-most piece are obtained by prolonging the two lateral linear pieces of $p(z)$ inside the interval $[y^{\text{MIN}}, y^{\text{MAX}}]$, while the central piece is the tangent line crossing $p(z)$ at y_0. By construction, this function satisfies $p_{y_0}^{\text{AUX}}(z) \leq p(z)$ for all $z \in \mathbb{R}$. Let us compute the break points, which we denote by y^L and y^R (see Figure 2.15). To compute y^L, we must intersect the first two pieces, i.e.

$$\underbrace{-My^L + p(y^{\text{MIN}}) + My^{\text{MIN}}}_{\text{left piece}} = \underbrace{p(y_0) + (y^L - y_0)p'(y_0)}_{\text{central piece}},$$

which results in

$$y^L = \frac{p(y^{\text{MIN}}) + My^{\text{MIN}} - p(y_0) + y_0 p'(y_0)}{p'(y_0) + M}. \tag{2.58}$$

Similarly, we can compute y^R, which is equal to

$$y^R = y_0 + \frac{p(y^{\text{MAX}}) - p(y_0)}{p'(y_0)}. \tag{2.59}$$

Notice that the value of $p_{y_0}^{\text{AUX}}(y^L)$ is equal to $p(y^{\text{MAX}})$.

Now, similarly to $r(z)$, let us define functions $r^{\text{AUX},t}(z)$ corresponding to $p^t(z)$ for

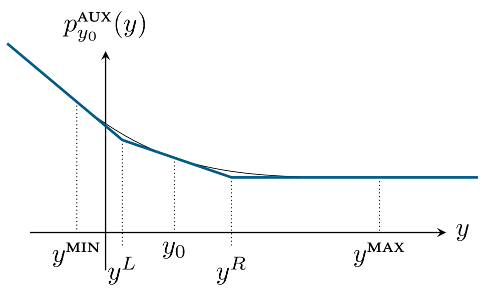


Figure 2.15: Graphical representation of the surrogate function $p^{\text{AUX}}_{y_0}(z)$ (in blue). The original primal function $p(z)$ is the black curve near the blue one.

all t. Then, we use them to compute the upper bound on $\epsilon^t(y_0)$, in a similar way as in (2.56a)–(2.56b). The functions $r^{\text{AUX},t}(z)$ are defined as

$$r^{\text{AUX},t}(z) = p^{\text{AUX}}_{y_0}(z) - zp^{t'}(y_0) - p(y_0) + y_0 p^{t'}(y_0), \qquad t \in \mathbb{N}. \tag{2.60}$$

As before, these functions also depend on y_0, which is omitted in the notation because it is fixed. It holds $r^{\text{AUX},t}(z) \leq r(z)$ (since $p^{\text{AUX}}_{y_0}(z) \leq p(z)$). Being $p^{\text{AUX}}_{y_0}(z)$ piece-wise linear with three pieces, then also $r^{\text{AUX},t}(z)$ is piece-wise linear with three pieces. Similarly to (2.56b), let us now define the overestimate of $\epsilon^t(y_0)$ as

$$\epsilon^{\text{AUX},t}(y_0) \triangleq - \inf_{z\in\mathbb{R}} r^{\text{AUX},t}(z) = - \min_{z\in[y^L,y^R]} r^{\text{AUX},t}(z),$$

where the equality holds since the function $r^{\text{AUX},t}(z)$ admits minimum in $[y^L, y^R]$ (by following the same reasoning used for $r(z)$). Since $r^{\text{AUX},t}(z) \leq r(z)$ for all z, the same holds for the minimum of such functions over $[y^L, y^R]$, from which we see that indeed $\epsilon^{\text{AUX},t}(y_0) \geq \epsilon^t(y_0)$. Since $r^{\text{AUX},t}(z)$ is linear in the interval $[y^L, y^R]$, the minimum is attained either at y^L or at y^R:

$$\epsilon^{\text{AUX},t}(y_0) = - \min \Big\{ r^{\text{AUX},t}(y^L), \;\; r^{\text{AUX},t}(y^R) \Big\}. \tag{2.61}$$

Let us compute the value of the function at y^L and y^R, i.e.

$$\begin{aligned} r^{\text{AUX},t}(y^L) &= p^{\text{AUX}}_{y_0}(y^L) - y^L p^{t'}(y_0) - p(y_0) + y_0 p^{t'}(y_0) \\ &= p(y_0) + p'(y_0)(y^L - y_0) - y^L p^{t'}(y_0) - p(y_0) + y_0 p^{t'}(y_0) \\ &= (p'(y_0) - p^{t'}(y_0))(y^L - y_0), \end{aligned}$$

and, similarly,

$$r^{\text{AUX},t}(y^R) = (p'(y_0) - p^{t'}(y_0))(y^R - y_0),$$

which always have opposite sign since $y^L \leq y_0 \leq y^R$. Thus, we can distinguish two cases. If $r^{\text{AUX},t}(y^L) \leq 0$, then the minimum in (2.61) is attained at $r^{\text{AUX},t}(y^L)$ and therefore

$$\begin{aligned}
\epsilon^{\text{AUX},t}(y_0) = -r^{\text{AUX},t}(y^L) &= \underbrace{-(p'(y_0) - p^{t\prime}(y_0))(y^L - y_0)}_{\geq 0} \\
&= \underbrace{|p'(y_0) - p^{t\prime}(y_0)|}_{\leq \eta \ \forall t \geq N} \underbrace{|y^L - y_0|}_{\leq |y^{\text{MIN}} - y^{\text{MAX}}|} \\
&\leq \eta |y^{\text{MIN}} - y^{\text{MAX}}|, \qquad \forall t \geq N.
\end{aligned}$$

Likewise, if $r^{\text{AUX},t}(y^R) \leq 0$, we obtain

$$\epsilon^{\text{AUX},t}(y_0) = -r^{\text{AUX},t}(y^R) = -(p'(y_0) - p^{t\prime}(y_0))(y^R - y_0) \leq \eta |y^{\text{MIN}} - y^{\text{MAX}}|, \quad \forall t \geq N.$$

In either cases, it holds

$$\epsilon^t(y_0) \leq \epsilon^{\text{AUX},t}(y_0) \leq \underbrace{\eta |y^{\text{MIN}} - y^{\text{MAX}}|}_{\beta}, \qquad \forall t \geq N.$$

which is independent of the chosen y_0. Thus we conclude

$$0 \leq \epsilon^t \leq \max_{y \in \mathbb{R}} \epsilon^t(y) \leq \max_{y \in \mathbb{R}} \epsilon^{\text{AUX},t}(y) \leq \beta, \qquad \forall t \geq N. \tag{2.62}$$

Since $\beta > 0$ is arbitrary, it follows that $\lim\limits_{t \to \infty} \epsilon^t = 0$. ■

Note that, in order for Proposition 2.5 to hold, Assumption 2.6 is important. Indeed, if Assumption 2.6 does not hold, it can be seen that in the previous proof that Lemma 2.11 could not be applied, and thus one could use the fact that $|p^{t\prime}(y_0) - p'(y_0)| < \eta$. As a consequence, (2.62) would not be valid and it would not be possible to conclude that $\lim_{t \to \infty} \epsilon^t = 0$.

Main Result

Let us now provide the main theoretical result for the convergence analysis of Algorithm 2. First, we formalize the assumption on the tie-break rule (cf. (2.52)).

Assumption 2.7 (Tie-break rule)**.** *There is a tie-break rule such that, at each iteration $t \geq 0$, each agent $i \in \{1, \ldots, N\}$ selects μ_i^t deterministically among all the Lagrange multipliers of Problem* (2.45)*.* △

This assumption is not restrictive in practice, since numerical solvers typically execute a set deterministic actions to solve Problem (2.45). As such, they yield always the same μ_i^t for a given y_i^t. The convergence theorem is reported next.

Theorem 2.3. *Let Assumptions 2.1, 2.2, 2.6, 2.3 and 2.7 hold. Moreover, let $\mu^\star$ be an optimal Lagrange multiplier of Problem* (2.1) *associated to the constraint $\sum_{i=1}^N g_i(x_i) \leq b$ and assume $M > \|\mu^\star\|_1$. Then, assuming the allocation vectors y_i^0 are initialized such that $\sum_{i=1}^N y_i^0 = b$, the sequences $\{x_i^t\}_{t\geq 0}$ and $\{y_i^t\}_{t\geq 0}$ generated by Algorithm 2 for all $i \in \{1,\dots,N\}$ are such that*

(i) the sequence $\{(y_1^t, \dots y_N^t)\}_{t\geq 0}$ converges to an optimal solution of Problem (2.2)*;*

(ii) $\lim_{t\to\infty} \sum_{i=1}^N f_i(x_i^t) = f^\star$, where $f^\star$ is the optimal cost of (2.1)*;*

(iii) every limit point of $\{(x_1^t, \dots x_N^t)\}_{t\geq 0}$ is an optimal solution of (2.1).

Proof. We will use a line of proof similar to the one of Theorem 2.1 but without block randomization. By Corollary 2.1, Problem (2.28) has the same optimal cost as Problem (2.1). Recall that $p_i^t(y_i)$ denotes the optimal cost of Problem (2.45) for all $i \in \{1,\dots,N\}$ and $t \geq 0$. Let us consider a subgradient method applied to Problem (2.28). Instead of the standard subgradient method, we replace the subgradients of $\tilde{p}_i\big(\{z_{ij}^t, z_{ji}^t\}_{j\in\mathcal{N}_i}\big)$ with a subgradient of $\tilde{p}_i^t\big(\{z_{ij}^t, z_{ji}^t\}_{j\in\mathcal{N}_i}\big) \triangleq p_i^t\Big[\sum_{j\in\mathcal{N}_i}(z_{ij}^t - z_{ji}^t)\Big]$ and therefore consider the update

$$z_{ij}^{t+1} = z_{ij}^t - \alpha^t \tilde{p}_i^{t\prime}\big(\{z_{ij}^t, z_{ji}^t\}_{j\in\mathcal{N}_i}\big) \qquad \forall (i,j) \in \mathcal{E}, \tag{2.63}$$

initialized at some $z^0 \in \mathbb{R}^{|\mathcal{E}|}$. As we will see in a moment, the update (2.63) is in fact an ϵ-subgradient method applied to Problem (2.28). By Lemma 2.2, the subderivatives of p_i^t at $\sum_{j\in\mathcal{N}_i}(z_{ij}^t - z_{ji}^t)$ are equal to

$$p_i^{t\prime}\Big[\sum_{j\in\mathcal{N}_i}(z_{ij}^t - z_{ji}^t)\Big] = -\mu_i^t,$$

where μ_i^t is a Lagrange multiplier of Problem (2.48) (with $y_i = \sum_{j\in\mathcal{N}_i}(z_{ij} - z_{ji})$) associated to the constraint $g_i(x_i) \leq y_i + \rho_i$. Following the same reasoning of Theorem 2.1, using the change of coordinates (2.26) we can show that the subderivatives of $\tilde{p}_i^t$ are equal to

$$\tilde{p}_i^{t\prime}\big(\{z_{ij}^t, z_{ji}^t\}_{j\in\mathcal{N}_i}\big) = \mu_j^t - \mu_i^t, \qquad \forall (i,j) \in \mathcal{E},$$

from which it follows that the update (2.63) can be rewritten as

$$z_{ij}^{t+1} = z_{ij}^t + \alpha^t(\mu_j^t - \mu_i^t) \qquad \forall (i,j) \in \mathcal{E}. \tag{2.64}$$

By Proposition 2.5, together with Assumption 2.7 and Lemma 2.11, we finally see that the update (2.64) is an ϵ-subgradient method applied to Problem (2.28), with $\epsilon^t = \max_i \epsilon_i^t$

going to 0 as t goes to ∞. Thus, by following the arguments of [11, Section 3.3] and by also using the boundedness of the subderivatives (Lemma 2.8) and Assumption 2.3, we conclude that the sequence $\{z^t\}_{t\geq 0}$ generated by (2.64) converges to an optimal solution $z^\star$ of Problem (2.28).

Let us rewrite the update (2.64) in terms of y by using the change of coordinate (2.26),

$$\begin{aligned} y_i^{t+1} &= \sum_{i=1}^{N} \Bigg[\sum_{j\in\mathcal{N}_i} (z_{ij}^t - z_{ji}^t) + b/N \Bigg] \\ &= y_i^t + \alpha^t \sum_{j\in\mathcal{N}_i^t} \big(\mu_i^t - \mu_j^t\big), \end{aligned} \tag{2.65}$$

where we also used the fact that the graph is undirected and that $\sum_{i=1}^N y_i^t = b$ for all $t \geq 0$ (by induction). Note that (2.65) is exactly the algorithm update (2.46). Thus, we conclude that the sequence $\{y^t\}_{t\geq 0}$ converges to $y^\star$, with components equal to

$$y_i^\star = \sum_{j\in\mathcal{N}_i} (z_{ij}^\star - z_{ji}^\star) + b/N, \qquad i = 1, \ldots, N.$$

Thus, point *(i)* follows by Corollary 2.1.

We have thus shown that the sequence $\{y^t\}_{t\geq 0}$ generated by Algorithm 2 converges to an optimal solution of Problem (2.2) even if we replaced the original cost functions $f_i(x_i)$ with their estimation $f_i^t(x_i)$ in Problem (2.45). Points *(ii)* and *(iii)* can be proven by following the arguments of Theorem 2.1 (in particular, here (2.42) holds by the uniform convergence of f_i^t to f_i, cf. Assumption 2.6). ■

Let us briefly comment on the previous result. Note that the conclusions are identical to those that we concluded in Section 2.3. However, we recall in the local problems (2.45) the true objective function f_i is replaced by an estimated version f_i^t that asymptotically approach the true one. Even though one can intuitively expect that the result of Theorem 2.1 is recovered, the main contribution here consists of the digression on the behavior of the primal functions and on the uniform convergence of dual functions.

2.6.4 Numerical Example

In this section, we show numerical computations to corroborate the theoretical results. To estimate the cost functions, we consider a method similar to the one in [109], without all the machinery to guarantee smoothness and strong convexity.

Formally, fix an agent i and consider $K \in \mathbb{N}$ samples $z_i^1, \ldots, z_i^K \in X_i$. To build the estimated function, we let the agent compute $\gamma_1, \ldots, \gamma_K \in \mathbb{R}^{n_i}$ by solving the feasibility problem

$$\begin{aligned} &\text{find } \gamma^1, \dots, \gamma^K \\ &\text{subj. to } f_i(z_i^h) + (z_i^\ell - z_i^h)^\top \gamma^h \leq f_i(z_i^\ell), \qquad h, \ell = 1, \dots, K,\ h \neq \ell \end{aligned} \tag{2.66}$$

The purpose of Problem (2.66) is to compute the slope of the linear pieces that build up the estimation of f_i and has K variables and $K(K-1)$ convexity constraints. With the solutions $(\gamma^1, \dots, \gamma^K)$ at hand, the oracle returns the estimated functions in the following form

$$f_i^K(x) = \max_{k \in \{1,\dots,K\}} \{ f_i(z_i^k) + (x - z_i^k)^\top \gamma^k \}, \qquad i = 1, \dots, N. \tag{2.67}$$

Note that $f_i^K(\cdot)$ is a piece-wise linear function. Moreover, f_i^K is the pointwise maximum of a finite number of affine functions, its epigraph is a non-empty polyhedron, and hence f_i^K is convex, closed and proper (see [114, Theorem 1]).

At each iteration, each agent i collects a new sample of the domain and solves (2.66). As the size of Problem (2.66) grows at each iteration, we proceed as follows. Initially the samples are independent and identically distributed in the whole domain. As more points are added to the model, we start to reduce the space where sampling new points into balls centered in the current approximated solution. In fact, restricting the sampling space, force the model to refine the surrogate function in those area containing the approximated solution. We experimentally tested that, thanks to the latter expedient, it is possible to reduce the number of samples to keep in the memory. In fact, if the cost function becomes interesting only near the minimum, from a certain point on, estimating the part of the function far from the minimum becomes irrelevant, and all samples far from the minimum can be canceled. By following this intuition, together with the fact that the sample space is shrinking more and more around the potential solution, the samples collected further away in terms of time can be removed from the model. This relieves the function estimation, whose complexity is dependent on the number of points used.

We consider a network of $N = 10$ agents in a 3-dimensional domain. We generate a random Erdős-Rényi graph with edge probability 0.2. We consider quadratic local functions f_i and linear local constraints g_i. Figure 2.16 shows the evolution of the algorithm in terms of the cost error $|f^\star - \sum_{i=1}^N f_i(x_i^t)|$. Despite the use of surrogate cost functions, the objective value converges to the optimal cost of the original problem with true cost function, as we expected from the theoretical results.

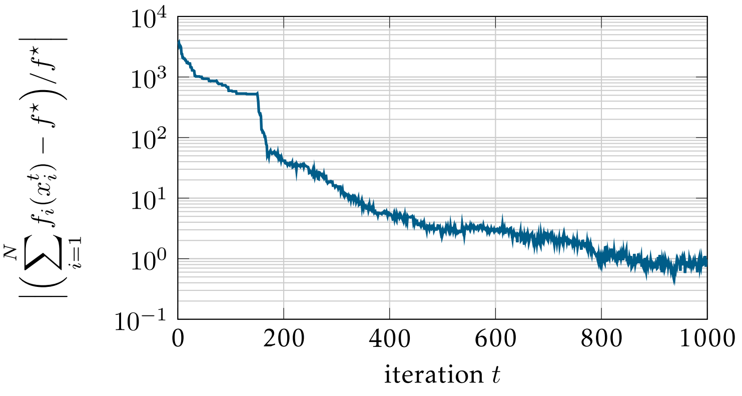


Figure 2.16: Evolution of the cost error for the numerical example with unknown cost functions.

Chapter 3

Distributed Primal Decomposition for Mixed-integer Optimization

In this chapter, we consider Mixed-Integer Linear Programs (MILPs) with a constraint-coupled structure. By relying on the results of Chapter 2, we design a distributed solution procedure that allows agents in the network to compute a feasible (possibly suboptimal) solution to the original MILP. We first introduce the distributed optimization set-up together with some preliminaries. Then, we present our Distributed Primal Decomposition for MILPs provide an asymptotic and finite-time analysis. We then introduce another distributed algorithm for MILPs based on the so-called Benders Decomposition and discuss its convergence properties. Finally, we extend the framework to general nonconvex programs. The results of this chapter are based on [23–26]

3.1 Literature Review

MILPs arise in several control contexts, such as distributed optimal control problems where a large set of nonlinear control systems must cooperatively solve a common control task and their states, outputs and/or inputs are coupled through a coupling constraint. Here, the constraint-coupled structure results directly from the problem formulation, and the integer constraints stem from the MILP approximation of the original optimal control problem [7]. In [61] a Lagrange relaxation approach is used to decompose MILPs arising in demand response control in smart grids. In [112] a heuristic for embedded mixed-integer programming is studied to obtain approximate solutions. First attempts to obtain a distributed approximate solution for MILPs are [51, 64]. Recently, a distributed algorithm, based on cutting-planes, has been investigated in [117] to solve a different class of MILPs with shared decision variable. A randomized approach for robust mixed-integer programming has been developed in [32] for the same set-up. Although these methods could be applied to our set-up, each agent would compute the entire solution vector, thus incurring in scalability and privacy issues. A pioneering work on fast, master-based parallel algorithms to find approximate solutions of our problem set-up is [121]. Here the key idea is to tighten the coupling constraint and then apply a dual decomposition method to get mixed-integer points violating the restricted coupling constraint but not the original one. In [46], an improved iterative tightening procedure has been considered to obtain enhanced performance guarantees. A fully distributed implementation of the latter methodology is proposed in [45].

In distributed nonlinear optimal control, the optimization problem to be solved is a general nonconvex problem. Let us first review works for the case in which nonconvexity is in the cost function. In [14] a distributed stochastic gradient method with gossip communication is analyzed. A distributed Frank-Wolfe algorithm with convergence rate analysis is provided in [122], while [40] and [102] propose distributed gradient tracking algorithms, based on successive convex approximations, for undirected and directed networks (respectively). A more general set-up occurs when nonconvexity is also in the feasible set. In [128], a distributed dual subgradient method to obtain a an approximate solution is considered, whereas [119] proposes a parallel sequential quadratic programming algorithm when nonconvexity is in the feasible set only. Most works on distributed nonconvex optimization deal with problems that are coupled in the cost function, while only few references (e.g. [119]) consider the set-up of distributed control, in which the coupling among the systems is in the constraints.

3.2 Distributed MILP Set-up and Preliminaries

Let us start by introducing the Mixed-Integer Linear Programming set-up, together with some preliminaries that act as building blocks for the development of our methodology.

3.2.1 Constraint-Coupled MILP

Let us consider a network of N agents aiming to solve the MILP

$$\begin{aligned} \min_{x_1,\dots,x_N} \quad & \sum_{i=1}^{N} c_i^\top x_i \\ \text{subj. to} \quad & \sum_{i=1}^{N} A_i x_i \le b \\ & x_i \in X_i^{\text{MILP}}, \qquad i = 1,\dots,N, \end{aligned} \tag{3.1}$$

where, for all $i \in \{1,\dots,N\}$, the decision vector x_i has $p_i + q_i$ components (thus $c_i \in \mathbb{R}^{p_i+q_i}$) with $p_i, q_i \in \mathbb{N}$ and the local constraint set is of the form

$$X_i^{\text{MILP}} = P_i \cap (\mathbb{Z}^{p_i} \times \mathbb{R}^{q_i}),$$

for some nonempty compact polyhedron $P_i \subset \mathbb{R}^{p_i+q_i}$. In Figure 3.1, we show an example of the local mixed-integer set X_i^{MILP}.

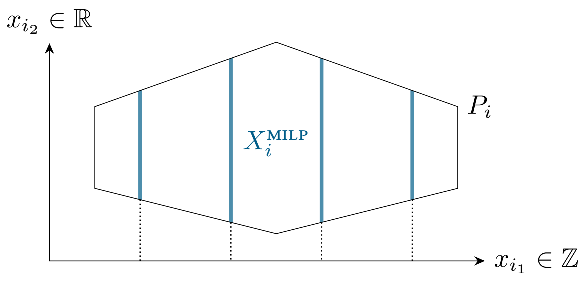


Figure 3.1: Two-dimensional example of the mixed-integers set $X_i^{\text{MILP}} = P_i \cap (\mathbb{Z} \times \mathbb{R})$. The black hexagon is the polyhedron P_i and the dotted lines correspond to integers along the horizontal axis. The resulting feasible set is the union of the blue vertical lines.

The decision variables are intertwined by S linear *coupling* constraints, described by the matrices $A_i \in \mathbb{R}^{S\times(p_i+q_i)}$ and the vector $b \in \mathbb{R}^S$. We assume that Problem (3.1) is feasible and denote by $(x_1^\star, \dots, x_N^\star)$ an optimal solution with cost $J^{\text{MILP}} = \sum_{i=1}^N c_i^\top x_i^\star$. In many control applications, the number of decision variables is typically much larger than the number of coupling constraints. Therefore, we concentrate on *large-scale* instances of Problem (3.1) satisfying $N \gg S$.

Since we consider a distributed context, each agent i is assumed to have a *partial knowledge* of Problem (3.1) i.e. it knows only its local data X_i^{MILP}, c_i, A_i and b. The goal for each agent is to compute its portion $x_i^\star$ of an optimal solution of (3.1), by means of neighboring communication and local computation. Agents communicate according to a connected and undirected graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1, \dots, N\}$ is the set of vertices and $\mathcal{E} \subseteq V \times V$ is the set of edges. If $(i,j) \in \mathcal{E}$, then also $(j,i) \in \mathcal{E}$ and nodes i and j can exchange information. The neighbor set of agent i is $\mathcal{N}_i = \{j \in V \mid (i,j) \in \mathcal{E}\}$. This assumption can be readily extended to randomized time-varying graphs (cf. Section 2.3.1), however this is not the main focus of this chapter and we prefer to keep the discussion technically simpler to present the results more clearly.

MILP (3.1) has a constraint-coupled structure similar to the one of Problem (2.1) (with linear cost and constraints). However, the mixed-integer constraints considered here introduce significant additional challenges that do not allow for the methodology of the previous chapter. Indeed, Problem (3.1) is not convex and we cannot apply the duality-based arguments of Section 2.2.3 (at least directly). To solve MILP (3.1), one could employ enumeration schemes, such as branch-and-bound or cutting-plane techniques, however this would not exploit its separable structure and would result into a computationally intensive algorithm. In the next subsection, we recall an approximate version of the problem that preserves its structure while allowing for the application of decomposition techniques.

3.2.2 Linear Programming Approximation of the Target MILP

Following [45, 46, 121], let us consider a modified version of Problem (3.1) where the right-hand side of the coupling constraint is restricted by a vector $\sigma \geq \mathbf{0}$ and each mixed-integer set X_i^{MILP} is replaced by its convex hull, denoted by $\text{conv}(X_i^{\text{MILP}})$. The following *convex* problem is obtained

$$\begin{aligned} \min_{z_1,\ldots,z_N} \quad & \sum_{i=1}^{N} c_i^\top z_i \\ \text{subj. to} \quad & \sum_{i=1}^{N} A_i z_i \leq b - \sigma \\ & z_i \in \text{conv}(X_i^{\text{MILP}}), \qquad i = 1, \ldots, N, \end{aligned} \tag{3.2}$$

where $z_i \in \mathbb{R}^{p_i+q_i}$ for all $i \in \{1, \ldots, N\}$. We introduced the symbol z_i to clearly distinguish mixed-integer variables $x_i \in X_i^{\text{MILP}}$ from their continuous counterparts $z_i \in \text{conv}(X_i^{\text{MILP}})$. An example of the convexified mixed-integer set is shown in Figure 3.2. Problem (3.2) is a Linear Program (LP) since the sets $\text{conv}(X_i^{\text{MILP}})$ are bounded polyhedra [74].

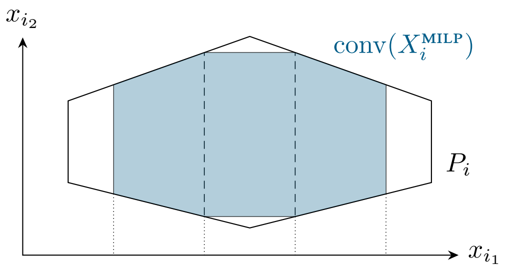


Figure 3.2: Convex hull (shaded area) of the set X_i^{MILP} from the example in Figure 3.1. The dashed and dotted lines correspond to integers values on the horizontal axis. Note how $\text{conv}(X_i^{\text{MILP}}) \subset P_i$ is again a polyhedron but there are points in P_i that are not in $\text{conv}(X_i^{\text{MILP}})$.

Remark 3.1. In order to solve MILPs, several algorithms resort to the convex hull of the constraint set. However, we point out that Problem (3.2) considers only the convex hulls of the local constraint sets X_i. The constraint set of Problem (3.2), is *not* the convex hull of the constraint set of MILP (3.1). Indeed, denoting $\mathcal{X}$ and $\mathcal{X}_{\text{LP}}$ the constraint sets of MILP (3.1) and LP (3.2) (with $\sigma = 0$), in general it holds $\mathcal{X} \subseteq \text{conv}(\mathcal{X}) \subseteq \mathcal{X}_{\text{LP}}$. This fact is tightly linked to the properties of the Lagrangian dual problem of MILP (3.1) (see also Appendix A.3.2). △

The main point in solving the (convex) Problem (3.2) in place of the (nonconvex) original MILP (3.1) is to reconstruct a feasible solution of (3.1) starting from a solution of (3.2). During this process, it is often the case that satisfaction of the coupling constraints is lost. The restriction σ essentially creates a margin for violations and makes sure that the reconstructed mixed-integer solution satisfies the (unrestricted) original coupling constraint. Note that, if σ was zero, Problem (3.2) would be a relaxation of Problem (3.1) and would be trivially feasible. However, as done in the related approaches [45, 46, 121], at some point we will fix $\sigma > 0$ and we will need to explicitly assume feasibility of Problem (3.2).

All the restriction-based approaches [45, 46, 121] are based on the following key result, which is also the basis for our development.

Proposition 3.1. *Let Problem* (3.2) *be feasible and let* $(\bar{z}_1, \dots, \bar{z}_N)$ *be any vertex of its feasible set. Then, there exists an index set* $I_{\mathbb{Z}} \subseteq \{1, \dots, N\}$*, with cardinality* $|I_{\mathbb{Z}}| \geq N - S$*, such that* $\bar{z}_i \in X_i^{\text{MILP}}$ *for all* $i \in I_{\mathbb{Z}}$*.* △

An early proof is given in the reference [8], while a more recent proof can be found in [121], which however refers to the solutions recovered from the Lagrangian dual of Problem (3.2) (under the assumption of unique dual solution). For completeness, we provide a self-contained proof in Section 3.6.1.

The powerful result in Proposition 3.1 is a consequence of the more general Shapley-Folkman lemma on the Minkowski addition of sets. For any vertex $(z_1, \dots, z_N)$ of Problem (3.2), it guarantees that a large amount of blocks z_i are mixed integer (recall that $N \gg S$). We will pursue the same idea of the inspiring restriction-based approaches, that is, to compute an optimal vertex of Problem (3.2) and to change only those blocks that are not *already* mixed integer. However, the most important novelty introduced by our approach is the utilization of primal decomposition to solve Problem (3.2) instead of the dual decomposition techniques employed by [45, 46, 121]. Notably, the different decomposition scheme results in a large improvement of the solution performance and relaxed problem assumptions if compared to competing duality-based approaches.[1]

Before getting into the details of the proposed approach, let us recall how the primal decomposition approach applies to Problem (3.2). In particular, here we consider the "vanilla" formulation of primal decomposition without applying the relaxation method discussed in Section 2.2.3. The master problem reads

$$
\begin{aligned}
\min_{y_1, \dots, y_N} \quad & \sum_{i=1}^{N} p_i(y_i) \\
\text{subj. to} \quad & \sum_{i=1}^{N} y_i = b - \sigma \\
& y_i \in Y_i, \qquad i = 1, \dots, N,
\end{aligned}
\tag{3.3}
$$

[1]Indeed, the larger is σ, the stronger is the assumption that Problem (3.2) is feasible. As discussed in Section 3.3.3, the primal decomposition approach allows for an amount of restriction that is no worse than [121]. In fact, numerical experiments highlight extremely lower restriction magnitudes, which, as a byproduct, result also in much less suboptimality of the computed solutions.

where, for each $i \in \{1, \dots, N\}$, $p_i : Y_i \to \mathbb{R}$ is the optimal cost of the i-th subproblem

$$\begin{aligned} p_i(y_i) = \; & \min_{z_i} \; c_i^\top z_i \\ & \text{subj. to } \; A_i z_i \leq y_i \\ & \qquad\qquad z_i \in \text{conv}(X_i^{\text{MILP}}), \end{aligned} \tag{3.4}$$

and for all $i \in \{1, \dots, N\}$ the set $Y_i \subseteq \mathbb{R}^S$ is the set of y_i for which Problem (3.4) is feasible, i.e. $Y_i = \{y_i \mid \exists z_i \in \text{conv}(X_i^{\text{MILP}}) \text{ with } A_i z_i \leq y_i\}$. Similarly to LP (3.2), Problem (3.4) is a linear program. For convenience, let us recall the equivalence between LP (3.2) and Problems (3.3)–(3.4) in the following lemma.

Lemma 3.1 (Lemma 2.1). *Let Problem* (3.2) *be feasible. Then,*

(i) the optimal costs of Problems (3.2) *and* (3.3) *are equal;*

(ii) if $(y_1^\star, \dots, y_N^\star)$ *is an optimal solution of* (3.3) *and, for all i,* $z_i^\star$ *is an optimal solution of* (3.4) *with* $y_i = y_i^\star$*, then* $(z_1^\star, \dots, z_N^\star)$ *is an optimal solution of* (3.2). $\triangle$

3.3 Distributed Primal Decomposition for MILPs

In this section we propose a novel distributed algorithm to compute a feasible solution (possibly suboptimal) of MILP (3.1). We begin by describing the distributed algorithm. Then, we provide an analysis of the proposed approach and show numerical computations on randomly generated MILPs. The results of this section are based on [23, 24].

3.3.1 Distributed Algorithm Description

Let us introduce our Distributed Primal Decomposition for MILPs. Informally, the algorithm is derived from the Distributed Primal Decomposition algorithm for convex problems (cf. Section 2.2.5) applied to Problem (3.2), with an additional final step that adapts the computed solution to satisfy the mixed-integer constraints.

Each agent i maintains a local allocation estimate $y_i^t \in \mathbb{R}^S$, initialized such that $\sum_{i=1}^N y_i^0 = b - \sigma$. At each iteration $t \geq 0$, the vector y_i^t is updated according to (2.12)–(2.13). After $T_f > 0$ iterations, the agent computes a tentative mixed-integer solution based on the last computed allocation $y_i^{T_f}$ with (3.7), where the notation lex-min denotes that v_i, ξ_i and x_i are minimized in a lexicographic order [86]. This formulation of the mixed-integer recovery procedure is convenient as it allows for a concise statement of the algorithm. In Section 3.3.2 we discuss in more detail the meaning of Problem (3.7) and an operative way to solve it. Algorithm 3 summarizes the steps from the perspective of agent i.

```
Algorithm 3 Distributed Primal Decomposition for MILPs
  Initialization: set $T_f > 0$ and $y_i^0$ such that $\sum_{i=1}^N y_i^0 = b - \sigma$
  Repeat for $t = 0, 1, \ldots, T_f - 1$
    Compute $\mu_i^t$ as a Lagrange multiplier of
        $\min_{z_i, \rho_i} \; c_i^\top z_i + M\rho_i$
        $\text{subj. to} \; \mu_i : A_i z_i \le y_i^t + \rho_i \mathbf{1}$
        $z_i \in \text{conv}(X_i^{\text{MILP}}), \; \rho_i \ge 0$          (3.5)
    Receive $\mu_j^t$ from $j \in \mathcal{N}_i$ and update
        $y_i^{t+1} = y_i^t + \alpha^t \sum_{j \in \mathcal{N}_i} (\mu_i^t - \mu_j^t)$          (3.6)
  Return $x_i^{T_f}$ as optimal solution of
        $\text{lex-min}_{v_i, \xi_i, x_i} \; v_i$
        $\text{subj. to} \; c_i^\top x_i \le \xi_i$
        $A_i x_i \le y_i^{T_f} + v_i \mathbf{1}$          (3.7)
        $x_i \in X_i^{\text{MILP}}, \; v_i \ge 0.$
```

A sensible choice for y_i^0 is $y_i^0 = (b - \sigma)/N$. As shown in Chapter 2 (cf. Proposition 2.1), the local allocation vectors $\{y_1^t, \ldots, y_N^t\}_{t \ge 0}$ converge asymptotically to an optimal solution $(y_1^\star, \ldots, y_N^\star)$ of Problem (3.3). Moreover, owing to Proposition 3.1, the asymptotic solution $z_i^\star$ of Problem (2.12) is already mixed integer for at least $N - S$ agents. As for the remaining (at most S) agents for which $z_i^\star \notin X_i^{\text{MILP}}$, a recovery procedure is needed to guarantee that they also have have a mixed-integer solution. This is done via step (3.7). Since in practice we can only allow for a finite number of iterations (denoted with T_f), in general the final allocation $y_i^{T_f}$ differs from $y_i^\star$. Thus, we let the recovery step (3.7) be performed by *all* the agents (not only those agents for which $z_i^\star \notin X_i^{\text{MILP}}$). In the next subsections, we provide an asymptotic and finite-time analysis of Algorithm 3.

Remark 3.2 (Initial coordination)**.** In order to run Algorithm 3, some initial coordination among the agents is required. Specifically, the agents first need to compute the restriction σ as specified in Section 3.3.3. Then, they must agree on a valid value of the parameter M as specified in Section 2.3.4. Note that both of these operations can be performed in a completely distributed way (see the respective sections). $\triangle$

Remark 3.3. From an implementation point of view, an explicit description of conv(X_i^{MILP}) in terms of inequalities might not be available and agents may not be able to compute a Lagrange multiplier of Problem (3.5). Nevertheless, an estimate of μ_i^t can still be obtained by either using column generation techniques to approximate $\text{conv}(X_i^{\text{MILP}})$ (see e.g. [120]) or by *locally* running a dual subgradient method on (2.12), which involves the solution of (small) local MILPs without ever computing the inequalities corresponding to $\text{conv}(X_i^{\text{MILP}})$. Indeed, a subgradient of the dual function of (3.5) at a given $\bar{\mu}_i$ can be computed as $A_i \bar{x}_i - y_i^t$, with $\bar{x}_i$ an optimal solution of

$$\min_{x_i \in X_i^{\text{MILP}}} (c_i^\top + \bar{\mu}_i^\top A_i) x_i = \min_{z_i \in \text{conv}(X_i^{\text{MILP}})} (c_i^\top + \bar{\mu}_i^\top A_i) z_i,$$

where equality follows by linearity of the cost. See also Remark 3.8 for an alternative local procedure based on cutting planes. $\triangle$

3.3.2 Discussion on Mixed-Integer Solution Recovery

In this subsection, which is of prominent importance for our analysis, we describe in more detail the approach behind Problem (3.7) and how it allows agents to recover a "good" solution of MILP (3.1). We first describe the procedure at steady state and then show how we cope with the finite number of iterations T_f.

Let $(z_1^\star, \dots, z_N^\star)$ be an optimal solution of the LP (3.2) and let $(y_1^\star, \dots, y_N^\star)$ be a corresponding allocation of the master problem (3.3) computed asymptotically by Algorithm 3, which satisfies

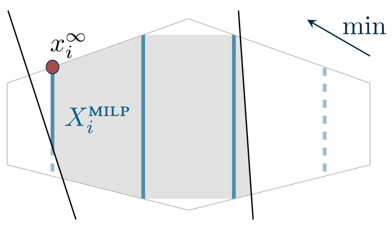
$$A_i z_i^\star \le y_i^\star, \qquad \forall i \in \{1, \dots, N\}.$$

A straight approach to obtain a mixed-integer solution from the allocation $y_i^\star$ would be to solve for all i the optimization problem

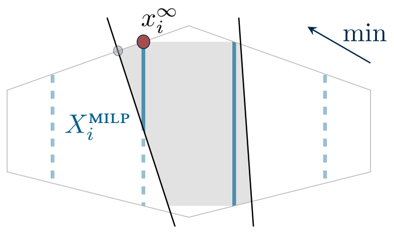
$$\begin{aligned} \min_{x_i} \; & c_i^\top x_i \\ \text{subj. to } \; & A_i x_i \le y_i^\star \\ & x_i \in X_i^{\text{MILP}}. \end{aligned} \tag{3.8}$$

Problem (3.8) is a (small) local MILP that should be compared to the convex subproblem (3.4). Depending on the allocation constraint $A_i x_i \le y_i^\star$, Problem (3.8) may be feasible or not. In view of Proposition 3.1, at least $N - S$ blocks $z_i^\star$ of optimal solution of the LP (3.2) are already mixed-integer and thus optimal for the local MILPs (3.8). Recall that $N \gg S$, thus *the majority* of subproblems (3.8) would admit optimal solution, while the total number of (possibly) infeasible instances would be *at most S*. If (3.8) is infeasible for some agent i, we let the agent relax the allocation constraint and find a mixed-integer vector with *minimal* violation. In Figure 3.3, we provide a pictorial representation of this procedure.

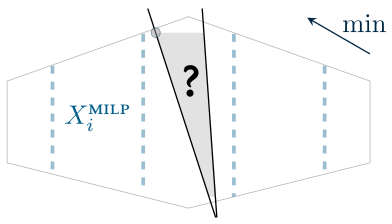
The procedure just outlined is entirely encoded in the lex-min minimization (3.7).



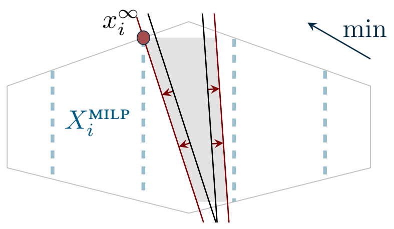
(a) Agent i is one of the agents for which $x_i^\infty = z_i^\star \in X_i^{\text{MILP}}$. Thus, Problem (3.8) is feasible and admits an optimal solution (red dot).



(b) In this case the solution on the convex hull $z_i^\star$ is not mixed integer (transparent gray dot), however Problem (3.8) is still feasible and admits an optimal solution (red dot).



(c) The solution on the convex hull $z_i^\star$ is not mixed integer (transparent gray dot) and there are no mixed-integer solutions. In this case Problem (3.8) is infeasible.



(d) If scenario (c) occurs, the constraint $A_i z_i \leq y_i^\star$ is enlarged just enough by $v_i^\infty \mathbf{1}$ to include the closest mixed-integer vector (red dot).

Figure 3.3: Illustration of the three possible scenarios for Problem (3.8). The local set is $X_i^{\text{MILP}} = P_i \cap (\mathbb{Z} \times \mathbb{R})$, where P_i is the gray hexagon and X_i^{MILP} is the union of the blue vertical stripes (see Figure 3.1). We assume the allocation constraint $A_i z_i \leq y_i^\star$ has two components, which are depicted with the black lines. The shaded gray area is the feasible part for the allocation constraint when intersected with $\text{conv}(X_i^{\text{MILP}})$ (see Figure 3.2). The resulting feasible set of Problem (3.8) (if not empty) is denoted with solid blue vertical lines.

To see this, let us now show how to operatively solve (3.7) with the asymptotic solution $y_i^\star$ in place of $y_i^{T_f}$. First, the needed violation of $A_i x_i \leq y_i^\star$ is determined by computing v_i^∞ as the optimal cost of

$$\begin{aligned} \min_{v_i, x_i} \; & v_i \\ \text{subj. to } \; & A_i x_i \leq y_i^\star + v_i \mathbf{1} \\ & x_i \in X_i^{\text{MILP}}, \;\; v_i \geq 0. \end{aligned} \tag{3.9}$$

Then, the value of v_i is fixed to v_i^∞ and x_i^∞ is computed as the optimal solution of

$$\begin{aligned} \min_{x_i} \; & c_i^\top x_i \\ \text{subj. to } \; & A_i x_i \leq y_i^\star + v_i^\infty \mathbf{1} \\ & x_i \in X_i^{\text{MILP}}. \end{aligned} \tag{3.10}$$

Compared to Problem (3.8), the sequence of problems (3.9)–(3.10) is also able to handle infeasible instances. If Problem (3.8) is feasible, then $v_i^\infty = 0$ and the entire procedure is equivalent to solving (3.8) directly. Instead, if Problem (3.8) is infeasible, a violation $v_i^\infty \mathbf{1}$ of the allocation constraint is permitted. We point out that, due to the violations, the aggregate mixed-integer solution $(x_1^\infty, \ldots, x_N^\infty)$ may exceed the total available resource of Problem (3.2), namely $b - \sigma$ (which we recall to be a restricted version of the original one). In the next subsection we show how to design the restriction σ to ensure that the original constraint $\sum_{i=1}^N A_i x_i^\infty \leq b$ is *never* exceeded.

Now, let us intuitively discuss how this approach can be adapted to cope with the finite number of iterations (a detailed analysis of this approach is given in Section 3.3.5). The local allocation y_i^t can be thought of being composed of two terms, i.e. $y_i^t = y_i^\star + \Delta_i^t \mathbf{1}$, where Δ_i^t vanishes. For this reason, when agent i solves Problem (3.7) (or the equivalent problems (3.9)–(3.10)), we should expect that the needed violation tends to the asymptotic one, be it zero or nonzero. That is, the solution of Problem (3.9) at a given iteration t satisfies $v_i^t \leq v_i^\infty + \Delta_i^t$. Hence, step (3.7) must be performed by *all* the agents. By employing an additional (small) restriction, it is possible to guarantee that – after a sufficiently large number of iterations – the total violation is embedded into the restriction (see Section 3.3.5).

3.3.3 Design of the Coupling Constraint Restriction

In this subsection, we discuss a design procedure for the restriction σ. As before, we concentrate on the "asymptotic" case, while the extension to finite time is given in Section 3.3.5. As already discussed, the purpose of the restriction σ is to compensate for possible violations caused by Problem (3.7). However, it is desirable to have σ as small as possible. Indeed, the larger is σ, the higher is the optimal cost of the LP (3.2), which in turn deteriorates the cost of the reconstructed mixed-integer solution (in the limit, if σ is too large the LP (3.2) would become infeasible and the entire approach would not be applicable at all).

We now propose a method to find a small *a-priori* restriction to guarantee feasibility of the computed solution. Intuitively, the restriction must take into account the worst-case violation due to the mismatch between the reconstructed solution $(x_1^\infty, \ldots, x_N^\infty)$ and the LP solution $(z_1^\star, \ldots, z_N^\star)$. Such worst case occurs when all the agents for which $z_i^\star \notin X_i^{\text{MILP}}$ have infeasible instances of (3.8), leading to a positive violation $v_i^\infty > 0$. Thus, we define the a-priori restriction $\sigma^\infty \in \mathbb{R}^S$ as

$$\sigma^\infty = S \cdot \max_{i \in \{1,\ldots,N\}} \sigma_i^{\text{LOC}}, \tag{3.11}$$

where the term S is due to Proposition 3.1, $\sigma_i^{\text{LOC}} \in \mathbb{R}^S$ is the worst-case violation of agent i (to be defined soon) and $\max$ is intended component wise (we stick to this convention from now on also for the $\min$ operator). Let us quantify σ_i^{LOC}. Recall that the needed violation is computed through Problem (3.10). Then, we essentially need to characterize the mismatch between $y_i^\star$ (allocation of the convex problem) and $A_i x_i$ for any feasible $x_i \in X_i^{\text{MILP}}$ (mixed-integer vector). Since $\text{conv}(X_i^{\text{MILP}})$ is bounded, we define a lower-bound vector, denoted by $\ell_i \in \mathbb{R}^S$, for any *admissible* local allocation y_i,

$$\ell_i \triangleq \min_{x_i \in \text{conv}(X_i^{\text{MILP}})} A_i x_i = \min_{x_i \in X_i^{\text{MILP}}} A_i x_i, \tag{3.12}$$

where the equality follows by linearity of the cost. By construction, it holds $\ell_i \leq A_i z_i^\star \leq y_i^\star$. Then, the worst-case violation that may occur is $v_i^{\text{MAX}} \mathbf{1}$, where v_i^{MAX} is the optimal cost of

$$\begin{aligned} \min_{v_i, x_i} \;\; & v_i \\ \text{subj. to } \;\; & A_i x_i \leq \ell_i + v_i \mathbf{1} \\ & x_i \in X_i^{\text{MILP}}, \;\; v_i \geq 0. \end{aligned} \tag{3.13}$$

Note that Problem (3.13) allows each agent i to find the "first" feasible vector, i.e. with minimal resource usage. In order to reduce possible conservativeness of the violation, which can occur when $v_i^{\text{MAX}} > \max_{x_i \in X_i^{\text{MILP}}} [A_i x_i - \ell_i]_s$ for some component s of the coupling constraint, the computation of σ_i^{LOC} can be replaced by the saturated version

$$\sigma_i^{\text{LOC}} = \min \Big\{ v_i^{\text{MAX}} \mathbf{1}, \; \max_{x_i \in X_i^{\text{MILP}}} (A_i x_i - \ell_i) \Big\}.$$

In numerical computations, we have found that usually $v_i^{\text{MAX}} \ll \max_{x_i \in X_i^{\text{MILP}}} [A_i x_i - \ell_i]_s$. In such a case, this last step is not necessary and one could directly define $\sigma_i^{\text{LOC}} = v_i^{\text{MAX}} \mathbf{1}$.

We point out that the computation of σ^∞ must be performed in the initialization phase, which can be also carried out in a fully distributed way by using a max-consensus algorithm. In Figure 3.4, we illustrate an example of the restriction.

Remark 3.4. In [121], an alternative approach based on dual decomposition is explored to compute a feasible solution for MILP (3.1). The restriction proposed in [121] is

$$\sigma^{\text{DD}} = S \cdot \max_{i \in \{1, \ldots, N\}} \max_{x_i \in X_i^{\text{MILP}}} (A_i x_i - \ell_i). \tag{3.14}$$

The i-th term $\max\limits_{x_i \in X_i^{\text{MILP}}} (A_i x_i - \ell_i)$ may be overly conservative (especially for large sets X_i^{MILP}) and in our approach it is replaced with σ_i^{LOC}, which is the resource utilization of the feasible vector *closest* to ℓ_i and, intuitively, does not grow with the size of X_i^{MILP}. Independently of the problem at hand, it can be easily seen that $\sigma^\infty \leq \sigma^{\text{DD}}$. $\triangle$

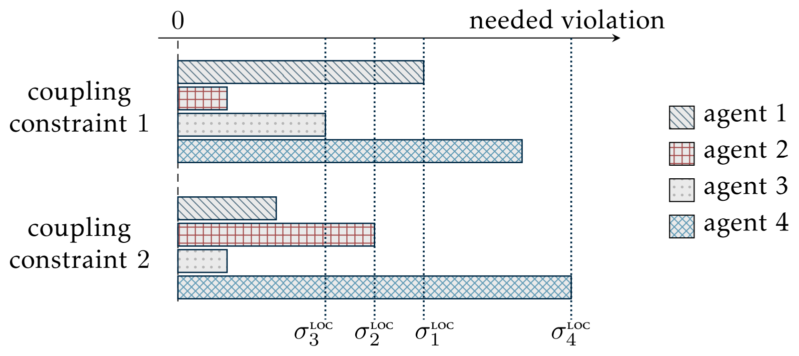


Figure 3.4: Pictorial representation of the procedure to compute the restriction for $N = 4$ agents and $S = 2$ coupling constraints. Each bar represents the quantity $[A_i x_i - \ell_i]_s$ at an optimal solution of (3.13), which is the violation associated to the first feasible vector with minimal resource usage. The four dotted lines represent v_i^{MAX} for $i = 1, \ldots, 4$, where we also assumed $\sigma_i^{\text{LOC}} = v_i^{\text{MAX}} \mathbf{1}$.

3.3.4 Asymptotic Analysis

Let us now provide a theoretical analysis of Algorithm 3. We start in this section by providing asymptotic results, while in the next section we discuss finite-time results. We will work under the following simplifying assumption.

Assumption 3.1. *For a given $\sigma \geq 0$, the optimal solution of Problem* (3.2) *is unique.* △

This assumption ensures that the optimal solution of (3.2) is a vertex (hence Proposition 3.1 applies) and can be guaranteed by simply adding a small, random perturbation to the cost vectors c_i. Notice that Assumption 3.1 is also needed in dual decomposition approaches such as [45, 46, 121], however, dual decomposition requires also uniqueness of the dual optimal solution of Problem (3.2), while our approach is less demanding since this is not necessary.

In this section, we proceed under the assumption that $\sigma = \sigma^\infty$ as in (3.11) and that the algorithm is executed until convergence to $(y_1^\star, \ldots, y_N^\star)$, i.e. an optimal solution of Problem (3.3) (recall that steps (2.12)–(2.13) implement the distributed algorithm in Section 2.2.5 applied to Problem (3.2), so that Proposition 2.1 (ii) applies). The next theorem shows that the asymptotic solution computed by Algorithm 3 is feasible for the target MILP (3.1).

Theorem 3.1 (Feasibility). *Let $\sigma = \sigma^\infty$ as in* (3.11), *and let Problem* (3.2) *be feasible and satisfy Assumption 3.1. Let $(y_1^\star, \ldots, y_N^\star)$ be an optimal solution of Problem* (3.3). *Then, the vector $(x_1^\infty, \ldots, x_N^\infty)$, with each x_i^∞ optimal solution of* (3.7) *with $y_i^t = y_i^\star$, is feasible for MILP* (3.1), *i.e. $x_i^\infty \in X_i^{\text{MILP}}$ for all $i \in \{1, \ldots, N\}$ and $\sum_{i=1}^N A_i x_i^\infty \leq b$.* △

The proof is provided in Section 3.6.2.

Remark 3.5. The proof of Theorem 3.1 reveals that the same result can be obtained by using an allocation $(y_1, \dots, y_N)$ associated to *any* vertex of the feasible set of Problem (3.2) (rather than an optimal allocation $(y_1^\star, \dots, y_N^\star)$). Proposition 3.1 can still be applied and the proof remains unchanged. △

Theorem 3.1 guarantees that the computed solution is feasible for the target MILP (3.1), but, in general, there is a certain degree of suboptimality. In the following, we provide suboptimality bounds under Slater's constraint qualification for LP (3.2).

Assumption 3.2. *For a given $\sigma > 0$, there exist vectors $\widehat{z}_1 \in \text{conv}(X_1^{\text{MILP}}), \dots, \widehat{z}_N \in \text{conv}(X_N^{\text{MILP}})$ that satisfy*

$$\zeta \triangleq \min_{s \in \{1,\dots,S\}} \Big[b - \sigma - \sum_{i=1}^N A_i \widehat{z}_i \Big]_s > 0, \tag{3.15}$$

where $[\cdot]_s$ extracts the s-th component. The cost associated to $(\widehat{z}_1, \dots, \widehat{z}_N)$ is denoted by $J^{\text{SL}} = \sum_{i=1}^N c_i^\top \widehat{z}_i$. △

The following result establishes an a-priori suboptimality bound on the mixed-integer solution with $\sigma = \sigma^\infty$ as in (3.11).

Theorem 3.2 (A-Priori Suboptimality Bound)**.** *Consider the same assumptions and quantities of Theorem 3.1 and let also Assumption 3.2 hold. Then, $(x_1^\infty, \dots, x_N^\infty)$ satisfies the suboptimality bound $\sum_{i=1}^N c_i^\top x_i^\infty - J^{\text{MILP}} \leq B$, where J^{MILP} is the optimal cost of* (3.1) *and B is defined as*

$$B \triangleq \left(S + \frac{N \|\sigma^\infty\|_\infty}{\zeta} \right) \max_{i \in \{1,\dots,N\}} \gamma_i, \tag{3.16}$$

with ζ defined in (3.15)*, and*

$$\gamma_i \triangleq \max_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i - \min_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i, \qquad i \in \{1, \dots, N\}.$$

The proof is provided in Section 3.6.2.

Note that, although the bound provided by Theorem 3.2 is formally analogous to [121, Theorem 3.3], there is an implicit difference due to the restriction values (cf. Remark 3.4). In particular, our bound is tighter since σ^∞ is less than or equal to the restriction proposed by [121]. An even tighter bound can be derived by using the steady-state solution of the algorithm and computing also the primal solution of (2.12) for all i. For this reason, we call this bound a posteriori, since it depends on the solution of (3.2) computed by Algorithm 3.

Corollary 3.1 (A-Posteriori Suboptimality Bound). *Consider the same assumptions and quantities of Theorem 3.2. Then,* $\sum_{i=1}^{N} c_i^\top x_i^\infty - J^{\text{MILP}} \le B'$*, where* B' *is defined as*

$$B' \triangleq \sum_{i \in I_{\mathbb{R}}} (c_i^\top x_i^\infty - c_i^\top z_i^\star) + \frac{\|\sigma^\infty\|_\infty}{\zeta} \bigg(J^{\text{SL}} - \sum_{i=1}^{N} c_i^\top z_i^\star \bigg), \tag{3.17}$$

with ζ *and* J^{SL} *defined in Assumption 3.2 and* $I_{\mathbb{R}}$ *containing the indices of agents such that* $z_i^\star \notin X_i^{\text{MILP}}$ *(*$|I_{\mathbb{R}}| \le S$*).*

Proof. It is sufficient to follow the same line of Theorem 3.2, but stopping the proof of (i) at (3.49) and stopping the proof of (ii) at (3.50). ■

3.3.5 Finite-time Analysis

Let us now provide a finite-time analysis of the distributed algorithm. To this end, we assume that the restriction is equal to an enlarged version of the asymptotic restriction in (3.11), i.e.

$$\sigma^{\text{FT}} = \sigma^\infty + \delta \mathbf{1}, \tag{3.18}$$

for an arbitrary $\delta > 0$. As it will be clear, the purpose of the additional restriction $\delta\mathbf{1}$ is to compensate the distance of the estimated local allocations y_i^t to the optimal ones $y_i^\star$. We assume Problem (3.2) is feasible and that Assumption 3.1 holds with this new restriction. We provide two results that extend the results of Section 3.3.4 to a finite-time setting. Let us recall the assumption on the step size.

Assumption 3.3. *The step-size sequence* $\{\alpha^t\}_{t\ge 0}$*, with each* $\alpha^t \ge 0$*, satisfies* $\sum_{t=0}^{\infty} \alpha^t = \infty$*,* $\sum_{t=0}^{\infty} \big(\alpha^t\big)^2 < \infty$*.* △

At a high level, finite-time feasibility hinges upon the fact that the allocation sequence $\{y_1^t, \ldots, y_N^t\}_{t \ge 0}$ approaches an optimal allocation. Eventually, the distance of the current allocation to the optimal one can be embedded in the additional restriction δ. The next theorem formalizes this result.

Theorem 3.3 (Finite-time feasibility). *Let* $\sigma = \sigma^{\text{FT}}$ *as in* (3.18)*, for some* $\delta > 0$*, and let Problem* (3.2) *be feasible and satisfy Assumption 3.1. Consider the mixed-integer sequence* $\{x_1^t, \ldots, x_N^t\}_{t \ge 0}$ *generated by Algorithm 3 under Assumption 3.3, with* $\sum_{i=1}^{N} y_i^0 = b - \sigma^{\text{FT}}$*. There exists a sufficiently large time* $T_\delta > 0$ *such that the vector* $(x_1^t, \ldots, x_N^t)$ *is a feasible solution for Problem* (3.1)*, i.e.* $x_i^t \in X_i^{\text{MILP}}$ *for all* $i \in \{1, \ldots, N\}$ *and* $\sum_{i=1}^{N} A_i x_i^t \le b$*, for all* $t \ge T_\delta$*.* △

The proof is provided in Section 3.6.2.

Operatively, Theorem 3.3 can be applied to Algorithm 3 by setting a sufficiently large number of iterations $T_f \geq T_\delta$. It remains open to determine an upper bound of T_δ a priori, however from the proof it emerges that the value of T_δ essentially depends on the convergence rate of the distributed primal decomposition algorithm for convex problems applied to (3.2). Specifically, T_δ is the number of iterations required to reach a δ-optimal allocation. In principle, the smaller is δ, the longer it takes to reach the time T_δ for which the mixed-integer vector $(x_1^t, \ldots, x_N^t)$ satisfies the coupling constraint *for all* $t \geq T_\delta$. Next, we provide a suboptimality bound for the solution computed by Algorithm 3.

Theorem 3.4 (Finite-time suboptimality bound). *Consider the same assumptions and quantities of Theorem 3.3 and let also Assumption 3.2 hold. Then, there exists a time* $T_\delta > 0$ *such that the vector* $(x_1^t, \ldots, x_N^t)$ *satisfies the bound* $\sum_{i=1}^N c_i^\top x_i^t - J^{\text{MILP}} \leq B^t$ *for all* $t \geq T_\delta$*, with* B^t *being*

$$B^t \triangleq \sum_{i=1}^{N} (c_i^\top x_i^t - J_i^{\text{LP},t}) + \delta \sum_{i=1}^{N} \|\mu_i^t\|_1 + \Gamma \|\sigma^{\text{FT}}\|_\infty, \tag{3.19}$$

where J^{MILP} *is the optimal cost of* (3.1), $J_i^{\text{LP},t}$ *and* μ_i^t *are the optimal cost and a Lagrange multiplier of* (2.12) *at time* t, $\Gamma = \frac{N}{\zeta} \sum_{i=1}^N \Big(\max_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i - \min_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i \Big)$*, and* ζ *is associated to any Slater point (cf. Assumption 3.2).*

The proof is provided in Section 3.6.2.

We point out that, differently from the asymptotic case, to compute the bound (3.19) agents do not need the optimal solution of Problem (2.12) but only the optimal cost $J_i^{\text{LP},t}$. This can be obtained as a byproduct of a local dual subgradient method (see also Remark 3.3). Notice also that the bound (3.19) is "a posteriori" (i.e. it depends on the computed solution). If an a-priori bound with restriction σ^{FT} is desired, one can still apply Theorem 3.2 with σ^{FT} in place of σ^∞.

3.3.6 Monte Carlo Numerical Computations

In this section, we provide a computational study on randomly generated MILPs to compare Algorithm 3 with [121]. We consider large-scale problems with a total of 4500 optimization variables (3000 are integer and 1500 are continuous).

There are $N = 300$ agents and $S = 5$ coupling constraints. The local constraints X_i^{MILP} are subsets of $\mathbb{Z}^{10} \times \mathbb{R}^5$ satisfying $D_i x_i \leq d_i$, where D_i and $d_i \in \mathbb{R}^{20}$ have random entries in $[0, 1]$ and $[20, 40]$ respectively. Box constraints $-60\,\mathbf{1} \leq x_i \leq 60\,\mathbf{1}$ are added to ensure compactness. The cost vector is $c_i = D_i^\top \hat{c}_i$, where $\hat{c}_i$ has random entries in $[0, 5]$. As for the coupling, the matrices A_i are random with entries in $[0, 1]$ and the resource vector $b \in \mathbb{R}^5$ is random with values in two different intervals. Specifically, we first pick values in $[-20N, -15N]$, which results in a "loose" b, then we pick values in $[-180N, -175N]$, which results in a "tight" b.

A total of 100 MILPs with loose b and 100 MILPs with tight b are generated. For each problem, we check feasibility of problem (3.2) for both the asymptotic restriction of our method (3.11) and the restriction in [121]. Then, both algorithms are executed up to asymptotic convergence to evaluate the mixed-integer solution suboptimality. The results are summarized in Figures 3.5 and 3.6, where the restriction size is computed as $\|\sigma\|/\|b\|$ and the suboptimality is (over-)estimated as $\big(\sum_{i=1}^{N} c_i^\top x_i^\infty - J^{\text{LP}}\big)/J^{\text{LP}}$, with J^{LP} being the optimal cost of (3.2). In the figure, the caption "number of solvable problems" indicates the number of instances for which Problem (3.2) is feasible. For loose b, both methods are always applicable but our approach provides better solution performance than [121]. For tight resource vectors, our method is still applicable in the 70% of the cases, and provides an average suboptimality of 6.91%, while the approach in [121] cannot be applied due to infeasibility of Problem (3.2) (caused by the too large restrictions). It is worth noting that the values of tight b cannot be further reduced. Indeed, for smaller values of b, the target MILP (3.1) becomes infeasible.

Finally, we show the evolution of Algorithm 3 on a single problem instance over an Erdős-Rényi graph with edge probability 0.2. Figure 3.7 shows the value of the coupling constraints along the algorithmic evolution, with $\delta = 0.5$ (cf. the finite-time restriction (3.18)). Note that feasibility is achieved in finite time, within approx. 400 iterations, confirming Theorem 3.3.

	Algorithm 3		[121]	
	b loose	b tight	b loose	b tight
num. of solvable problems	100%	70%	100%	0%
size of restriction	7.4%	0.72%	66.9%	6.63%
suboptimality of solution	0.06%	6.91%	0.7%	N.A.

Figure 3.5: Monte Carlo simulations on random MILPs: performance comparison of Algorithm 3 and of the method in [121]. The second and the third row are averaged over the Monte Carlo trials. See the text for further details.

3.4 Distributed Benders Decomposition for MILPs

The solution method introduced in the previous section is independent of the algorithmic strategy used to solve the primal decomposition master Problem (3.3). Indeed, the asymptotic results in Section 3.3.4 are based on the assumption that an optimal allocation is known, however it is not necessary that it is computed with the Distributed Primal Decomposition algorithm for convex problems discussed in Section 2.2.5. In this section, we propose a new distributed algorithm that computes a feasible solution to the original

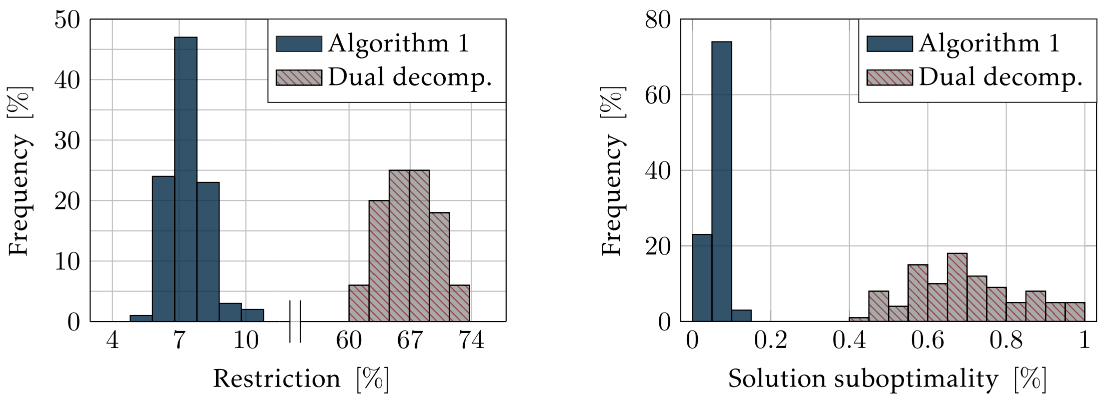


Figure 3.6: Monte Carlo simulations on random MILPs with loose b: comparison histograms of Algorithm 3 and of the method in [121]. See the text for details.

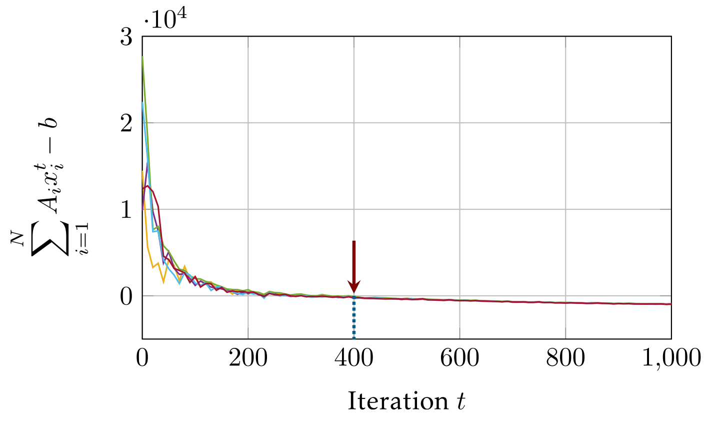


Figure 3.7: Components of the coupling constraint associated to $(x_1^t, \dots, x_N^t)$ generated by Algorithm 3 on a random MILP. In this simulation we forced the agents to compute a mixed-integer solution according to (3.7) at every iteration. The red arrow indicates the iteration T_δ in which the solution becomes definitively feasible.

MILP combining the strategy described in the previous sections with an alternative approach to solve the master Problem (3.3). Differently from Algorithm 3, the new distributed algorithm is based on the so-called cutting-plane technique, which allows for *finite-time convergence* to an optimal allocation and allows the agents to compute a feasible solution to the MILP (3.1) in finite time without resorting to the additional restriction discussed in Section 3.3.5.

Let us consider again the mixed-integer linear program (3.1), recalled here

$$\begin{aligned} \min_{x_1,\dots,x_N} \quad & \sum_{i=1}^N c_i^\top x_i \\ \text{subj. to} \quad & \sum_{i=1}^N A_i x_i \le b \\ & x_i \in X_i^{\text{MILP}}, \qquad i = 1,\dots,N. \end{aligned}$$

We maintain the same assumptions on the information sparsity as in Section 3.2.1. A relevant feature of the new distributed algorithm is that it can work under much more general assumptions on the communication network than Algorithm 3. In particular, in this section we assume that the N agents communicate according to a time-varying digraph $\mathcal{G}_c^t = (V, \mathcal{E}^t)$, where t denotes time, $V = \{1,\dots,N\}$ is the vertex set and $\mathcal{E}^t \subseteq V \times V$ is the edge set at time t. We will denote by $\mathcal{N}_i^t$ the in-neighbor set of agent i at time t, i.e. $\mathcal{N}_i^t = \{j \mid (j,i) \in \mathcal{E}^t\}$. We make the following assumption on the graph connectivity.

Assumption 3.4. *The communication graph $\mathcal{G}_c^t$ is jointly strongly connected, i.e. the graph $\mathcal{G}_\infty^t \triangleq (\{1,\dots,N\}, \mathcal{E}_\infty^t)$, with $\mathcal{E}_\infty^t = \bigcup_{\tau=t}^\infty \mathcal{E}^\tau$, is strongly connected for all $t \ge 0$.* $\triangle$

Assumption 3.4 has an extremely broad scope. Notably, it guarantees that the new algorithm can also be implemented asynchronously (more details will be given in Section 3.4.5). Moreover, it also ensures that the algorithm can work in unreliable communication networks, and, in particular, in networks subject to packet losses.

3.4.1 Review of Benders Decomposition

The new distributed algorithm is based on the so-called Benders decomposition, which has tight connections with the primal decomposition method discussed so far. We now recall the basics of this technique applied to our problem using the formalism of [13, Section 6.5].

Linear Programming Reformulation of Master Problem

Similarly to Section 3.2.2, the decomposition scheme is applied to the LP approximation (3.2) by introducing a hierarchical problem structure. The master problem is

$$\begin{aligned} \min_{y_1,\ldots,y_N} \quad & \sum_{i=1}^{N} p_i(y_i) \\ \text{subj. to} \quad & \sum_{i=1}^{N} y_i = b - \sigma, \end{aligned} \tag{3.20}$$

where, for all $i \in \{1, \ldots, N\}$, $p_i(y_i) : \mathbb{R}^S \to \overline{\mathbb{R}}$ is again equal to the optimal cost of the i-th subproblem

$$\begin{aligned} p_i(y_i) = \min_{z_i} \quad & c_i^\top z_i \\ \text{subj. to} \quad & A_i z_i \leq y_i \\ & z_i \in \text{conv}(X_i^{\text{MILP}}). \end{aligned} \tag{3.21}$$

together with the convention that $p_i(y_i) = \infty$ if Problem (3.21) is infeasible (for this reason this time we define the function p_i with codomain in the extended reals $\overline{\mathbb{R}}$). Of course, in solving Problem (3.20), we should only consider those $(y_1, \ldots, y_N)$ for which none of the $p_i(y_i)$ are equal to infinity.

Let us show more deeply the structure of Problem (3.20). To this end, for each fixed $y_i \in \mathbb{R}^S$, consider the dual problem of (3.21), i.e.

$$\max_{\mu_i \geq 0} \left[-y_i^\top \mu_i + \min_{z_i \in \text{conv}(X_i^{\text{MILP}})} \big((c_i + A_i^\top \mu_i)^\top z_i \big) \right]. \tag{3.22}$$

By adding an epigraph variable to Problem (3.22) (Appendix A.1), we obtain the equivalent formulation

$$\begin{aligned} \max_{\mu_i, \eta_i} \quad & \eta_i - y_i^\top \mu_i \\ \text{subj. to} \quad & \mu_i \geq 0 \\ & \eta_i \leq \min_{z_i \in \text{conv}(X_i^{\text{MILP}})} \big((c_i + A_i^\top \mu_i)^\top z_i \big). \end{aligned} \tag{3.23}$$

Due to linearity of the cost in the inner minimization, we may restate the last constraint as

$$\eta_i \leq \min_{z_i \in \text{vert}(X_i^{\text{MILP}})} \big((c_i + A_i^\top \mu_i)^\top z_i \big), \tag{3.24}$$

where $\text{vert}(X_i^{\text{MILP}})$ is the set of vertices of $\text{conv}(X_i^{\text{MILP}})$, which has a finite number of elements, denoted in the following by $\widehat{x}_i^j$ for $j \in \mathcal{J}_i \triangleq \{1, \ldots, |\text{vert}(X_i^{\text{MILP}})|\}$. The scalar η_i is less than or equal to the minimum of $(c_i + A_i^\top \mu_i)^\top z_i$ over $\text{vert}(X_i^{\text{MILP}})$ if and only if it is less than or equal to the value of such function for each element in $\text{vert}(X_i^{\text{MILP}})$.

Therefore, we obtain the following reformulation of Problem (3.23),

$$\begin{aligned} \max_{\mu_i,\eta_i} \; & \eta_i - y_i^\top \mu_i \\ \text{subj. to } \; & \mu_i \geq 0 \\ & \eta_i \leq (c_i + A_i^\top \mu_i)^\top \widehat{x}_i^j, \qquad \forall\, j \in \mathcal{J}_i. \end{aligned} \tag{3.25}$$

Note that the last formulation (3.25) of the dual problem is a linear program and its optimal cost is equal to $p_i(y_i)$ (by LP strong duality). Let $\mathcal{P}_i$ be the feasible set of Problem (3.25),

$$\mathcal{P}_i = \big\{ (\mu, \eta_i) \mid \mu_i \geq 0 \text{ and } \eta_i \leq (c_i + A_i^\top \mu_i)^\top \widehat{x}_i^j, \;\; \forall j \in \mathcal{J}_i \big\}.$$

Differently from the primal formulation (3.21), the dual feasible set $\mathcal{P}_i$ does not depend on the value of y_i. Let $v_i^j = (v_{\mu_i}^j, v_{\eta_i}^j), j \in \mathcal{V}_i$ be the vertices and $w_i^j = (w_{\mu_i}^j, w_{\eta_i}^j), j \in \mathcal{R}_i$ be a complete set of extreme rays of $\mathcal{P}_i$. For each fixed y_i, there are two possibilities to consider: either *(i)* Problem (3.21) is feasible and admits an optimal solution in the compact set $\text{conv}(X_i^{\text{MILP}})$, in which case the dual problem (3.25) also admits an optimal solution, or *(ii)* Problem (3.21) is infeasible and Problem (3.22) is unbounded. To guarantee that Problem (3.22) is not unbounded we impose

$$w_{\eta_i}^j - y_i^\top w_{\mu_i}^j \leq 0, \qquad \forall\, j \in \mathcal{R}_i. \tag{3.26}$$

If (3.26) holds, then $p_i(y_i) < \infty$ is the optimal cost of Problem (3.22) and the optimum must be attained at a vertex of $\mathcal{P}_i$, in particular,

$$p_i(y_i) = \max_{j \in \mathcal{V}_i} \big(v_{\eta_i}^j - y_i^\top v_{\mu_i}^j \big).$$

Equivalently, $p_i(y_i)$ is the smallest number ν_i such that

$$v_{\eta_i}^j - y_i^\top v_{\mu_i}^j \leq \nu_i, \qquad \forall\, j \in \mathcal{V}_i. \tag{3.27}$$

The previous two characterizations (3.26) and (3.27) can be embedded in the master problem (3.20), by which we finally obtain the linear programming formulation of (3.20),

$$\begin{aligned} \min_{\substack{y_1,\ldots,y_N \\ \nu_1,\ldots,\nu_N}} \; & \sum_{i=1}^N \nu_i \\ \text{subj. to } \; & \sum_{i=1}^N y_i = b - \sigma \\ & v_{\eta_i}^j - y_i^\top v_{\mu_i}^j \leq \nu_i, \qquad \forall\, j \in \mathcal{V}_i, \;\; i \in \{1, \ldots, N\} \\ & w_{\eta_i}^j - y_i^\top w_{\mu_i}^j \leq 0, \qquad \forall\, j \in \mathcal{R}_i, \; i \in \{1, \ldots, N\}. \end{aligned} \tag{3.28}$$

In the following, we will compactly denote the overall optimization variable of (3.28) by using the boldface symbols $(\mathbf{y}, \boldsymbol{\nu}) \in \mathbb{R}^{N(S+1)}$. The distributed algorithm introduced next is based on the solution of the master problem in this last formulation (3.28). However, we point out that the number of inequality constraints of (3.28) can be extremely large, since it is usually exponential in the number of integer variables of each X_i^{MILP}. Moreover, the constraints are not known in advance because the vertices v_i^j and the extreme rays w_i^j of $\mathcal{P}_i$ should be explicitly computed. Thus, it is not affordable to solve the problem by pre-computing all the constraints. To overcome these issues, we will employ a cutting-plane method, which informally consists of repeatedly generating separating hyperplanes to approximate the original feasible set[2].

Before outlining the solution mechanism, let us highlight the connection between Problem (3.28) and the primal decomposition master problem (3.3). The two problems are conceptually identical. However, in the jargon of Benders decomposition the constraints (3.26) are called *feasibility cuts*, since they are generated to ensure that the local problems are feasible. Thus, they are effectively a characterization of the constraints $y_i \in Y_i$ in Problem (3.3), i.e.

$$Y_i = \left\{ y_i \in \mathbb{R}^S \mid w_{\eta_i}^j \le y_i^\top w_{\mu_i}^j \ \ \forall j \in \mathcal{R}_i \right\}.$$

The constraints (3.27), on the other hand, are called *optimality cuts*, since they are generated to ensure that the epigraph variables ν_i estimate correctly the value of $p_i(y_i)$ and, consequently, that the optimal solution of the master problem (3.28) has indeed minimal cost. The Benders decomposition formulation highlights more clearly the linear programming nature of the master problem, which is the key for the development of a finite-time convergent algorithm.

Online Constraint Generation

Within the Benders decomposition tecnique, the typical way to deal with Problem (3.28) is to use an *online* constraint generation mechanism. Instead of considering the full master problem (3.28), a relaxed instance is considered with the same objective but only a subset of the constraints in (3.28). After solving such relaxed instance, new constraints are generated (if necessary), and the process is repeated until convergence.

Let us now recall in more detail the constraint generation method, since it will be used in the distributed algorithm. Suppose that a relaxed instance of the master problem (3.28) has been solved, and denote by $(\bar{\mathbf{y}}, \bar{\boldsymbol{\nu}})$ the computed optimal solution. To assess whether this solution can be accepted as true optimal solution of (3.28), we need to check that all the constraints are satisfied. Instead of doing this calculation

[2]In cutting-plane algorithms, the generated constraints are typically called *cutting planes* (or simply *cuts*) because they "cut away" a portion of the space, marking it as infeasible.

directly, we consider for all $i \in \{1, \dots, N\}$ the subproblems (3.21) with $y_i = \bar{y}_i$, which are assumed to be solved through their dual formulation (3.25). There are three cases to consider for all $i \in \{1, \dots, N\}$.

(i) Problem (3.25) with $y_i = \bar{y}_i$ is unbounded. In this case, the primal problem (3.21) is infeasible. Let $w_i^\kappa = (w_{\eta_i}^\kappa, w_{\mu_i}^\kappa)$ be an extreme ray along which Problem (3.25) is unbounded. Then, it satisfies $w_{\eta_i}^\kappa - \bar{y}_i^\top w_{\mu_i}^\kappa > 0$. Note that this is a violated constraint of Problem (3.28), thus we can generate the feasibility cut

$$w_{\eta_i}^\kappa - y_i^\top w_{\mu_i}^\kappa \leq 0.$$

(ii) Problem (3.25) with $y_i = \bar{y}_i$ admits an optimal solution. Let $v_i^\kappa = (v_{\eta_i}^\kappa, v_{\mu_i}^\kappa)$ be an optimal vertex. If it holds $v_{\eta_i}^\kappa - \bar{y}_i^\top v_{\mu_i}^\kappa > \bar{\nu}_i$, this is a violated constraint of Problem (3.28), thus we can generate the optimality cut

$$v_{\eta_i}^\kappa - y_i^\top v_{\mu_i}^\kappa \leq \nu_i.$$

(iii) If none of the previous cases occurs, both the conditions (3.26) and (3.27) hold for the considered $(\bar{y}_i, \bar{\nu}_i)$. In this case, no cutting-planes must be generated and $\bar{\nu}_i = p_i(\bar{y}_i)$.

In the following, we will denote the output of this constraint generation procedure for each fixed $i \in \{1, \dots, N\}$ as CONSTRAINTORACLE$(\bar{y}_i, \bar{\nu}_i)$ (note that $\bar{y}_i$ and $\bar{\nu}_i$ are the only data required for the constraint generation). It should be noted that, if there is a violated constraint, the CONSTRAINTORACLE procedure is always able to detect this fact and to generate it. On the contrary, if the procedure does not generate new constraints, it means that both (3.26) and (3.27) hold true. In the next, this constraint generation mechanism will be embedded within the distributed algorithm solving Problem (3.28).

3.4.2 Distributed Algorithm Description

The distributed algorithm to solve Problem (3.28) is based on the Constraints Consensus algorithm [84] with online generation of cutting planes (such as in the works [18, 117]). Differently from the Distributed Primal Decomposition algorithm, in this class of algorithms agents do not exchange their local solutions or multipliers but instead they exchange a set of constraints forming a *basis* of their current solution at a vertex (Appendix A.3). Given an optimal vertex $(\mathbf{y}, \boldsymbol{\nu}) \in \mathbb{R}^{N(S+1)}$ of Problem (3.28), it is well known from linear programming that a basis associated to $(\mathbf{y}, \boldsymbol{\nu})$ consists of $N(S+1)$ active constraints (i.e. constraints satisfied with the equality at $(\mathbf{y}, \boldsymbol{\nu})$) such that the relaxed LP, containing only the constraints in the basis, has the same optimal cost of (3.28). Note that in general there may exist multiple optimal vertices having the same cost in a linear program. This may lead to inconsistent choices of the bases among the agents, because each agent can compute its basis on a potentially different vertex. In the inspiring Constraints Consensus [84], this is prevented with a tie-break rule to ensure that all the nodes agree on the same solution. In particular, we will consider the *lexicographically minimal optimal solution* of Problem (3.28), also called *lex-optimal solution* (Appendix A.3). By analogy, we will call *lex-optimal basis* the basis corresponding to the lex-optimal solution.

Let us now introduce our algorithm *Distributed Benders Decomposition for MILPs*. Agents maintain and update a local estimate of the lex-optimal basis of Problem (3.28). We denote the basis of agent i at time t with B_i^t and we say that the tuple (a, b, f, j), with $a \in \mathbb{R}^S$, $b, f \in \mathbb{R}$ and $j \in \{1, \ldots, N\}$, belongs to B_i^t if the constraint $a^\top y_j + b\nu_j \leq f$ is in the basis of agent i at time t. At each iteration t, agent i has a current guess of the lex-optimal solution of Problem (3.28), which we denote as $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t) \in \mathbb{R}^{N(S+1)}$ (in order to keep the notation light[3]) and is associated to the basis B_i^t. Since the computed solution may violate some of the constraints, at each iteration t the agent generates a constraint (if any) by calling ConstraintOracle(y_i^t, ν_i^t), where $(y_i^t, \nu_i^t) \in \mathbb{R}^{S+1}$ denotes the i-th block of $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t) \in \mathbb{R}^{N(S+1)}$. Then, agent i forms a local instance of Problem (3.28) with the newly generated constraint and the constraints received from the neighbors, which is solved to obtain a new guess of the solution, and the associated basis is communicated to neighbors. This scheme converges in finite time to an optimal solution $(\mathbf{y}^\star, \boldsymbol{\nu}^\star)$ of Problem (3.28). Then, agent i uses $y_i^\star$ (the i-th block of $\mathbf{y}^\star$) to compute a tentative mixed-integer solution using the technique described in Section 3.3. Algorithm 4 summarizes the steps performed by each agent i. As regards the initial coordination to run Algorithm 4, similar reasonings as in Remark 3.2 hold.

For consistency with the previous section, in the table we used the symbol x_i^∞ to denote the mixed-integer solution computed by the algorithm. However, we stress that Algorithm 4 converges in finite time, which means that the quantity x_i^∞ must not be intended as computed asymptotically (even though it coincides with the asymptotic solution provided by Algorithm 3).

In order to cope with unbounded problems at an early stage of the algorithm execution, Problem (3.29) requires an additional bounding box $-R\mathbf{1} \leq y, \boldsymbol{\nu} \leq R\mathbf{1}$, with $R > 0$ a sufficiently large number. In Section 3.4.3, we provide details on how to implement the constraint generation step ConstraintOracle. Problem (3.29) can be solved by using any lexicographic solver for LPs, while Problem (3.30) can be solved as described in Section 3.3.2.

[3]We are slightly overloading the notation. Indeed, y_i denotes the i-th component of $\mathbf{y}$ (i.e. the subscript identifies the component of the vector), while $\mathbf{y}_i^t$ denotes the estimate of $\mathbf{y}$ of agent i at time t (i.e. the subscript identifies the agent that has computed the vector). However, we prefer not to introduce further symbols to differentiate between the two roles of the subscript since this second case occurs only for $\mathbf{y}_i^t$, $\boldsymbol{\nu}_i^t$ and B_i^t.

Algorithm 4 Distributed Benders Decomposition for MILPs

```
**Initialization:** set $y_i^0 \in \mathbb{R}^S, \nu_i^0 \in \mathbb{R}$ and $B_i^0 = \emptyset$
**Repeat** for $t = 0, 1, \ldots$ **until convergence**
    **Generate** constraint tuple (if any) as $h_i^t = \textsc{ConstraintOracle}(y_i^t, \nu_i^t)$
    **Receive** bases $B_j^t$ from $j \in \mathcal{N}_i^t$ and set
        $H_{\text{TMP}} = B_i^t \cup \big( \cup_{j \in \mathcal{N}_i^t} B_j^t \big) \cup \{h_i^t\}$
    **Compute** $(\mathbf{y}_i^{t+1}, \boldsymbol{\nu}_i^{t+1})$ as the lex-optimal solution (with basis $B_i^{t+1}$) of
        $\min_{\substack{y_1,\ldots,y_N \\ \nu_1,\ldots,\nu_N}} \sum_{i=1}^N \nu_i$
        subj. to $\sum_{i=1}^N y_i = b - \sigma$                                        (3.29)
                 $a^\top y_j + b\nu_j \le f, \quad \forall\, (a, b, f, j) \in H_{\text{TMP}}$
                 $-R\mathbf{1} \le \mathbf{y}, \boldsymbol{\nu} \le R\mathbf{1}$
**Return** $x_i^\infty$ as the optimal solution of
        $\text{lex-min}_{v_i, \xi_i, x_i} \; v_i$
        subj. to $c_i^\top x_i \le \xi_i$                                                  (3.30)
                 $A_i x_i \le y_i^\star + v_i \mathbf{1}$
                 $x_i \in X_i^{\text{MILP}}, \; v_i \ge 0.$
```

Remark 3.6 (Size of communicated bases)**.** As already discussed, a basis consist of $N(S+1)$ active constraints. Since the S equality constraints of Problem (3.28) are always active at any feasible vector and are known by all the agents, the size of communicated bases is at most of $N(S+1) - S$ active inequality constraints. Indeed, constraints of the additional bounding box that are part of the basis need not be communicated. △

3.4.3 Routine for the Local Problem

Let us elaborate on the implementation of the constraint generation step ConstraintOracle. Owing to the discussion of Section 3.4.1, in order to generate a constraint, we must be able to solve an instance of Problem (3.25). We must also be able to provide an optimal vertex if the problem has finite optimal cost or specify an extreme ray along which the problem is unbounded. In theory, this task could be accomplished by any numerical solver based on the simplex algorithm. However, as it happens for Problem (3.28), the constraints in Problem (3.25) are not known a-priori. Therefore, we now formulate a *local* finite-time algorithm to solve Problem (3.25), in which it is assumed that y_i is fixed to a given $\bar{y}_i \in \mathbb{R}^S$. In order to avoid excessive technicalities, we present the local routine with the assumption that no constraint in (3.25) is orthogonal to the direction of minimization (i.e. we assume $\bar{y}_i \neq A_i \widehat{x}_i^j$ for all $j \in \mathcal{J}_i$) and then discuss how to handle the general case. To differentiate the notation with the distributed algorithm, here we denote the iterations with the letter k.

Procedure 5 Local Routine for Problem (3.25)

```
**Initialization:** set $\mu_i^0 \geq 0$ and $\eta_i^0 = +\infty$
**Repeat** for $k = 0, 1, \dots$
  **Compute** $x_i^k$ as optimal solution of
      $$\min_{x_i \in \text{vert}(X_i^{\text{MILP}})} \big(c_i + A_i^\top \mu_i^k\big) x_i \tag{3.31}$$
  **If** $\eta_i^k = (c_i + A_i^\top \mu_i^k)^\top x_i^k$ **then** set $K_f = k$ **and go to final step**
  **Else compute** $(\mu_i^{k+1}, \eta_i^{k+1})$ as optimal solution of
      $$\max_{\mu_i, \eta_i} \ \eta_i - \bar{y}_i^\top \mu_i \tag{3.32a}$$
      $$\text{subj. to } \mu_i \leq Q\mathbf{1} \tag{3.32b}$$
      $$\mu_i \geq 0 \tag{3.32c}$$
      $$\eta_i \leq c_i^\top x_i^\ell + \mu_i^\top A_i x_i^\ell, \qquad \ell = 0, \dots, k \tag{3.32d}$$
**Final step: If** $\mu_i^{K_f} < Q\mathbf{1}$ **then** optimal cost of (3.25) is finite – return $(\mu_i^{K_f}, \eta_i^{K_f})$
**Else** Problem (3.25) is unbounded – return unbounded extreme ray of Problem (3.32)
without constraint $\mu_i \leq Q\mathbf{1}$.
```

The routine approximates Problem (3.25) with iterative refinements by generating the necessary constraints. The fictious initialization $\eta_i^0 = +\infty$ ensures that Problem (3.32) is solved at least once, moreover Problem (3.32) always admits an optimal solution due to the artificial bounding box $\mu_i \leq Q\mathbf{1}$.

The next proposition summarizes the convergence result of the proposed local routine under the mentioned simplifying assumption.

Proposition 3.2. *Let $\bar{y}_i \in \mathbb{R}^S$ be given and let the sequence $\{(\mu_i^k, \eta_i^k)\}_{k \geq 0}$ be generated by Procedure 5 for a sufficiently large $Q > 0$. Moreover, assume that $\bar{y}_i \neq A_i \widehat{x}_i^j$ for all $j \in \mathcal{J}_i$. Then, after a finite number of iterations $K_f > 0$,*

(i) if Problem (3.25) *with $y_i = \bar{y}_i$ has finite optimal cost, the routine returns an optimal vertex;*

(ii) if Problem (3.25) *with $y_i = \bar{y}_i$ is unbounded, the routine returns an unbounded extreme ray.*

The proof is provided in Section 3.6.3.

Remark 3.7 (Computational load reduction)**.** A useful feature of Procedure 5 is that the constraints generated with a given $\bar{y}_i$ can be stored and re-used in future invocations of the routine with different $\bar{y}_i$. Indeed, as it emerges from the proof, the generated constraints are drawn from Problem (3.25) and thus they do not depend on the value of $\bar{y}_i$. △

Remark 3.8. With slight modifications, Procedure 5 can also be used as local routine to compute a Lagrange multiplier of Problem (3.5). In particular, the constraint (3.32b) ought to be replaced with $\mu_i^\top \mathbf{1} \leq M$ and the algorithm should directly return the solution of Problem (3.32) upon convergence (the final step is not needed). This local routine represents an alternative to the local dual subgradient scheme discussed in Remark 3.3 and guarantees finite-time convergence to an optimal dual solution of Problem (3.5). △

Let us briefly comment the general case in which $\bar{y}_i$ may be equal to $A_i \widehat{x}_i^j$ for some $j \in \mathcal{J}_i$. In this case, Problems (3.25) and (3.56) can have multiple optimal solutions, and similarly for Problem (3.32) at some iteration k. In particular cases, it may happen that, although the optimal cost of Problems (3.25) and (3.56) is finite, the LP solver may select an optimal solution of (3.32) on the bounding box. Thus, if the asymptotically computed vector $(\bar{\mu}_i, \bar{\eta}_i)$ is on the bounding box, in order to clearly distinguish between the two cases of Proposition 3.2, it is necessary to first compute all the optimal vertices of (3.32) and to solve Problem (3.31) for all of them. After generating all the associated constraints, the bounding box $\mu_i \leq Q\mathbf{1}$ can be safely removed and hence the execution of the algorithm can be resumed from Problem (3.32).

3.4.4 Convergence Analysis

In this section we provide a convergence analysis of Algorithm 4. The result we provide is twofold. First, we prove that all agents are consensual in finite time on an optimal solution of Problem (3.28). Consequently, the mixed-integer solution obtained with the computed allocation is a feasible solution of the original MILP (3.1) (with the suboptimality bounds provided in Section 3.3.4). The following theorem formalizes these facts.

Theorem 3.5. *Let σ be equal to* (3.11) *and let Problem* (3.2) *be feasible. Moreover, let Assumptions 3.1 and 3.4 hold and let $M, R > 0$ be sufficiently large. Consider the sequences $\{(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)\}_{t \geq 0}$ and the mixed-integer vectors x_i^∞ generated by Algorithm 4 for all $i \in \{1, \ldots, N\}$. Then,*

(i) there exists a sufficiently large (finite) time $T > 0$ such that $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t) = (\mathbf{y}^\star, \boldsymbol{\nu}^\star)$ for all $t \geq T$ and $i \in \{1, \ldots, N\}$, where $(\mathbf{y}^\star, \boldsymbol{\nu}^\star)$ is an optimal solution of Problem (3.28)*;*

(ii) *the vector* $(x_1^\infty, \ldots, x_N^\infty)$ *is a feasible solution for Problem* (3.1), *i.e.* $x_i^\infty \in X_i^{\text{MILP}}$ *for all* $i \in \{1, \ldots, N\}$ *and* $\sum_{i=1}^N A_i x_i^\infty \leq b$. △

Throughout the analysis, we denote by J_i^t the optimal cost of Problem (3.29) of the i-th agent at iteration t. To prove Theorem 3.5 we rely on two intermediate results, namely local convergence and consensus, that we now formalize.

Lemma 3.2 (Local convergence). *Under the assumptions of Theorem 3.5, for all* $i \in \{1, \ldots, N\}$ *the following holds:*

(i) *the cost sequence* $\{J_i^t\}_{t\geq 0}$ *and the solution sequence* $\{(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)\}_{t\geq 0}$ *converge in finite time to constant values* $\bar{J}_i$ *and* $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$;

(ii) *the* i*-th block of the vector* $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$ *satisfies conditions* (3.26) *and* (3.27).

The proof is provided in Section 3.6.3. The next lemma is a reformulation of [84, Theorem IV.3 (ii)] (see also [117, Lemma 4.3] for a more self-contained proof).

Lemma 3.3 (Consensus). *Under the assumptions of Theorem 3.5, assume that the sequences* $\{J_i^t\}_{t\geq 0}$ *and* $\{(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)\}_{t\geq 0}$ *defined in Lemma 3.2, converge to constant values* $\bar{J}_i$ *and* $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$. *Then,* $\bar{J}_i = \bar{J}_j$, $\bar{\mathbf{y}}_i = \bar{\mathbf{y}}_j$ *and* $\bar{\boldsymbol{\nu}}_i = \bar{\boldsymbol{\nu}}_j$ *for all* $i, j \in \{1, \ldots, N\}$. △

We are now ready to prove Theorem 3.5.

Proof of Theorem 3.5. To prove *(i)*, it suffices to prove that, for all $i \in \{1, \ldots, N\}$, $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)$ converges in finite time to an optimal solution of Problem (3.28). By Lemma 3.2, for all $i \in \{1, \ldots, N\}$, the cost sequences J_i^t and the solution sequences $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)$ converge in finite time to $\bar{J}_i$ and $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$, and moreover conditions (3.26) and (3.27) hold for the i-th block of $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$. By Lemma 3.3, there exist a common $\bar{J} \in \mathbb{R}$ and a common $(\bar{\mathbf{y}}, \bar{\boldsymbol{\nu}})$ such that $\bar{J}_i = \bar{J}$, $\bar{\mathbf{y}}_i = \bar{\mathbf{y}}$ and $\bar{\boldsymbol{\nu}}_i = \bar{\boldsymbol{\nu}}$. Thus, $(\bar{\mathbf{y}}, \bar{\boldsymbol{\nu}})$ satisfies all the constraints of Problem (3.28), and therefore $\bar{J} \geq J^\star$, where $J^\star$ denotes the optimal cost of Problem (3.28). To prove that $\bar{J} \leq J^\star$, simply note that each agent obtains $\bar{J}$ as the optimal cost of Problem (3.29), which has a reduced number of constraints than the original Problem (3.28), and by assumption the bounding box is sufficiently large, so that we can assume that $R > \|(\mathbf{y}^\star, \boldsymbol{\nu}^\star)\|_\infty$. Thus $\bar{J} = J^\star$, which implies that $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$ is feasible and cost-optimal, therefore it is an optimal solution of (3.28). Part *(ii)* follows by Theorem 3.1. ■

3.4.5 Alternative Formulation and Further Discussion

In [26], an alternative approach to the Benders Decomposition scheme discussed in this section has been described. In particular, instead of following the procedure recalled in Section 3.4.1, a constraint generation distributed algorithm has been formulated starting from the epigraph version of Problem (2.7). The resulting algorithm is basically a distributed Benders decomposition scheme in which no feasibility cuts are generated (since the subproblems (2.8) are feasible for all i). The local routine is formulated in such a way that no extreme rays are returned, with a slightly different formulation of the additional bounding box in Problem (3.32). The convergence proof of the local routine has some differences but the analysis of the distributed algorithm can be almost repeated verbatim.

As we already mentioned, Assumption 3.4 makes it possible to implement Algorithm 4 in an asynchronous manner. In particular, each agent can i perform the local computations at its own speed. In the meanwhile, whenever a a basis is received from the in-neighbors, it is stored in a temporary memory (if two bases are received from the same in-neighbor the old one is overwritten). When solving Problem (3.29), agent i uses all the bases collected until that moment and finally, after a basis corresponding to the new solution is found, it is sent to the out-neighbors. In this respect, a remarkable property of the algorithm is that the agents can detect convergence in a fully distributed way under slightly more restrictive assumptions on the communication graph. In particular, the following stopping criterion can be used. If the graph is uniformly jointly strongly connected with period equal to B seconds, each agent i can conclude that convergence has occurred if its local solution $(y_i^t, \boldsymbol{\nu}_i^t)$ has not changed after $2BN + 1$ seconds [31, Theorem 1].

3.4.6 Numerical Example

In this section, we provide a numerical example that validates the theoretical analysis of Section 3.4.4. We only show how Algorithm 4 behaves on a single instance of Problem (3.1), since we already provided a detailed analysis on the restriction magnitude and solution performance in Section 3.3.6. By using the same generation model as in Section 3.3.6 (with $\hat{c}_i \in [0, 0.1]$), we generate a random MILP with $N = 30$ agents, $S = 3$ coupling constraints and resource vector b with entries in $[20, 120]$. As for the communication network, we randomly generate an Erdős-Rényi undirected connected graph with edge probability equal to 0.1.

First, we focus on how the algorithm solves Problem (3.28). In Figure 3.8, the evolution of the optimal cost of Problem (3.29) for all $i \in \{1, \ldots, N\}$, compared to the optimal cost of Problem (3.28), is shown. In an outer approximation fashion, the algorithm selects infeasible points $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)$ for Problem (3.28) that eventually become feasible and equal to each other. In our simulation, agents reach consensus in 56 iterations. The figure highlights also that in the early iterations (up to iteration 12 in this example) there are still insufficient constraints in the network, so that the solutions of Problem (3.29) are attained at the bounding box.

Next, we show the sequence of mixed-integer solutions $(x_1^t, \ldots, x_N^t)$ with the assumption that Problem (3.30) is solved at each iteration. In Figure 3.9, the evolution of primal

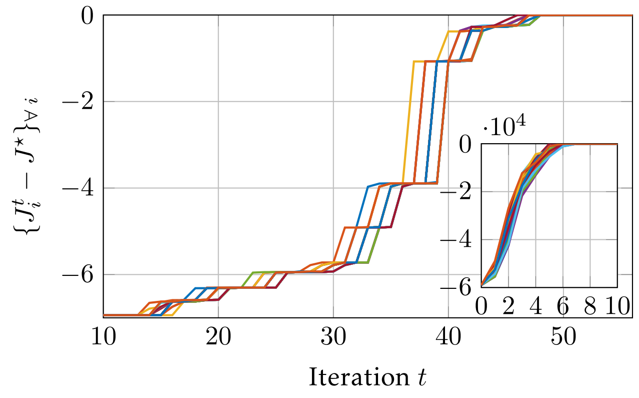


Figure 3.8: Evolution of the optimal cost J_i^t of Problem (3.29) for all $i \in \{1, \ldots, N\}$, compared to the optimal cost $J^\star$ of Problem (3.28). The inset figure shows the behavior of the algorithm in the early iterations.

feasibility with respect to the coupling constraint is shown. The algorithm is allowed to violate the constraints during the evolution, but, according to Theorem 3.1, it becomes feasible when convergence to an optimal solution of Problem (3.20) occurs. Note that from iteration 49 to the last iteration the value of the constraint is constant. In fact, convergence to $(x_1^\infty, \ldots, x_N^\infty)$ occurs at iteration 49, however it takes some time for the agents to autonomously detect convergence of the distributed scheme (cf. Section 3.4.5).

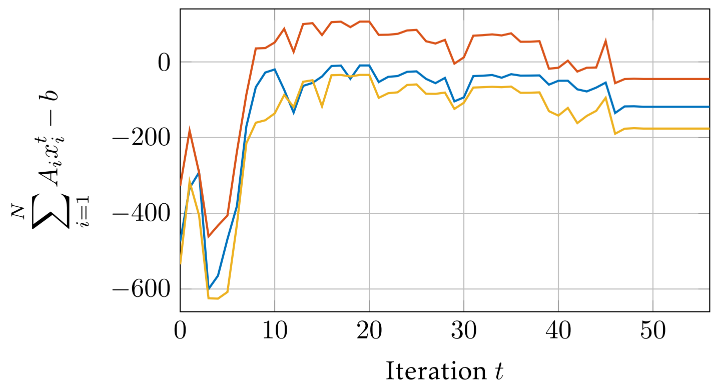


Figure 3.9: Evolution of the coupling constraint value. As soon as convergence is reached, the mixed-integer solution is guaranteed to be feasible for Problem (3.1). Indeed, all the lines in the graph eventually drop below zero.

3.5 Extension to General Nonconvex Programs

The solution approach for MILPs introduced in the previous sections depends crucially on the solution properties of the convex relaxation (3.2) formalized in Proposition 3.1. However, MILPs are only a special case in which such properties hold true. In this section, we extend the method to a broader class of nonconvex problems. The results of this section have appeared as [25].

3.5.1 Distributed Nonconvex Set-up and Convex Approximation

Let us formalize the distributed nonconvex optimization set-up considered throughout this section. We consider a network of N agents that must solve the optimization problem

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subj. to} \quad & \sum_{i=1}^{N} g_i(x_i) \le b \\ & x_i \in X_i, \qquad i = 1,\ldots,N, \end{aligned} \tag{3.33}$$

where $x_1 \in \mathbb{R}^{n_1}, \ldots, x_N \in \mathbb{R}^{n_N}$ are the decision variables (one for each agent), and each $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is the cost function associated to x_i. Each variable x_i must satisfy individual constraints $x_i \in X_i$, where $X_i \subset \mathbb{R}^{n_i}$ is a closed bounded set that can be nonconvex. Moreover, the variables are intertwined by means of $S \in \mathbb{N}$ coupling constraints $\sum_{i=1}^{N} g_i(x_i) \le b$, where each function $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^S$ is used to model the contribution of x_i to the coupling constraints. Again, we focus on large-scale instances of Problem (3.33) where the number of agents is considerably larger than the number of coupling constraints, i.e. $N \gg S$, which is a challenging scenario in distributed control applications. We make the following standing assumption.[4]

Assumption 3.5. *Problem* (3.33) *is feasible. Moreover, for all $i \in \{1,\ldots,N\}$, the set X_i is compact and the function f_i and each component of g_i are convex.* $\triangle$

Consistently with the previous derivations, throughout this section we assume that each agent i knows only its local constraint X_i, its local cost function f_i and its own contribution g_i to the coupling constraints. As before, we assume agents communicate according to a connected and undirected graph (cf. Section 3.2.1).

Let us now consider a convex approximation of Problem (3.33) that plays the role of

[4]It is left to future investigation to determine whether this assumption can be relaxed to the case in which also f_i and g_i are nonconvex. In principle, the analysis performed in this section should be extended to the case in which f_i and g_i are nonconvex by using their convex closure and by following the arguments of [8].

Problem (3.2) for MILPs. The convex approximation reads

$$\begin{aligned}
\min_{x_1,\dots,x_N} \quad & \sum_{i=1}^{N} f_i(x_i) \\
\text{subj. to} \quad & \sum_{i=1}^{N} g_i(x_i) \le b - \sigma \\
& x_i \in \text{conv}(X_i), \qquad i = 1,\dots,N,
\end{aligned} \tag{3.34}$$

where we recall that $\text{conv}(X_i)$ denotes the convex hull of X_i and $\sigma \ge 0$ is the restriction vector. Note that, differently from the mixed-integer case in which the convex relaxation is a linear program, here we cannot assess any strong duality property on Problem (3.34) (indeed in Section 3.5.5 we will assume it explicitly).

As in Section 3.2.2, under the assumption of uniqueness, the optimal solution of the convex problem (3.34) satisfies the nonconvex constraints $x_i \in X_i$ for most indices i. The following proposition formalizes this fact.

Proposition 3.3. *Let Assumption 3.5 hold and let Problem* (3.34) *be feasible with unique optimal solution* $(x_1^\star,\dots,x_N^\star)$. *Then, there exists an index set* $I \subset \{1,\dots,N\}$, *with cardinality* $|I| \ge N - S - 1$, *such that* $x_i^\star \in X_i$ *for all* $i \in I$. △

A proof of this result is indirectly provided by [8, Proposition 5.26], however, in Section 3.6.4 we provide a self-contained proof for completeness. Differently from the mixed-integer case, for the general nonconvex problem (3.33) the tightest bound that we can obtain is $|I| \ge N - S - 1$ (we recall that in the mixed-integer case we had $|I_{\mathbb{Z}}| \ge N - S$). Note that optimal control problems of the type (1.4) with nonlinear dynamics and positive definite quadratic costs satisfy Assumption 3.5 and have unique optimal solution as required by (3.3).

Before introducing the solution approach for nonconvex problems, let us recall how the primal decomposition reads for Problem (3.34). Formally, the master problem is

$$\begin{aligned}
\min_{y_1,\dots,y_N} \quad & \sum_{i=1}^{N} p_i(y_i) \\
\text{subj. to} \quad & \sum_{i=1}^{N} y_i = b - \sigma \\
& y_i \in Y_i, \qquad i = 1,\dots,N,
\end{aligned} \tag{3.35}$$

where, for all $i \in \{1,\dots,N\}$, $p_i : Y_i \to \mathbb{R}$ is the function associating each local allocation

$y_i \in \mathbb{R}^S$ to the optimal cost of the corresponding subproblem,

$$\begin{aligned} p_i(y_i) = \min_{x_i} \; & f_i(x_i) \\ \text{subj. to } \; & g_i(x_i) \le y_i \\ & x_i \in \text{conv}(X_i), \end{aligned} \tag{3.36}$$

and $Y_i \subseteq \mathbb{R}^S$ is the set of y_i for which Problem (3.36) is feasible. Clearly, Lemma 3.1 holds true also for Problem (3.34) and Problems (3.36)–(3.35).

3.5.2 Solution Approach for Nonconvex Problems

Let us now extend the framework introduced in Section 3.3.2 to the nonconvex set-up (3.33). The method introduced here will be used as a building block for the distributed algorithm in Section 3.5.4. As for the mixed-integer case, the leading idea is to solve the convex problem (3.34) and then to run a local correction procedure (executed locally by each agent) to change only the blocks of optimal solution that do not *already* satisfy the local nonconvex constraints. Since the corrected solution may result into a violation of the coupling constraints $\sum_{i=1}^{N} g_i(x_i) \le b - \sigma$, in the next section we discuss how to choose the a-priori restriction σ appropriately to ensure that $\sum_{i=1}^{N} g_i(x_i) \le b$.

Let us now formalize the local procedure to correct the solution of Problem (3.34). Let $(y_1^\star, \dots, y_N^\star)$ denote an optimal solution of (3.35). As done in Section 3.3.1, we assume that each agent i is provided with an allocation vector y_i^t that asymptotically goes to $y_i^\star$. However, since the agents can only perform a limited number of iterations, we denote by $y_i^{\text{END}} \in \mathbb{R}^S$ the actual allocation vector obtained by each agent i at the end of distributed algorithm (to be formalized soon). If y_i^{END} was equal to $y_i^\star$, by solving Problem (3.36), at least $N - S - 1$ agents would obtain a vector sayisfying the nonconvex constraints $x_i \in X_i$ (cf. Proposition 3.3). However, in practice y_i^{END} will only be an approximation of $y_i^\star$ so that all the agents are required to execute the following local procedure to ensure satisfaction of the local nonconvex constraints. The algorithm is summarized in Procedure 6.

Essentially, Procedure 6 is the counterpart of Problem (3.7) for nonconvex problems. We do not give detailed comments on the procedure and we refer the reader to Section 3.3.2 for a more comprehensive discussion. One can note that Procedure 6 has a structure that is slightly different from the one outlined in Section 3.3.2 because it avoids nonconvex problems whenever possible (since global optimal solutions may be hard to obtain), however the conceptual idea is the same. The procedure always yields a vector satisfying (by construction) the local nonconvex constraints $x_i \in X_i$. Similarly to

Procedure 6 Get-Nonconvex-Sol

```
Input: $i$-th allocation $y_i^{\text{END}}$
Compute $x_i^{\text{CONV}}$ as optimal solution of (3.36) with $y_i^{\text{END}}$
If $x_i^{\text{CONV}} \in X_i$ then output $x_i^{\text{OUT}} = x_i^{\text{CONV}}$
Else
    Compute $x_i^{\text{NC}}$ as a feasible solution of
        $\min_{x_i} f_i(x_i)$
        subj. to $g_i(x_i) \le y_i^{\text{END}}$      (3.37)
        $x_i \in X_i$
    If (3.37) is feasible then output $x_i^{\text{OUT}} = x_i^{\text{NC}}$
    Else output $x_i^{\text{OUT}} = x_i^{\text{VIOL}}$ as a feasible solution of
        $\min_{x_i} f_i(x_i)$
        subj. to $g_i(x_i) \le y_i^{\text{END}} + v_i \mathbf{1}$      (3.38)
        $x_i \in X_i$
        with minimal violation $v_i > 0$
    End If
End If
```

the mixed-integer case, in order to compute v_i agents can solve the problem

$$\begin{aligned} \min_{x_i, v_i} \;& v_i \\ \text{subj. to } \;& g_i(x_i) \le y_i^{\text{END}} + v_i \mathbf{1} \\ & x_i \in X_i. \end{aligned}$$

A global optimal solution of the nonconvex problems (3.37) and (3.38) is desirable to improve the overall cost, however a feasible solution is sufficient for the distributed algorithm to produce a feasible solution to the original problem.

3.5.3 Restriction Vector and Preliminary Analysis

In order to make sure the overall solution $(x_1^{\text{OUT}}, \dots, x_N^{\text{OUT}})$ is feasible for the original Problem (3.33) we need to design the restriction vector σ appropriately. We now report the computation steps of σ, which should be compared to those described in Section 3.3.3.

For all $i \in \{1, \dots, N\}$, let us define a vector ℓ_i, representing the resource lower bound

$$\ell_i \triangleq \min_{x_i \in \text{conv}(X_i)} g_i(x_i),$$

where $\min$ is intended component wise. Then, let us define v_i^{MAX} as the optimal cost of

$$\begin{aligned} v_i^{\text{MAX}} \triangleq \min_{x_i, v_i} \; & v_i \\ \text{subj. to } \; & x_i \in X_i, \;\; v_i \geq 0 \\ & g_i(x_i) \leq \ell_i + v_i \mathbf{1} \end{aligned} \tag{3.39}$$

By construction, v_i^{MAX} represents the maximum violation required by agent i to compute a feasible vector in X_i. A local minimum of (3.39) is sufficient for the forthcoming analysis, however a global minimum results in a less conservative restriction. A restriction vector σ, representing the worst-case overall violation, is

$$\sigma^\infty = (S+1) \cdot \max_{i \in \{1,\ldots,N\}} v_i^{\text{MAX}} \, \mathbf{1} \tag{3.40}$$

where the coefficient $S+1$ is due to the maximum number of agents that can simultaneously violate (by Proposition 3.3). We point out that, as done in Section 3.3.3, one can define a saturated version of (3.40) to reduce possible conservativeness of the solution (this extension follows the same arguments of the mixed-integer case and will not be discussed here). Once again, note that σ^∞ can be computed in a distributed way by using a $\max$-consensus.

To conclude this section, we give a preliminary analysis of the framework introduced so far. To this end, let us consider the fictious case in which the local procedure is called with the optimal solution of Problem (3.35) (the general case is discussed in Section 3.5.5). In order to apply Proposition 3.3 to Problem (3.34), we make the following assumption.

Assumption 3.6. *Problem* (3.34) *is feasible and its optimal solution is unique.* △

The following result extends Theorem 3.1 to the nonconvex setting.

Theorem 3.6. *Let Assumptions 3.5 and 3.6 hold and let* $(y_1^\star, \ldots, y_N^\star)$ *be an optimal solution of Problem* (3.35)*, with* σ *equal to* (3.40)*. Let* $x^{\text{OUT}} = (x_1^{\text{OUT}}, \ldots, x_N^{\text{OUT}})$*, where each* x_i^{OUT} *is the output of* Get-Nonconvex-Sol *with input* $y_i^\star$*. Then,* x^{OUT} *is feasible for the original problem* (3.33)*.*

The proof is provided in Section 3.6.4.

Theorem 3.6 guarantees that, asymptotically, the solution computed by Get-Nonconvex-Sol is feasible for Problem (3.33). However, in general, there might be a certain degree of suboptimality that can be estimated by repeating the same reasonings of Theorem 3.2.

3.5.4 Distributed Algorithm Description

Let us finally formalize our distributed algorithm to compute a feasible solution to the nonconvex problem (3.33). The algorithm is obtained by integrating the framework of

Section 3.5.2 with the Distributed Primal Decomposition algorithm for convex problems discussed in Section 2.2.5.

Let $t \in \mathbb{N}$ denote the iteration index and let Get-Nonconvex-Sol(y_i^t) be the output of the local procedure of Section 3.5.2 with input equal to y_i^t, the i-th allocation at time t. In the following table we summarize our Distributed Primal Decomposition for Nonconvex problems from the perspective of agent i. As regards the initial coordination to run Algorithm 4, similar reasonings as in Remark 3.2 hold.

Algorithm 7 Distributed Primal Decomposition for Nonconvex problems

```
**Initialization**: set $T_f > 0$ and $y_i^0$ such that $\sum_{i=1}^N y_i^0 = b - \sigma$
**Repeat** for $t = 0, 1, \dots, T_f - 1$
    **Compute** $\mu_i^t$ as a Lagrange multiplier of
        $$\begin{aligned} \min_{x_i,\rho_i} \quad & f_i(x_i) + M\rho_i \\ \text{subj. to} \quad \mu_i : \; & g_i(x_i) \le y_i^t + \rho_i \mathbf{1} \\ & x_i \in \text{conv}(X_i), \;\; \rho_i \ge 0 \end{aligned} \tag{3.41}$$
    **Receive** $\mu_j^t$ from $j \in \mathcal{N}_i$ and update
        $$y_i^{t+1} = y_i^t + \alpha^t \sum_{j \in \mathcal{N}_i} \big( \mu_i^t - \mu_j^t \big) \tag{3.42}$$
**Set** $y_i^{\text{END}} = y_i^{T_f}$ **and return** $x_i^{\text{OUT}} = $ Get-Nonconvex-Sol$(y_i^{\text{END}})$
```

The algorithm is fully distributed, in the sense that at every iteration t the computation performed by each agent i involves only local information and the information gathered from its neighbors to perform (3.42). The initial allocations y_i^0 must initialized such that $\sum_{i=1}^N y_i^0 = b - \sigma$, e.g. with $y_i^0 = (b - \sigma)/N$. A remarkable property of Algorithm 7 is that it only requires the solution of convex problems in order to evolve (indeed Problem (3.41) is convex), while, as already noted in Section 3.5.2, nonconvex problems in Get-Nonconvex-Sol can also be solved suboptimally.

3.5.5 Algorithm Analysis

In this section we provide a theoretical analysis of Algorithm 7 as an extension of the results provided in Section 3.3.5. As in Section 3.3.5, we assume that the total restriction is equal to

$$\sigma^{\text{FT}} = \sigma^{\infty} + \delta \mathbf{1}, \tag{3.43}$$

where $\delta > 0$ is an arbitrarily small number. As already mentioned, we need to ensure that strong duality holds for the convex approximation (3.34), thus we make the following assumption.

Assumption 3.7 (Slater). *There exist vectors $\bar{x}_1 \in \text{conv}(X_1), \ldots, \bar{x}_N \in \text{conv}(X_N)$ such that $\sum_{i=1}^N g_i(\bar{x}_i) < b - \sigma^{\text{FT}}$.* $\triangle$

Under the preceding assumption and using the step-size Assumption 3.3 we are able to prove the following theorem, in which we assess that in finite time the solution sequence computed by DiP-Nonconvex is feasible for the original problem (3.33).

Theorem 3.7. *Let $\sigma = \sigma^{\text{FT}}$ for an arbitrary $\delta > 0$. Let Assumptions 3.3, 3.5, 3.6 and 3.7 hold. Moreover, let the local allocation vectors y_i^0 be initialized such that $\sum_{i=1}^N y_i^0 = b - \sigma^{\text{FT}}$. Then, there exists a sufficiently large $M > 0$ and $T_\delta > 0$ for which Algorithm 7 generates a sequence $\{x_1^t, \ldots, x_N^t\}_{t \geq 0}$ such that the vector $(x_1^t, \ldots, x_N^t)$ is a feasible solution for Problem* (3.33) *for all $t \geq T_\delta$.*

The proof is provided in Section 3.6.4.

Finite-time feasibility of Algorithm 7 is an appealing feature for model predictive control applications, since it can ensure recursive feasibility of the control algorithm. As for the parameter M, it must be greater than $\|\mu^\star\|_1$, where $\mu^\star$ denotes a dual optimal solution of Problem (3.34) (see Section 2.2.3). In practice, it suffices to choose M sufficiently large.

3.5.6 Numerical Example

In this section, we provide numerical computations performed with the Matlab software to corroborate the theoretical results and to highlight the main features of our algorithm. We consider a simplified scenario in which we are able to express $\text{conv}(X_i)$ explicitly.

Formally, consider a network of $N = 50$ agents, whose aim is to cooperatively find a feasible solution to an optimal control of the type

$$\begin{aligned} \min_{\{z_i(k+1), u_i(k)\}_{k,i}} \quad & \sum_{i=1}^N \sum_{k=0}^{K-1} \ell_i\big(z_i(k), u_i(k)\big) + V_i\big(z_i(K)\big) \\ \text{subj. to} \quad & z_i(k+1) = h_i(z_i(k), u_i(k)), \qquad \forall\, k, i \\ & z_i(k+1) \in \mathcal{Z}_i, \;\; u_i(k) \in \mathcal{U}_i, \qquad \forall\, k, i \\ & \sum_{i=1}^N \big(Z_i z_i(k) + U_i u_i(k)\big) \leq b, \qquad \forall\, k, \end{aligned} \tag{3.44}$$

where, for all i, z_i denotes the system state, u_i is the system input and h_i is the function representing the system dynamics. Clearly, Problem (3.44) can be re-mapped to

Problem (3.33) by suitably defining all the symbols. For simplicity, we consider 1-step predictions of the dynamics ($K = 1$). Each dynamical system i has 1-dimensional state and input, with dynamics $z_i(k+1) = z_i(k)^2 + q_i u_i(k)^2 + r_i$, where the parameters q_i and r_i are randomly drawn from $[1, 5]$ and $[-4, 0]$ respectively. The state and input constraints $\mathcal{Z}_i$ and $\mathcal{U}_i$ are box constraints (i.e. $z_i^{\text{LB}} \leq z_i(k) \leq z_i^{\text{UB}}$ and similarly for the set $\mathcal{U}_i$) where the lower and upper bounds have entries in $[-10, -5]$ and $[5, 10]$ respectively. We assume that the systems are initialized in the origin, i.e. $z_i(0) = 0$ for all i. Therefore, the local nonconvex feasible set X_i is a clipped parabola in $\mathbb{R}^2$, and $\text{conv}(X_i)$ can be obtained by replacing the dynamics constraints with the inequality version $z_i(k+1) \geq q_i u_i(k)^2 + r_i$. The agents must further satisfy $S = 3$ coupling constraints, where the matrices Z_i and U_i have entries in $[0, 1]$ and the vector b has entries in $[-3, 7]$. As for the cost functions, we assume that $\ell_i(z_i, u_i)$ and $V_i(z_i)$ are linear with random entries in $[-5, 5]$.

The communication graph is a random Erdős-Rényi graph with edge probability 0.2. A random problem has been generated, and a local minimum has been found using a centralized solver (fmincon). In order to check whether the instance is meaningful, we make sure it has a duality gap by solving the dual problem with a dual subgradient algorithm. We perform a simulation of the distributed algorithm with $\delta = 1$. The restriction σ^{FT} has infinity norm equal to 3.5, and agents computed in finite time a feasible solution to the nonconvex problem (3.44) (as expected from Theorem 3.7). In Figure 3.10 the distributed utilization of the coupling constraints is shown, where x_i^t denotes the stack of all the local optimization variables of each agent i, obtained as the output of Get-Nonconvex-Sol with allocation equal to y_i^t, and $g_i(x_i) = U_i u_i(0) + Z_i z_i(1)$.

Notably, the solution is feasible since the first iteration of the distributed algorithm and has 19% suboptimality with respect to the solution computed by fmincon.

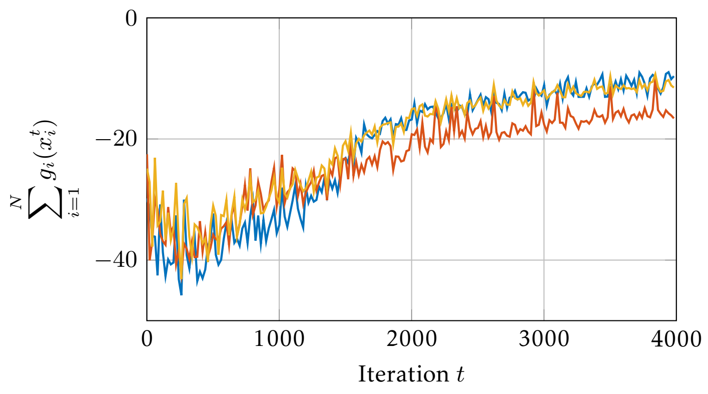


Figure 3.10: Evolution of the coupling constraint utilization (which has $S = 3$ components). The solution computed by the algorithm is feasible for the coupling constraints since the first iteration, indeed the maximum value is below 0 in the whole graph.

3.6 Proofs

3.6.1 Proofs for Section 3.2

Proof of Proposition 3.1

We discuss the case $\sigma = \mathbf{0}$ (the proof is independent of the value of σ and assumes only feasibility of Problem (3.2)). By following similar arguments as in [121, Theorem 2.5], Problem (3.2) can be equivalently reformulated by expressing the local constraints in terms of their vertices, i.e.

$$\begin{aligned} \min_{p} \quad & \sum_{i=1}^{N} \sum_{j \in \mathcal{J}_i} p_i^j c_i^\top x_i^j \\ \text{subj. to} \quad & \sum_{i=1}^{N} \sum_{j \in \mathcal{J}_i} p_i^j A_i x_i^j \leq b \\ & \sum_{j \in \mathcal{J}_i} p_i^j = 1, \qquad i = 1, \dots, N \\ & p \geq \mathbf{0}, \end{aligned} \tag{3.45}$$

where, for all $i \in \{1, \dots, N\}$, $\mathcal{J}_i \triangleq \{1, \dots, |\text{vert}(X_i^{\text{MILP}})|\}$ is the set of indices of the elements in $\text{vert}(X_i^{\text{MILP}})$ and x_i^j denotes the j-th element of $\text{vert}(X_i^{\text{MILP}})$, for all $j \in \mathcal{J}_i$. For any feasible vector z of Problem (3.2) there exists a feasible vector p of Problem (3.45) such that

$$z_i = \sum_{j \in \mathcal{J}_i} p_i^j x_i^j \quad \text{for all } i \in \{1, \dots, N\}. \tag{3.46}$$

Let us manipulate (3.45). By introducing positive slack variables $q \in \mathbb{R}^S$, the coupling constraint can be transformed to equality constraint, i.e.

$$\begin{aligned} \min_{p,q} \quad & \sum_{i=1}^{N} \sum_{j \in \mathcal{J}_i} p_i^j c_i^\top x_i^j \\ \text{subj. to} \quad & \sum_{i=1}^{N} \sum_{j \in \mathcal{J}_i} p_i^j A_i x_i^j + q = b \\ & \sum_{j \in \mathcal{J}_i} p_i^j = 1, \qquad i = 1, \dots, N \\ & p, q \geq \mathbf{0}. \end{aligned} \tag{3.47}$$

Problem (3.47) is an LP in standard form, with positivity constraints on all the variables and $N + S$ equality constraints. Now, let us consider a vertex $(\bar{z}_1, \dots, \bar{z}_N)$ of the feasible set of (3.2), or, equivalently, a vertex of the feasible set of (3.47), denoted by $(p^\star, q^\star)$.

The number of nonzero entries of $p^\star$ is equal to $\text{supp}(p^\star) = \sum_{i=1}^N \text{supp}(p_i^\star)$. Being $(p^\star, q^\star)$ a basic solution, it has at most $N+S$ nonzero entries. In symbols, $\text{supp}(p^\star) \leq \text{supp}((p^\star, q^\star)) \leq N+S$. Since by construction $\text{supp}(p_i^\star) \geq 1$ for all i, it follows that $\text{supp}(p^\star) \geq N$. Summarizing, it holds

$$N \leq \sum_{i=1}^N \text{supp}(p_i^\star) \leq N+S, \qquad \text{with } \text{supp}(p_i^\star) \geq 1 \ \forall i.$$

Then, for only a maximum of S indices, it is possible that $\text{supp}(p_i^\star) > 1$. Let $I_\mathbb{Z}$ denote the set of indices for which this does not hold (i.e. indices i such that $\text{supp}(p_i^\star) = 1$). Then, $|I_\mathbb{Z}| \geq N - S$. For all $i \in I_\mathbb{Z}$, by (3.46), $\text{supp}(p_i^\star) = 1$ implies $\bar{z}_i \in X_i^{\text{MILP}}$. ■

3.6.2 Proofs for Section 3.3

Proof of Theorem 3.1

For the sake of analysis, let us denote by $(z_1^\star, \ldots, z_N^\star)$ the optimal solution of the restricted LP (3.2). By Assumption 3.1, $(z_1^\star, \ldots, z_N^\star)$ is a vertex, so that by Proposition 3.1 there exists $I_\mathbb{Z} \subseteq \{1, \ldots, N\}$, with $|I_\mathbb{Z}| \geq N - S$, such that $z_i^\star \in X_i^{\text{MILP}}$ for all $i \in I_\mathbb{Z}$. By Lemma 3.1, $z_i^\star$ is an optimal solution of Problem (3.4), with $y_i = y_i^\star$, for all $i \in \{1, \ldots, N\}$. Thus, for all $i \in I_\mathbb{Z}$, $z_i^\star \in X_i^{\text{MILP}}$ is the optimal solution of (3.7) with $y_i^t = y_i^\star$. Then, it holds $A_i x_i^\infty \leq y_i^\star$ for all $i \in I_\mathbb{Z}$.

Let us focus on the set $I_\mathbb{R} = \{1, \ldots, N\} \setminus I_\mathbb{Z}$, which contains indices such that $z_i^\star \notin X_i^{\text{MILP}}$. Let us further partition $I_\mathbb{R} = I_{\text{FEAS}} \cup I_{\text{INFEAS}}$, where the indices collected in I_{FEAS} correspond to feasible subproblems (3.4), from which it follows that $A_i x_i^\infty \leq y_i^\star$ for all $i \in I_{\text{FEAS}}$, while the remaining index set I_{INFEAS} corresponds to infeasible subproblems (3.4). We have

$$A_i x_i^\infty \overset{(a)}{\leq} y_i^\star + v_i^\infty \mathbf{1} \overset{(b)}{\leq} y_i^\star + v_i^{\text{MAX}} \mathbf{1}, \qquad \forall\, i \in I_{\text{INFEAS}},$$

where *(a)* follows by construction of x_i^∞ and *(b)* follows since any optimal solution of Problem (3.13), say x_i^{L}, is feasible for Problem (3.9) (since $A_i x_i^{\text{L}} \leq \ell_i + v_i^{\text{MAX}} \leq y_i^\star + v_i^{\text{MAX}}$), from which it follows that $v_i^\infty \leq v_i^{\text{MAX}}$ (by optimality). Also, notice that, since $x_i^\infty \in X_i^{\text{MILP}}$ and X_i^{MILP} is compact, then $A_i x_i^\infty \leq \max\limits_{x_i \in X_i^{\text{MILP}}} A_i x_i$ and it holds

$$A_i x_i^\infty - y_i^\star \leq \max_{x_i \in X_i^{\text{MILP}}} A_i x_i - y_i^\star \leq \max_{x_i \in X_i^{\text{MILP}}} A_i x_i - \ell_i,$$

where $\max$ is intended component wise. Thus, we have shown that

$$A_i x_i^\infty - y_i^\star \leq \min \Big\{ v_i^{\text{MAX}} \mathbf{1}, \ \max_{x_i \in X_i^{\text{MILP}}} (A_i x_i - \ell_i) \Big\}.$$

It follows that $A_i x_i^\infty \le y_i^\star + \sigma_i^{\text{LOC}}$ for all $i \in I_{\text{INFEAS}}$. By summing over $i \in I_{\text{INFEAS}}$ the term σ_i^{LOC}, we obtain

$$\sum_{i \in I_{\text{INFEAS}}} \sigma_i^{\text{LOC}} \le |I_{\text{INFEAS}}| \max_{i \in I_{\text{INFEAS}}} \sigma_i^{\text{LOC}} \le S \max_{i \in \{1,\dots,N\}} \sigma_i^{\text{LOC}} = \sigma^\infty, \tag{3.48}$$

where max is intended component wise. Collecting the previous conditions leads to

$$\begin{aligned}\sum_{i=1}^N A_i x_i^\infty &= \sum_{i \in I_{\mathbb{Z}}} A_i x_i^\infty + \sum_{i \in I_{\text{FEAS}}} A_i x_i^\infty + \sum_{i \in I_{\text{INFEAS}}} A_i x_i^\infty \\ &\le \sum_{i=1}^N y_i^\star + \sum_{i \in I_{\text{INFEAS}}} \sigma_i^{\text{LOC}} \\ &\le b - \sigma^\infty + \sigma^\infty = b,\end{aligned}$$

where we used $\sum_{i=1}^N y_i^\star = b - \sigma^\infty$. The proof follows. ■

Proof of Theorem 3.2

Following the ideas in [121, Theorem 3.3], let us split the bound in three terms

$$\sum_{i=1}^N c_i^\top x_i^\infty - J^{\text{MILP}} = \sum_{i=1}^N (c_i^\top x_i^\infty - c_i^\top z_i^\star) + \Big(\sum_{i=1}^N c_i^\top z_i^\star - J^{\text{LP}}\Big) + \big(J^{\text{LP}} - J^{\text{MILP}}\big),$$

where $(z_1^\star, \dots, z_N^\star)$ is the optimal solution of Problem (3.2) and J^{LP} denotes the optimal cost of Problem (3.2) with $\sigma = \mathbf{0}$. Next, we analyze each term independently.

(i) $\sum_{i=1}^N (c_i^\top x_i^\infty - c_i^\top z_i^\star)$. By Proposition 3.1, there exists $I_{\mathbb{Z}}$, with $|I_{\mathbb{Z}}| \ge N - S$, such that $z_i^\star \in X_i^{\text{MILP}}$ for all $i \in I_{\mathbb{Z}}$. Thus, for $i \in I_{\mathbb{Z}}$, it holds $x_i^\infty = z_i^\star$, implying $c_i^\top x_i^\infty - c_i^\top z_i^\star = 0$. Therefore, by defining $I_{\mathbb{R}} \triangleq \{1, \dots, N\} \setminus I_{\mathbb{Z}}$, the sum reduces to

$$\sum_{i=1}^N (c_i^\top x_i^\infty - c_i^\top z_i^\star) = \sum_{i \in I_{\mathbb{R}}} (c_i^\top x_i^\infty - c_i^\top z_i^\star), \tag{3.49}$$

with $|I_{\mathbb{R}}| \le S$. Since $c_i^\top x_i^\infty \le \max_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i$ and $\min_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i \le c_i^\top z_i^\star$, it follows that

$$\sum_{i=1}^N (c_i^\top x_i^\infty - c_i^\top z_i^\star) \le \sum_{i \in I_{\mathbb{R}}} \gamma_i \le S \max_{i \in \{1,\dots,N\}} \gamma_i.$$

(ii) $\sum_{i=1}^N c_i^\top z_i^\star - J^{\text{LP}}$. By following similar arguments to [121, Theorem 3.3], one can show that

$$\sum_{i=1}^{N} c_i^\top z_i^\star - J^{\text{LP}} \le \frac{\|\sigma^\infty\|_\infty}{\zeta} \sum_{i=1}^{N} (c_i^\top \widehat{z}_i - c_i^\top z_i^\star), \tag{3.50}$$

where $(\widehat{z}_1, \ldots, \widehat{z}_N)$ is any Slater point (cf. Assumption 3.2). Since $c_i^\top \widehat{z}_i \le \max_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i$ and $\min_{x_i \in X_i^{\text{MILP}}} c_i^\top x_i \le c_i^\top z_i^\star$, it follows that

$$\sum_{i=1}^{N} c_i^\top z_i^\star - J^{\text{LP}} \le \frac{\|\sigma^\infty\|_\infty}{\zeta} \sum_{i=1}^{N} \gamma_i \le \frac{N\|\sigma^\infty\|_\infty}{\zeta} \max_{i \in \{1,\ldots,N\}} \gamma_i.$$

(iii) $J^{\text{LP}} - J^{\text{MILP}}$. Being J^{LP} the cost of (3.2) with $\sigma = 0$, which is a relaxed version of (3.1), then $J^{\text{LP}} - J^{\text{MILP}} \le 0$. Combining the results above, the bound follows. ■

Proof of Theorem 3.3

Let $\{y_1^t, \ldots, y_N^t\}_{t \ge 0}$ denote the allocation vector sequence generated by Algorithm 3. By Proposition 2.1, the sequence $\{y_1^t, \ldots, y_N^t\}_{t \ge 0}$ converges to an optimal solution $(y_1^\star, \ldots, y_N^\star)$ of Problem (3.3) with $\sigma = \sigma^\infty + \delta \mathbf{1}$. Thus, for all $i \in \{1, \ldots, N\}$ and $\epsilon_i > 0$, there exists $T_{\epsilon_i} > 0$ such that $t \ge T_{\epsilon_i} \Rightarrow \|y_i^t - y_i^\star\|_\infty \le \epsilon_i$. If we let $T = \max_{i \in \{1,\ldots,N\}} T_{\epsilon_i}$, then $y_i^t \le y_i^\star + \epsilon_i \mathbf{1}$ for all $t \ge T$ and $i \in \{1, \ldots, N\}$.

To prove the statement, we compare the state of the algorithm at an iteration $t \ge T_\delta$ and the quantities that would be computed at infinity, for all $i \in \{1, \ldots, N\}$. To this end, let us denote by (v_i^∞, x_i^∞) the optimal solution of Problem (3.7) with $y_i^t = y_i^\star$ (we discard the ξ_i part of the solution). As shown in Section 3.3.2, v_i^t is the optimal cost of

$$\begin{aligned} \min_{v_i, x_i} \;& v_i \\ \text{subj. to } \;& A_i x_i \le y_i^t + v_i \mathbf{1} \\ & x_i \in X_i^{\text{MILP}}, \;\; v_i \ge 0. \end{aligned} \tag{3.51}$$

Note that the pair $(\epsilon_i + v_i^\infty, x_i^\infty)$ is feasible for Problem (3.51) for all $t \ge T$. Indeed, it holds $x_i^\infty \in X_i$, $\epsilon_i + v_i^\infty \ge 0$, and moreover $A_i x_i^\infty \le y_i^\star + v_i^\infty \mathbf{1} \le y_i^t + \epsilon_i \mathbf{1} + v_i^\infty \mathbf{1}$ for all $t \ge T$. Being v_i^t the optimal cost of (3.51), we have

$$v_i^t \le \epsilon_i + v_i^\infty, \qquad \forall t \ge T. \tag{3.52}$$

We now follow arguments similar to the proof of Theorem 3.1. For all $i \in \{1, \ldots, N\}$, let $z_i^\star$ denote the optimal solution of Problem (3.2). For the sake of analysis, let us split the agent set $\{1, \ldots, N\}$ as $I_{\mathbb{Z}} \cup I_{\text{FEAS}} \cup I_{\text{INFEAS}}$, where $I_{\mathbb{Z}}$ contains agents for which $z_i^\star \in X_i^{\text{MILP}}$, I_{FEAS} contains agents for which $z_i^\star \notin X_i^{\text{MILP}}$ and $v_i^\infty = 0$, and I_{INFEAS} contains agents for which $z_i^\star \notin X_i^{\text{MILP}}$ and $v_i^\infty > 0$. Using the same arguments of Theorem 3.1, for all $i \in I_{\mathbb{Z}}$ it holds $v_i^\infty = 0$. Consider the agents $i \in I_{\mathbb{Z}} \cup I_{\text{FEAS}}$. By construction, it holds

$$A_i x_i^t \le y_i^t + v_i^t \le y_i^t + \epsilon_i, \qquad \forall\, i \in I_{\mathbb{Z}} \cup I_{\text{FEAS}} \text{ and } t \ge T, \tag{3.53}$$

where we used (3.52) and $v_i^\infty = 0$. As for the agents $i \in I_{\text{INFEAS}}$, again by (3.52), it holds $A_i x_i^t - y_i^t \le v_i^t \mathbf{1} \le v_i^\infty \mathbf{1} + \epsilon_i \mathbf{1}$ for all $t \ge T$, or equivalently

$$A_i x_i^t - y_i^t - \epsilon_i \mathbf{1} \le v_i^\infty \mathbf{1}, \qquad \forall t \ge T_\delta.$$

Moreover, note that, for $t \ge T_\delta$, it holds $A_i x_i^t - y_i^t - \epsilon_i \mathbf{1} \le A_i x_i^t - y_i^t \le \max_{x_i \in X_i^{\text{MILP}}} A_i x_i - \ell_i$, where $\max$ is intended component wise. Using the definition of σ_i^{LOC} in Section 3.3.3 and rearranging the terms, we obtain

$$A_i x_i^t \le y_i^t + \sigma_i^{\text{LOC}} + \epsilon_i \mathbf{1}, \quad \forall\, i \in I_{\text{INFEAS}} \text{ and } t \ge T. \tag{3.54}$$

Finally, by using (3.53) and (3.54), we can write

$$\begin{aligned}\sum_{i=1}^N A_i x_i^t &= \sum_{i=1}^N y_i^t + \sum_{i=1}^N \epsilon_i \mathbf{1} + \sum_{i \in I_{\text{INFEAS}}} \sigma_i^{\text{LOC}} \\ &\le b - \sigma^\infty - \delta \mathbf{1} + \sum_{i=1}^N \epsilon_i \mathbf{1} + \sigma^\infty, \qquad \forall t \ge T,\end{aligned}$$

which follows since $\sum_{i=1}^N y_i^t = \sum_{i=1}^N y_i^0 = b - \sigma^\infty - \delta\mathbf{1}$ (cf. Proposition 2.1 (i)) and by (3.48). Since ϵ_i are arbitrary, choosing $\epsilon_i = \delta/N$ for all i implies $\sum_{i=1}^N A_i x_i^t \le b$ for all $t \ge T_\delta \triangleq T$, which concludes the proof. ■

Proof of Theorem 3.4

Let $\{y_1^t, \ldots, y_N^t\}_{t \ge 0}$ denote the allocation vector sequence generated by Algorithm 3. By following similar arguments as in the proof of Theorem 3.3, we conclude that, for fixed $\epsilon_i > 0$, there exists a sufficiently large $T > 0$ such that $\|y_i^\star - y_i^t\|_\infty \le \epsilon_i$ for all $t \ge T$ and $i \in \{1, \ldots, N\}$.

Inspired by Theorem 3.2, let us split the suboptimality bound as $\sum_{i=1}^N c_i^\top x_i^t - J^{\text{MILP}} = \sum_{i=1}^N (c_i^\top x_i^t - J_i^{\text{LP},t}) + \big(\sum_{i=1}^N J_i^{\text{LP},t} - J^{\text{LP},\sigma^{\text{FT}}}\big) + \big(J^{\text{LP},\sigma^{\text{FT}}} - J^{\text{LP}}\big) + \big(J^{\text{LP}} - J^{\text{MILP}}\big)$, where $J^{\text{LP},\sigma^{\text{FT}}}$ denotes the optimal cost of Problem (3.2) with $\sigma = \sigma^{\text{FT}}$. The first term $\sum_{i=1}^N (c_i^\top x_i^t - J_i^{\text{LP},t})$ can be explicitly evaluated. As for the last two terms, by following similar arguments as in Theorem 3.2, we conclude that $\big(J^{\text{LP},\sigma^{\text{FT}}} - J^{\text{LP}}\big) + \big(J^{\text{LP}} - J^{\text{MILP}}\big) \le \Gamma \|\sigma^{\text{FT}}\|_\infty$.

Let us analyze in detail the second term. Notice that $\sum_{i=1}^N J_i^{\text{LP},t}$ is the optimal cost of the aggregate problem solved by the agents at iteration t, i.e.

$$\begin{aligned} \min_{\substack{z_1,\dots,z_N,\\ \rho_1,\dots,\rho_N}} \quad & \sum_{i=1}^{N} (c_i^\top z_i + M\rho_i) \\ \text{subj. to} \quad & A_i z_i \leq y_i^t + \rho_i \mathbf{1}, && \forall\, i, \\ & z_i \in \text{conv}(X_i^{\text{MILP}}), \;\; \rho_i \geq 0, && \forall\, i. \end{aligned} \tag{3.55}$$

Moreover, $J^{\text{LP},\sigma^{\text{FT}}}$ can be seen as the optimal cost of a perturbed version of Problem (3.55) (in particular, the constraints $A_i z_i \leq y_i^t + \rho_i \mathbf{1}$ are perturbed to $A_i z_i \leq y_i^\star + \rho_i \mathbf{1}$). By applying perturbation theory [16], we have for all $t \geq T$

$$\sum_{i=1}^{N} J_i^{\text{LP},t} - J^{\text{LP},\sigma^{\text{FT}}} \leq \sum_{i=1}^{N} \|y_i^\star - y_i^t\|_\infty \|\mu_i^t\|_1 \leq \sum_{i=1}^{N} \epsilon_i \|\mu_i^t\|_1.$$

By choosing $\epsilon_i = \delta/N$ for all $i \in \{1,\dots,N\}$, we finally obtain

$$\sum_{i=1}^{N} J_i^{\text{LP},t} - J^{\text{LP},\sigma^{\text{FT}}} \leq \sum_{i=1}^{N} \epsilon_i \|\mu_i^t\|_1 = \delta \sum_{i=1}^{N} \|\mu_i^t\|_1, \qquad \forall t \geq T_\delta \triangleq \max_{i \in \{1,\dots,N\}} T_{\epsilon_i},$$

and the proof follows. ∎

3.6.3 Proofs for Section 3.4

Proof of Proposition 3.2.

The algorithm is essentially a constraint generation algorithm inspired to the standard method recalled in Section 3.4.1. We first show that steps (3.31)–(3.32) provide in a finite number of steps (K_f) the (unique) optimal vertex of the optimization problem

$$\begin{aligned} \max_{\mu_i, \eta_i} \quad & \eta_i - \bar{y}_i^\top \mu_i \\ \text{subj. to} \quad & \mu_i \geq 0, \;\; \mu_i \leq Q\mathbf{1}, \\ & \eta_i \leq (c_i + A_i^\top \mu_i)^\top \widehat{x}_i^j, && \forall\, j \in \mathcal{J}_i, \end{aligned} \tag{3.56}$$

which is Problem (3.25) with the addition of the constraint $\mu_i \leq Q\mathbf{1}$. Indeed, as a consequence of the assumption $\bar{y}_i \neq A_i \widehat{x}_i^j$ for all $j \in \mathcal{J}_i$, Problem (3.56) and Problem (3.32) admit unique optimal solution (attained at a vertex) for all $k \geq 0$. Then, we prove that the final step of the algorithm does provide the optimal vertex or an unbounded extreme ray of Problem (3.25).

Let us prove the first part. At the generic time step k the current solution estimate is (μ_i^k, η_i^k) and the value of μ_i^k is used to formulate Problem (3.31). First, note that the constraints (3.32d) are drawn from the constraints of Problem (3.56) (indeed, the vectors

x_i^k belong to vert(X_i^{MILP}) by construction). Moreover, if the termination test at time k fails, then either $\eta_i^k > (c_i + A_i^\top \mu_i^k)^\top x_i^k$ or $\eta_i^k < (c_i + A_i^\top \mu_i^k)^\top x_i^k$. The latter case is not possible because otherwise we would have

$$\eta_i^k < \min_{x_i \in \text{vert}(X_i^{\text{MILP}})} (c_i + A_i^\top \mu_i^k) x_i \le (c_i + A_i^\top \mu_i^k)^\top x_i^\ell, \qquad \ell = 0, \ldots, k-1,$$

contradicting the fact that (μ_i^k, η_i^k) is the optimal solution at iteration $k-1$ (indeed for sufficiently small $\epsilon > 0$ the vector $(\mu_i^k, \eta_i^k + \epsilon)$ would be feasible and with better objective value). Thus, when the termination fails the constraint $\eta_i \le (c_i + A_i^\top \mu_i)^\top x_i^k$ is violated by the current solution estimate (μ_i^k, η_i^k) and will be added to Problem (3.32). Now let us show that the final solution estimate is an optimal solution of Problem (3.56). At the final iteration the termination test $\eta_i^k = (c_i + A_i^\top \mu_i^k)^\top x_i^k$ is successful, from which it follows that

$$\eta_i^k = \min_{x_i \in \text{vert}(X_i^{\text{MILP}})} (c_i + A_i^\top \mu_i^k)^\top x_i \le (c_i + A_i^\top \mu_i^k)^\top \widehat{x}_i^j \qquad \forall j \in \mathcal{J}_i,$$

therefore all the constraints of Problem (3.56) are satisfied and (μ_i^k, η_i^k) is an optimal solution of Problem (3.56). We finally prove that the termination test must be successful after a finite number of iterations. If at iteration k the solution x_i^k of Problem (3.31) is equal to a previously computed x_i^ℓ for some $\ell \in \{1, \ldots, k-1\}$, we have that

$$\eta_i^k \le (c_i + A_i^\top \mu_i^k)^\top x_i^\ell = (c_i + A_i^\top \mu_i^k)^\top x_i^k,$$

where the inequality follows since (μ_i^k, η_i^k) satisfies the constraints of Problem (3.32) at iteration $k-1$. In this case, the termination condition is hit at iteration k. Since the number of possible optimal solutions of Problem (3.31) is finite, it follows that after a finite number of iterations it must hold $x_i^k = x_i^\ell$ for some $\ell \in \{1, \ldots, k-1\}$.

Let us prove the second part. Denote by $(\bar{\mu}_i, \bar{\eta}_i)$ the optimal solution of Problem (3.56), which is computed at the last iteration. Let us consider the two possible outputs of Procedure 5.

(i) If Problem (3.25) has finite optimal cost, by assumption the parameter Q of the bounding box is sufficiently large thus all the vertices of Problem (3.25) are also vertices of Problem (3.56) (including the optimal one). Hence, the condition $\bar{\mu}_i < Q\mathbf{1}$ is satisfied, implying that $(\bar{\mu}_i, \bar{\eta}_i)$ (which is the output of the algorithm) is an optimal vertex of Problem (3.25).

(ii) If Problem (3.25) is unbounded, the optimal vertex $(\bar{\mu}_i, \bar{\eta}_i)$ of Problem (3.56) must be attained on the bounding box, so that the condition $\bar{\mu}_i < Q\mathbf{1}$ does not hold anymore. Thus, by construction, an unbounded ray of Problem (3.32) without the constraint $\mu_i < Q\mathbf{1}$ is an unbounded ray of Problem (3.25). ■

Proof of Lemma 3.2

The line of proof of *(i)* is similar to [84, Theorem IV.3] and other arguments given in [117, Lemma 4.2], but we report the proof here for completeness. We first show convergence in cost and then we show convergence of the solution. At each iteration t the set of constraints of Problem (3.29) contain the constraints in the old basis B_i^t (for which the optimal cost is J_i^t). However, since Problem (3.29) at iteration t contains additional constraints apart from those in B_i^t, the cost sequence is monotonically non-decreasing, i.e.

$$J_i^{t+1} \geq J_i^t, \qquad \text{for all } t \geq 0 \text{ and } i \in \{1, \ldots, N\}, \tag{3.57}$$

and due to the additional bounding box $-R\mathbf{1} \leq \mathbf{y}, \boldsymbol{\nu} \leq R\mathbf{1}$, the sequence $\{J_i^t\}_{t \geq 0}$ is bounded. In view of the discussion in Section 3.4.1, the inequality constraints generated at each iteration by the agents are drawn from the inequality constraints of Problem (3.28), which implies that J_i^t can only assume values in a finite set (in particular, the set of costs associated to all the possible constraint combinations). By combining this fact with (3.57), it follows that J_i^t converges in finite time to $\bar{J}_i \in \mathbb{R}$, i.e. there exists $T_i > 0$ such that $J_i^t = \bar{J}_i$ for all $t \geq T_i$.

To prove convergence of the solution sequence, let us consider the sequence of the first component of $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)$ starting from time instant T_i, which we denote by $\{[\mathbf{y}_i^t]_1\}_{t \geq T_i}$. The sequence is monotonically non-decreasing. Indeed, for all $t \geq T_i$, the local cost value J_i^{t+1} is constant and equal to $\bar{J}_i$, the feasible set is a subset of the feasible set of Problem (3.29) with $H_{\text{TMP}} = B_i^t$, and the solution is constructed by taking into account the lexicographic ordering of vectors. Moreover, since the possible combinations of inequality constraints is finite, the vector $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t)$ can only assume a finite number of values. Therefore, the sequence $\{[\mathbf{y}_i^t]_1\}_{t \geq T_i}$ converges in finite time. By iterating the same arguments on the other components, we conclude that there exists a vector $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$ and a $T_i' > 0$ such that $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t) = (\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$ for all $t \geq T_i'$.

Now we prove *(ii)*. Let us consider the time instants after the solution sequence has converged, i.e. $t > T_i'$. At this point, we have $(\mathbf{y}_i^t, \boldsymbol{\nu}_i^t) = (\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$ for all $t > T_i'$. Assume, to get a contradiction, that the solution at time t violates a constraint $a^\top y_i + b\nu_i \leq f$. Then, by the discussion of Section 3.4.1 and without loss of generality, at the next time step the procedure ConstraintOracle will generate such a constraint because it is violated by the previously computed vector $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$. However, this contradicts the fact that $(\bar{\mathbf{y}}_i, \bar{\boldsymbol{\nu}}_i)$ is an optimal solution at time $t + 1 > T_i'$. ■

3.6.4 Proofs for Section 3.5

Proof of Proposition 3.3

Let us recall two needed lemmas. The following result is the Shapley-Folkman lemma.

Lemma 3.4 ([111])**.** *Let P_i, for $i \in \{1, \dots, N\}$, be a collection of subsets of $\mathbb{R}^d$, and let $v \in \text{conv}(\sum_{i=1}^{N} P_i)$. Then, there exists a subset $I_{\text{CONV}}(v) \subset \{1, \dots, N\}$, containing at most d indices, such that*

$$v \in \bigg(\sum_{i \notin I_{\text{CONV}}(v)} P_i + \sum_{i \in I_{\text{CONV}}(v)} \text{conv}(P_i) \bigg). \qquad \triangle$$

The following result is the well-known Caratheodory's Theorem for representing elements of the convex hull of a set.

Lemma 3.5 ([8])**.** *Let $P \subseteq \mathbb{R}^d$ and consider a point $v \in \text{conv}(P)$. Then, there exists a set $P' \subseteq P$, with $|P'| \leq d+1$, such that $v \in \text{conv}(P')$. Equivalently, v can be expressed as a convex combination of (at most) $d+1$ vectors in P.* $\triangle$

Without loss of generality, assume $\sigma = 0$. For the sake of analysis, let us denote by $f_R^\star$ the optimal cost of Problem (3.34), and let us define the sets

$$Y_i \triangleq \left\{ y_i \;\middle|\; y_i = \begin{bmatrix} g_i(x_i) \\ f_i(x_i) \end{bmatrix}, \, x_i \in X_i \right\} \subset \mathbb{R}^{S+1}, \quad \forall\, i, \tag{3.58}$$

and their Minkowski sum

$$Y \triangleq Y_1 + Y_2 + \ldots + Y_N. \tag{3.59}$$

By Assumption 3.5, Y, $\text{conv}(Y)$, and Y_i, $\text{conv}(Y_i), i \in \{1, \dots, N\}$ are all compact sets. Moreover, by definition of $\text{conv}(Y)$, it holds

$$f_R^\star = \min\{w \mid \text{there exists } (z, w) \in \text{conv}(Y) \text{ with } z \leq b\}.$$

Therefore, consider a vector $(\bar{z}, \bar{w}) \in \text{conv}(Y)$ such that

$$\bar{w} = f_R^\star, \qquad \bar{z} \leq b.$$

By applying Lemma 3.4 to the set $Y = \sum_{i=1}^{N} Y_i$, it follows that there exists a set $I_{\text{CONV}} \subset \{1, \dots, N\}$, with cardinality at most $S+1$, and vectors

$$\begin{aligned} (\bar{b}_i, \bar{w}_i) &\in \text{conv}(Y_i), \qquad i \in I_{\text{CONV}}, \\ \bar{x}_i &\in X_i, \qquad\qquad\;\; i \notin I_{\text{CONV}}, \end{aligned}$$

such that

$$\begin{aligned}\sum_{i \notin I} g_i(\bar{x}_i) + \sum_{i \in I_{\text{CONV}}} \bar{b}_i &= \bar{z} \leq b, \\ \sum_{i \notin I} f_i(\bar{x}_i) + \sum_{i \in I_{\text{CONV}}} \bar{w}_i &= \bar{w} = f_R^\star.\end{aligned} \tag{3.60}$$

Hence, by Lemma 3.5, for all $i \in I_{\text{CONV}}$ there must exist vectors $x_i^1, \ldots, x_i^{S+2} \in X_i$ and scalars $\alpha_i^1, \ldots, \alpha_i^{S+2}$ such that

$$\begin{aligned}\alpha_i^j \geq 0, \quad \forall\, j \in \{1, \ldots, S+2\}, \quad &\sum_{j=1}^{S+2} \alpha_i^j = 1, \\ \bar{b}_i = \sum_{j=1}^{S+2} \alpha_i^j g_i(x_i^j) &\geq g_i\Big(\sum_{j=1}^{S+2} \alpha_i^j x_i^j\Big), \\ \bar{w}_i = \sum_{j=1}^{S+2} \alpha_i^j f_i(x_i^j) &\geq f_i\Big(\sum_{j=1}^{S+2} \alpha_i^j x_i^j\Big),\end{aligned} \tag{3.61}$$

where the inequalities follow by convexity of f_i and of the components of g_i. By defining $\tilde{x}_i = \sum_{j=1}^{S+2} \alpha_i^j x_i^j$ for all $i \in I_{\text{CONV}}$, it follows that

$$\tilde{x}_i \in \text{conv}(X_i) \quad \text{and} \quad \tilde{x}_i \notin X_i, \qquad \text{for all } i \in I_{\text{CONV}},$$

and also

$$\begin{aligned}\sum_{i \in I_{\text{CONV}}} \bar{b}_i &\geq \sum_{i \in I_{\text{CONV}}} g_i(\tilde{x}_i), \\ \sum_{i \in I_{\text{CONV}}} \bar{w}_i &\geq \sum_{i \in I_{\text{CONV}}} f_i(\tilde{x}_i).\end{aligned} \tag{3.62}$$

By plugging (3.62) in (3.60), it follows

$$\begin{aligned}\sum_{i \notin I_{\text{CONV}}} g_i(\bar{x}_i) + \sum_{i \in I_{\text{CONV}}} g_i(\tilde{x}_i) &\leq b, \\ \sum_{i \notin I_{\text{CONV}}} f_i(\bar{x}_i) + \sum_{i \in I_{\text{CONV}}} f_i(\tilde{x}_i) &\leq f_R^\star.\end{aligned} \tag{3.63}$$

Since the vector with components $\tilde{x}_i$, $i \in I_{\text{CONV}}$ and $\bar{x}_i$, $i \notin I_{\text{CONV}}$ is feasible for Problem (3.34), then (3.63) is satisfied with the equality. Thus, the vector is feasible and cost-optimal for Problem (3.34), i.e. it is an optimal solution. Thus, if we denote by

$(x_1^\star, \ldots, x_N^\star)$ the (unique) optimal solution of Problem (3.34), it holds

$$\begin{aligned} x_i^\star &= \tilde{x}_i \notin X_i, \quad \forall\, i \in I_{\text{CONV}}, \\ x_i^\star &= \bar{x}_i \in X_i, \quad \forall\, i \notin I_{\text{CONV}}, \end{aligned} \tag{3.64}$$

and the proof follows with $I = \{1, \ldots, N\} \setminus I_{\text{CONV}}$. ■

Proof of Theorem 3.6

By construction, for all $i \in \{1, \ldots, N\}$, $x_i^{\text{OUT}} \in X_i$, so that it suffices to show that $\sum_{i=1}^N g_i(x_i^{\text{OUT}}) \leq b$. Following the notation of Get-Nonconvex-Sol (Procedure 6), for all i, x_i^{CONV} denotes the optimal solution of Problem (3.36) with $y_i = y_i^\star$, x_i^{NC} denotes a feasible solution of Problem (3.37) with $y_i^{\text{END}} = y_i^\star$ (if it exists), and x_i^{VIOL} denotes a feasible solution of Problem (3.38) with $y_i^{\text{END}} = y_i^\star$.

By Lemma 3.1, the vector $(x_1^{\text{CONV}}, \ldots, x_N^{\text{CONV}})$ is an optimal solution of Problem (3.34), which by Assumption 3.6 is unique. Thus, by Proposition 3.3, there exists a set $I \subset \{1, \ldots, N\}$, with cardinality at least $N - S - 1$, such that, for $i \in I$, it holds $x_i^{\text{OUT}} = x_i^{\text{CONV}} \in X_i$. Then, it holds by construction

$$g_i(x_i^{\text{OUT}}) \leq y_i^\star, \qquad \forall\, i \in I. \tag{3.65}$$

As for indices $i \notin I$ (i.e. belonging to $\{1, \ldots, N\} \setminus I$), let us partition the set as $I_{\text{FEAS}} \cup I_{\text{INFEAS}}$, where agents in I_{FEAS} can find a feasible solution to Problem (3.37), while agents in I_{INFEAS} cannot. Thus, for $i \in I_{\text{FEAS}}$, it holds $x_i^{\text{OUT}} = x_i^{\text{NC}}$, and thus

$$g_i(x_i^{\text{OUT}}) \leq y_i^\star, \qquad \forall\, i \in I_{\text{FEAS}}. \tag{3.66}$$

For $i \in I_{\text{INFEAS}}$, it holds $x_i^{\text{OUT}} = x_i^{\text{VIOL}}$, and thus

$$\begin{aligned} g_i(x_i^{\text{OUT}}) &= g_i(x_i^{\text{CONV}}) + \big(g_i(x_i^{\text{VIOL}}) - g_i(x_i^{\text{CONV}})\big) \\ &\leq y_i^\star + \big(g_i(x_i^{\text{VIOL}}) - g_i(x_i^{\text{CONV}})\big) \\ &\leq y_i^\star + v_i^{\text{MAX}} \mathbf{1} \qquad \forall\, i \in I_{\text{INFEAS}}, \end{aligned} \tag{3.67}$$

where the last inequality follows by definition of v_i^{MAX}. By summing (3.67) over $i \in I_{\text{INFEAS}}$, we get

$$\sum_{i \in I_{\text{INFEAS}}} g_i(x_i^{\text{OUT}}) \leq \sum_{i \in I_{\text{INFEAS}}} y_i^\star + \sum_{i \in I_{\text{INFEAS}}} \big(g_i(x_i^{\text{L}}) - \ell_i\big), \leq \sum_{i \in I_{\text{INFEAS}}} y_i^\star + \sigma^\infty$$

Collecting all the inequalities and using the fact $\sum_{i=1}^{N} y_i^\star = b - \sigma^\infty$, we obtain

$$\sum_{i=1}^{N} g_i(x_i^{\text{OUT}}) = \sum_{i \in I} g_i(x_i^{\text{OUT}}) + \sum_{i \in I_{\text{FEAS}}} g_i(x_i^{\text{OUT}}) + \sum_{i \in I_{\text{INFEAS}}} g_i(x_i^{\text{OUT}}) \leq \sum_{i=1}^{N} y_i^\star + \sigma^\infty = b,$$

and the proof follows. ■

Proof of Theorem 3.7

The proof of the theorem uses arguments that are similar to Theorem 3.6, and additionally takes into account the evolution of the allocation vector sequence $\{y_1^t, \ldots, y_N^t\}_{t \geq 0}$.

By Proposition 2.1, the sequence $\{y_1^t, \ldots, y_N^t\}_{t \geq 0}$ converges to $(y_1^\star, \ldots, y_N^\star)$, an optimal solution of Problem (3.35). Moreover, let us denote by $(x_1^\star, \ldots, x_N^\star)$ the optimal solution of Problem (3.34) and let us split the agent set $\{1, \ldots, N\}$ as $I \cup I_{\text{FEAS}} \cup I_{\text{INFEAS}}$, where I contains agents for which $x_i^\star \in X_i$, I_{FEAS} contains agents for which $x_i^\star \notin X_i$ and Problem (3.37), with $y_i^{\text{END}} = y_i^\star$, is feasible, and I_{INFEAS} contains the remaining agents, for which $x_i^\star \notin X_i$ and Problem (3.37), with $y_i^{\text{END}} = y_i^\star$, is infeasible. For all $i \in \{1, \ldots, N\}$, let us fix $\epsilon_i > 0$ such that

$$\sum_{i \in I \cup I_{\text{FEAS}}} 2\epsilon_i \leq \delta.$$

By the convergence of y_i^t, there exists a sufficiently large $T_i > 0$ such that

$$y_i^t \leq y_i^\star + \epsilon_i \mathbf{1}, \qquad \forall\, t \geq T_i \text{ and } i \in \{1, \ldots, N\}. \tag{3.68}$$

For all $i \in \{1, \ldots, N\}$, let us denote by v_i^t the needed violation of the local allocation at time t, i.e. $v_i^t = 0$ for those agents for which $x_i^t \in X_i$, $v_i^t = 0$ for those agents for which $x_i^t \notin X_i$ and Problem (3.37) with $y_i^{\text{END}} = y_i^t$ is feasible, while for the remaining agents v_i^t is the optimal cost of

$$\begin{aligned} \min_{x_i, v_i} \;\; & v_i \\ \text{subj. to } \;\; & g_i(x_i) \leq y_i^t + v_i \mathbf{1} \\ & x_i \in X_i. \end{aligned} \tag{3.69}$$

As for $i \in I \cup I_{\text{FEAS}}$, it holds

$$\begin{aligned} g_i(x_i^t) &\overset{(a)}{\leq} y_i^t + v_i^t \mathbf{1} \\ &\overset{(b)}{\leq} y_i^\star + (\epsilon_i + v_i^t)\mathbf{1} \\ &\overset{(c)}{\leq} y_i^\star + 2\epsilon_i \mathbf{1}, \qquad \forall\, t \geq T_i, \end{aligned} \tag{3.70}$$

where (a) follows by construction, (b) follows by (3.68), (c) follows either since $v_i^t = 0 < \epsilon_i$ or since (x_i^t, ϵ_i) is feasible for Problem (3.69) (for all $t \geq T_i$) and v_i^t is the optimal cost, so that $v_i^t \leq \epsilon_i$.

As for $i \in I_{\text{INFEAS}}$, it holds

$$g_i(x_i^t) = g_i(x_i^\star) + g_i(x_i^t) - g_i(x_i^\star) \leq y_i^\star + v_i^{\text{MAX}}\mathbf{1} \qquad \forall\, t \geq 0, \tag{3.71}$$

where $g_i(x_i^\star) \leq y_i^\star$ follows by construction and we used the definition of v_i^{MAX}.

By collecting (3.70) and (3.71) we obtain

$$\begin{aligned}
\sum_{i=1}^N g_i(x_i^t) &= \sum_{i \in I \cup I_{\text{FEAS}}} g_i(x_i^t) + \sum_{i \in I_{\text{INFEAS}}} g_i(x_i^t) \\
&\leq \underbrace{\sum_{i=1}^N y_i^\star}_{\leq b - \sigma^\infty - \delta\mathbf{1}} + \sum_{i \in I \cup I_{\text{FEAS}}} 2\epsilon_i \mathbf{1} + \underbrace{\sum_{i \in I_{\text{INFEAS}}} v_i^{\text{MAX}}\mathbf{1}}_{\leq \sigma^\infty} \\
&\leq b + \underbrace{\bigg(\sum_{i \in I \cup I_{\text{FEAS}}} 2\epsilon_i - \delta \bigg)}_{\leq 0} \mathbf{1} \\
&\leq b, \qquad\qquad \forall\, t \geq T_\delta,
\end{aligned}$$

where $T_\delta = \max\limits_i T_i$, so that the proof is complete. ■

Chapter 4

Distributed Stochastic Microgrid Control

In this chapter, we consider an important energy network application, namely distributed control of microgrids. We start by reviewing a mixed-integer optimal control problem for microgrid operation in a deterministic setting. We apply the methodology developed in Chapter 3 and show simulation results. Then, we consider a more complex scenario with renewable energy sources and recall a two-stage stochastic optimization approach. We extend the method of Chapter 3 to cope with the stochastic scenario and provide a bound on the worst-case constraint violation. Finally, we provide numerical computations in which a fictious microgrid control problem is synthetized using Generative Adversarial Networks (GANs). The results of this chapter are partially based on [27].

4.1 Literature Review

Microgrid control is an important topic in the control community. In [15], a distributed approach to optimal reactive power compensation is proposed. In [125], a stochastic optimization method for energy and reserve scheduling with renewable energy sources and demand-side participation is considered. The work [82] studies a stochastic unit commitment and economic dispatch problem with renewables and incorporating the battery operating cost. During the last years, Model Predictive Control (MPC) techniques are being applied to microgrid control. In [88], a centralized MPC approach for N interconnected smart grids is proposed, while in [37], a two-layer stochastic MPC approach for microgrids is developed. Due to its ability to model logical statements, recently also Mixed-Integer Linear Programming (MILP) is gathering significant attention. In [63], a MILP optimal control approach of residential microgrid is proposed. In [70] a mixed-integer nonlinear programming formulation is considered with experimental validation for islanded-mode microgrids. In [107], a MILP is formulated to achieve optimal load shifting in microgrids. The MPC and the MILP approach have been combined in [89], which tests a receding horizon implementation of the MILP approach on an experimental testbed. A stochastic version is then considered in [90], which aims at an environmental/economical operation of microgrids with renewable energy sources.

4.2 Distributed Mixed-integer Microgrid Model

In this section, we consider the mixed-integer microgrid control problem described in [89]. We show that the problem can be recast as a constraint-coupled Mixed-Integer Linear Program (MILP). Then, we apply the framework of Chapter 3 and show simulation results.

4.2.1 Mixed-Integer Microgrid Optimal Control

Let us begin by recalling the microgrid model and the optimal control MILP. A microgrid consists of N units, partitioned as follows. Storages are collected in I_{STOR}, generators in I_{GEN}, critical loads in I_{LO}, controllable loads in I_{CL} and one connection with the utility grid in I_{GRID}, so that the whole set of units is $\{1, \ldots, N\} = I_{\text{STOR}} \cup I_{\text{GEN}} \cup I_{\text{LO}} \cup I_{\text{CL}} \cup I_{\text{GRID}}$. We denote the length of the prediction horizon as $K \in \mathbb{N}$ and the time index in the optimal control problem as $k \in \mathbb{N}$.

Storages

For storage units $i \in I_{\text{STOR}}$, let $x_i(k) \in \mathbb{R}$ be the stored energy level at time k and let $u_i(k) \in \mathbb{R}$ denote the power exchanged with the storage unit at time k (positive for charging, negative for discharging). The dynamics to be satisfied amounts to $x_i(k+1) = x_i(k) + \eta_i u_i(k) - x_i^{\text{PL}}$, where η_i denotes the (dis)charging efficiency and x_i^{PL} is a physiological loss of energy. It is assumed that $\eta_i = \eta_i^c$ if $u_i(k) \geq 0$ (charging mode), whereas $\eta_i = 1/\eta_i^d$ if $u_i(k) < 0$ (discharging mode), with $0 < \eta_i^c, \eta_i^d < 1$. The piece-wise linear dynamics is reformulated by using mixed-integer inequalities [7]. To this end, in [89] the authors introduce additional variables $\delta_i(k) \in \{0, 1\}$ and $z_i(k) \triangleq \delta_i(k) u_i(k) \in \mathbb{R}$ for all k. Each $\delta_i(k)$ is one if and only if $u_i(k) \geq 0$ (i.e. the storage unit is in the charging state). After following the manipulations proposed in [89], we obtain the following model for the i-th storage unit,

$$x_i(k+1) = x_i(k) + (\eta_i^c - 1/\eta_i^d) z_i(k) + 1/\eta^d u_i(k) - x_i^{\text{PL}} \qquad \forall\, k \tag{4.1a}$$

$$E_i^1 \delta_i(k) + E_i^2 z_i(k) \leq E_i^3 u_i(k) + E_i^4 \qquad \forall\, k \tag{4.1b}$$

$$x_i^{\text{MIN}} \leq x_i(k) \leq x_i^{\text{MAX}} \qquad \forall\, k \tag{4.1c}$$

$$x_i(0) = x_{i,0}, \tag{4.1d}$$

Table 4.1: List of the main symbols and their definitions

Basic definitions	
$N \in \mathbb{N}$	Number of units in the system
$\mathbb{I} = \{1, \ldots, N\}$	Set of units
$\varepsilon > 0$	Very small number (e.g. machine precision)
Storages (indexed by $i \in I_{\text{STOR}}$)	
$x_i(k)$	State of charge at time k
$u_i(k)$	Exchanged power (≥ 0 if charging) at time k
$\delta_i(k)$	Charging (1) / discharging (0) state
$z_i(k)$	Auxiliary optimization variable
η_i^c, η_i^d	Charging and discharging efficiencies
x_i^{MIN}, x_i^{MAX}	Minimum and maximum storage level
x_i^{PL}	Physiological loss of energy
C_i	Maximum output power
OM_i	Operation and maintenance cost coefficient
Generators (indexed by $i \in I_{\text{GEN}}$)	
$u_i(k)$	Generated power (≥ 0) at time k
$\delta_i(k)$	On (1) / off (0) state ("on" iff $u_i(k) > 0$)
$\nu_i(k)$	Epigraph variable for quadratic generation cost
$\theta_i^{\text{U}}(k)$, $\theta_i^{\text{D}}(k)$	Epigraph variables for startup/shutdown costs
T_i^{UP}, T_i^{DOWN}	Minimum up/down time
u_i^{MIN}, u_i^{MAX}	Min. and max. power that can be generated
r_i^{MAX}	Maximum ramp-up/ramp-down
$\kappa_i^{\text{U}}(k)$, $\kappa_i^{\text{D}}(k)$	Startup and shutdown costs
OM_i	Operation and maintenance cost coefficient
Renewable energy sources (indexed by $i \in I_{\text{REN}}$)	
$P_i(k)$	Generated power at time k
Controllable loads (indexed by $i \in I_{\text{CL}}$)	
$\beta_i(k)$	Curtailment factor ($\in [\beta_i^{\text{MIN}}, \beta_i^{\text{MAX}}]$) at time k
$D_i(k)$	Consumption forecast at time k
β_i^{MIN}, β_i^{MAX}	Minimum and maximum allowed curtailment
Connection to the main grid (indexed by $i \in I_{\text{GRID}}$)	
$u_i(k)$	Imported power from the grid at time k
$\delta_i(k)$	Importing (1) or exporting (0) mode at time k
$\phi_i(k)$	Total expenditure for imported power at time k
$\phi_i^{\text{P}}(k)$, $\phi_i^{\text{S}}(k)$	Price for power purchase and sell at time k
P_i^{MAX}	Maximum exchangeable power

where (4.1a) is the dynamics, (4.1b) are mixed-integer inequalities expressing the logical constraints, (4.1c) are box constraints on the state of charge (for $0 < x_i^{\text{MIN}} < x_i^{\text{MAX}}$), and (4.1d) contains the initial conditions ($x_{i,0} \in \mathbb{R}$ is the initial state of charge). The matrices in (4.1b) are

$$E_i^1 = \begin{bmatrix} C_i \\ -(C_i + \varepsilon) \\ C_i \\ C_i \\ -C_i \\ -C_i \end{bmatrix}, \quad E_i^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad E_i^3 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \quad E_i^4 = \begin{bmatrix} C_i \\ -\varepsilon \\ C_i \\ C_i \\ 0 \\ 0 \end{bmatrix},$$

where $C_i > 0$ is the limit output power and $\varepsilon > 0$ is a very small number (typically machine precision). The cost associated to each storage i is

$$\text{Cost}_i = \sum_{k=0}^{K-1} \text{OM}_i \cdot |u_i(k)| = \sum_{k=0}^{K-1} \text{OM}_i \cdot (2z_i(k) - u_i(k)), \tag{4.2}$$

where $\text{OM}_i > 0$ is the operation and maintenance cost and $2z_i(k) - u_i(k) = |u_i(k)|$ is the absolute value of the power exchanged with the storage.

Generators

For generators $i \in I_{\text{GEN}}$, let $u_i(k) \in \mathbb{R}, u_i(k) \geq 0$ denote the generated power at time k. Since generators can be either on or off, as done for the storages we let $\delta_i(k) \in \{0, 1\}$ be an auxiliary variable that is equal to 1 if and only if $u_i(k) > 0$. As in the case of storages, we must consider constraints on the operating conditions of generators. Namely, if a generator is turned on/off, there is a minimum amount of time for which the unit must be kept on/off. This logical constraint is modeled by the inequalities

$$\delta_i(k) - \delta_i(k-1) \leq \delta_i(\tau), \qquad \tau = k+1, \dots, \min(k + T_i^{\text{UP}} - 1, T), \tag{4.3a}$$

$$\delta_i(k-1) - \delta_i(k) \leq \delta_i(\tau), \qquad \tau = k+1, \dots, \min(k + T_i^{\text{DOWN}} - 1, T), \tag{4.3b}$$

for all $k = 0, \dots, T-1$, where T_i^{UP} and T_i^{DOWN} are the minimum up and down time of generator i. We assume the generators have ramping constraints. Specifically, the power flow limit and the ramp-up/ramp-down limits are modeled respectively by

$$u_i^{\text{MIN}} \delta_i(k) \leq u_i(k) \leq u_i^{\text{MAX}} \delta_i(k), \tag{4.3c}$$

$$-r_i^{\text{MAX}} \delta_i(k) \leq u_i(k) - u_i(k-1) \leq r_i^{\text{MAX}} \delta_i(k), \tag{4.3d}$$

for all $k \in \{0, \dots, K-1\}$, where $u_i^{\text{MAX}} \geq u_i^{\text{MIN}} \geq 0$ denote the maximum and minimum power that can be generated and $r_i^{\text{MIN}} \geq 0$ denotes the maximum ramp-up/ramp-down.

The cost associated to generator units is composed of two parts, which are *(i)* a quadratic generation cost, and *(ii)* a start-up/shut-down cost. To model the quadratic generation cost, as done in [89] we consider a linearized version $\max_\ell \big(S_i^\ell u_i(k) + s_i^\ell\big)$ which is written in the epigraph form

$$\nu_i(k) \geq S_i^\ell u_i(k) + s_i^\ell, \qquad \forall\, \ell, \tag{4.3e}$$

for all $k \in \{0, \dots, K-1\}$, where $\nu_i(k) \in \mathbb{R}$ is the epigraph variable. The startup θ_i^{U} and shutdown cost θ_i^{D} at each time $k \in \{0, \dots, K-1\}$ are equal to

$$\begin{aligned}
\theta_i^{\text{U}}(k) &= \max\Big\{0,\ \kappa_i^{\text{U}}(k)[\delta_i(k) - \delta_i(k-1)]\Big\},\\
\theta_i^{\text{D}}(k) &= \max\Big\{0,\ \kappa_i^{\text{D}}(k)[\delta_i(k-1) - \delta_i(k)]\Big\},
\end{aligned}$$

where $\kappa_i^{\text{U}}(k), \kappa_i^{\text{D}}(k) > 0$ are the start-up and shut-down cost at time k. Also in this case we write them in epigraph form

$$\theta_i^{\text{U}}(k) \geq \kappa_i^{\text{U}}(k)[\delta_i(k) - \delta_i(k-1)], \tag{4.3f}$$

$$\theta_i^{\text{D}}(k) \geq \kappa_i^{\text{D}}(k)[\delta_i(k-1) - \delta_i(k)], \tag{4.3g}$$

$$\theta_i^{\text{U}}(k) \geq 0, \tag{4.3h}$$

$$\theta_i^{\text{D}}(k) \geq 0, \tag{4.3i}$$

for all $k \in \{0, \dots, K-1\}$, and treat the variables $\theta_i^{\text{U}}(k), \theta_i^{\text{D}}(k) \in \mathbb{R}$ as epigraph variables. Thus, the final expression for the cost of each generator i is

$$\text{COST}_i = \sum_{k=0}^{K-1} \big(\text{OM}_i \cdot \delta_i(k) + \nu_i(k) + \theta_i^{\text{U}}(k) + \theta_i^{\text{D}}(k)\big). \tag{4.4}$$

Loads

Loads are of two types, namely critical loads and controllable loads. For critical loads $i \in I_{\text{LO}}$, we will denote by $D_i(k)$ the consumption forecast at time k and we assume it is known a priori. This assumption is fairly realistic, since in real contexts one can use historical data to formulate load predictions. Alternatively, the consumers themselves can formulate a prediction of the load schedule for the next day. The consumption of critical loads will be considered in the power balance. There is no cost associated to this kind of loads.

For controllable loads $i \in I_{\text{CL}}$, let $D_i(k)$ be the consumption forecast at time k, which is also assumed to be known a priori. The actual power consumption at each time instant is $(1-\beta_i(k))D_i(k)$, where $\beta_i(k) \in \mathbb{R}$ is the curtailment factor. This is subject to the constraint that

$$\beta_i^{\text{MIN}} \leq \beta_i(k) \leq \beta_i^{\text{MAX}}, \qquad \forall\, k \in \{0, \ldots, K-1\}, \tag{4.5}$$

where $0 < \beta_i^{\text{MIN}} < \beta_i^{\text{MAX}} < 1$ are the bounds on the allowed curtailment. In case of curtailed power, the microgrid incurs in the cost

$$\text{COST}_i = \sum_{k=0}^{K-1} \varphi_i D_i(k) \beta_i(k), \tag{4.6}$$

where $\varphi_i > 0$ is a penalty weight.

Connection to the utility grid

For the connection with the utility grid $i \in I_{\text{GRID}}$, let $u_i(k) \in \mathbb{R}$ denote the imported (exported) power level from (to) the utility grid. We use the convention that imported power at time k is positive $u_i(k) \geq 0$. As before, since the power purchase price is different from the power sell price, we consider auxiliary variables $\delta_i(k) \in \{0,1\}$ and $\phi_i(k) \in \mathbb{R}$ for all k. The variable $\delta_i(k)$ models the logical statement $\delta_i(k) = 1$ if and only if $u_i(k) \geq 0$ (i.e. power is imported from the utility grid). Denoting by $\phi_i^{\text{P}}(k), \phi_i^{\text{S}}(k) \geq 0$ the price for power purchase and sell, we let $\phi_i(k) = \phi_i^{\text{P}}(k) u_i(k)$ if $\delta_i(k) = 1$ and $\phi_i(k) = \phi_i^{\text{S}}(k) u_i(k)$ if $\delta_i(k) = 0$. By denoting by $P_i^{\text{MAX}} \geq 0$ the maximum power flow, the corresponding mixed-integer inequalities are

$$E_i^1 \delta_i(k) + E_i^2 \phi_i(k) \leq E_i^3(k) u_i(k) + E_i^4, \qquad k = 0, \ldots, T-1, \tag{4.7}$$

where the matrices are defined as

$$E_i^1 = \begin{bmatrix} P_i^{\text{MAX}} \\ -(P_i^{\text{MAX}} + \varepsilon) \\ M_i \\ M_i \\ -M_i \\ -M_i \end{bmatrix}, \quad E_i^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad E_i^3(k) = \begin{bmatrix} 1 \\ -1 \\ \phi^{\text{P}}(k) \\ -\phi^{\text{P}}(k) \\ \phi^{\text{S}}(k) \\ -\phi^{\text{S}}(k) \end{bmatrix}, \quad E_i^4 = \begin{bmatrix} P_i^{\text{MAX}} \\ -\varepsilon \\ M_i \\ M_i \\ 0 \\ 0 \end{bmatrix},$$

with $M_i = P_i^{\text{MAX}} \max_k(\phi^{\text{P}}(k), \phi^{\text{S}}(k))$. Clearly, the cost associated with this unit is

$$\text{COST}_i = \sum_{k=0}^{K-1} \phi_i(k). \tag{4.8}$$

Power balance constraint and optimal control problem

To write the power balance constraint, we consider a more relaxed constraint than the one in [89]. Namely, we impose that the power drawn from the utility grid must be sufficient to meet the needs of the microgrid, i.e.

$$u_{I_{\text{GRID}}}(k) \geq \sum_{i \in I_{\text{STOR}}} u_i(k) - \sum_{i \in I_{\text{GEN}}} u_i(k) + \sum_{i \in I_{\text{CL}}} (1 - \beta_i(k)) D_i(k) + \sum_{i \in I_{\text{LO}}} D_i(k), \tag{4.9}$$

for all $k \in \{0, \ldots, K-1\}$. Note that the minimal cost for the power exchanged with the utility grid is obtained when power is sold (rather then bought). Thus, since the problem is in minimization form, the solution will be on the boundary of this constraint, and in fact optimal solutions satisfy this constraint with the equality.

The optimal control MILP can be finally posed as

$$\begin{aligned} \min_{u} \; & \sum_{k=0}^{K-1} \Bigg[\sum_{i \in I_{\text{STOR}}} (\text{OM}_i \cdot (2 z_i(k) - u_i(k)) + \sum_{i \in I_{\text{CL}}} \varphi_i D_i(k) \beta_i(k) + \phi_{\text{GRID}}(k) \\ & \qquad + \sum_{i \in I_{\text{GEN}}} (\text{OM}_i \cdot \delta_i(k) + \nu_i(k) + \theta_i^{\text{U}}(k) + \theta_i^{\text{D}}(k)) \Bigg] \\ \text{subj. to} \; & \text{storage constraints (4.1)} \\ & \text{generator constraints (4.3)} \\ & \text{constraints (4.5), (4.7), (4.9).} \end{aligned} \tag{4.10}$$

4.2.2 Constraint-coupled Reformulation

Let us now fit the microgrid control problem (4.10) into the constraint-coupled MILP (3.1),

$$\begin{aligned} \min_{x_1, \ldots, x_N} \; & \sum_{i=1}^{N} c_i^\top x_i \\ \text{subj. to} \; & \sum_{i=1}^{N} A_i x_i \leq b \\ & x_i \in X_i^{\text{MILP}}, \qquad i = 1, \ldots, N, \end{aligned} \tag{4.11}$$

where we recall that N is the number of agents and, for all $i \in \{1, \ldots, N\}$, $x_i \in \mathbb{R}^{n_i}$ is the i-th optimization variable with cost vector $c_i \in \mathbb{R}^{n_i}$, the set $X_i^{\text{MILP}} \subset \mathbb{Z}^{p_i} \times \mathbb{R}^{q_i}$ is a compact mixed-integer polyhedron, while the matrices $A_i \in \mathbb{R}^{S \times (p_i + q_i)}$ and the vector $b \in \mathbb{R}^S$ model the coupling constraints.

Device-specific quantities and coupling constraints

To achieve this reformulation, we now specify the quantities x_i, c_i, X_i^{MILP}, A_i and b.

Storages. We assume that each storage unit $i \in I_{\text{STOR}}$ is responsible for the optimization vector x_i consisting of the stack of $x_i(k), u_i(k), z_i(k) \in \mathbb{R}$ and $\delta_i(k) \in \{0,1\}$ for all $k \in \{0,\dots,K-1\}$ plus the variable $x_i(K) \in \mathbb{R}$. The constraints in X_i^{MILP} are given by (4.1), while the cost function is $c_i^\top x_i = \sum_{k=0}^{K-1} \text{OM}_i \cdot (2z_i(k) - u_i(k))$.

Generators. Each generator $i \in I_{\text{GEN}}$ is responsible for the optimization vector x_i consisting of the stack of $u_i(k), \nu_i(k), \theta_i^{\text{U}}(k), \theta_i^{\text{D}}(k) \in \mathbb{R}$ and $\delta_i(k) \in \{0,1\}$ for all $k \in \{0,\dots,K-1\}$. The constraints in X_i^{MILP} are given by (4.3a)–(4.3i), while the cost function is $c_i^\top x_i = \sum_{k=0}^{K-1} \big(\text{OM}_i \cdot \delta_i(k) + \nu_i(k) + \theta_i^{\text{U}}(k) + \theta_i^{\text{D}}(k)\big)$.

Critical loads. For the critical loads $i \in I_{\text{LO}}$ there are no variables to optimize, but they must be taken into account in the coupling constraints.

Controllable loads. For each controllable load $i \in I_{\text{CL}}$ the optimization vector x_i consists of the stack of $\beta_i(k) \in \mathbb{R}$, for all $k \in \{0,\dots,K-1\}$, with constraints given by (4.5). Note that, for this class of devices, the local constraint set is not mixed-integer. The cost function is $c_i^\top x_i = \sum_{k=0}^{K-1} \varphi_i D_i(k)\beta_i(k)$.

Connection to the utility grid. For this device $i \in I_{\text{GRID}}$, the optimization vector x_i consists of the stack of $u_i(k), \phi_i(k) \in \mathbb{R}$ and $\delta_i(k) \in \{0,1\}$ for all $k \in \{0,\dots,K-1\}$. The local constraints are (4.7), while the cost function is $c_i^\top x_i = \sum_{k=0}^{K-1} \phi_i(k)$.

Coupling constraints. Finally, the coupling constraints are given by (4.9), which can be encoded in the form $\sum_{i=1}^{N} A_i x_i \le b$ by appropriately defining the matrices A_i and the vector b in such a way that each agent matches its contribution to the microgrid power consumption (more details are given later). In particular, the right-hand side vector is equal to

$$b = -\sum_{i \in I_{\text{CL}}} \beta_i(k) D_i(k) - \sum_{i \in I_{\text{LO}}} D_i(k).$$

Application of Mixed-Integer Methodology

To Problem (4.11) we want to apply the MILP methodology developed in Chapter 3 and obtain asymptotic feasibility (cf. Theorem 3.1). To this end, we first compute the restriction using the definition (3.11), which is recalled here

$$\sigma^\infty = S \cdot \max_{i \in \{1,\dots,N\}} \sigma_i^{\text{LOC}},$$

where each σ_i^{LOC} is equal to the worst-case violation of agent i as discussed in Section 3.3.3. Note that the local constraint set X_i^{MILP} of controllable loads is not mixed integer and therefore it holds $\sigma_i^{\text{LOC}} = \mathbf{0}$ for all $i \in I_{\text{CL}}$.

As discussed in Section 3.3.1, the algorithm must be initialized with initial allocations vectors y_i^0 satisfying

$$\sum_{i=1}^{N} [y_i^0]_k = b_k = -\sum_{i \in I_{\text{CL}}} D_i(k) - \sum_{i \in I_{\text{LO}}} D_i(k), \qquad k = 0, \ldots, K-1.$$

In particular, we assume that all the nodes initialize their $y_i^0 = \mathbf{0}$ except the controllable loads that initialize $[y_i^0]_k = -D_i(k)$ for all k and the node $i \in I_{\text{GRID}}$ that is assumed to know the predicted values of the critical loads and thus initializes $[y_i^0]_k = -\sum_{i \in I_{\text{LO}}} D_i(k)$ for all k. In such a way, all the information that is not related to critical loads is kept private.

4.2.3 Simulation Results

Let us now show Montecarlo simulations of Algorithm 3 on the microgrid control problem. In particular, we test the magnitude of the restriction σ^∞ and the suboptimality of the computed solution and compare the performance of Algorithm 3 with the algorithm in [121]. We consider $N = 151$ agents: 50 storages, 50 generators, 50 controllable loads, and 1 connection to the main grid, with prediction horizon $K = 12$. Therefore, the total number of optimization variables is 6.086, out of which 4.874 are in $\mathbb{R}$ and 1.212 are in $\{0, 1\}$. A total amount of 50 random instances of the problem have been generated, with $S = K = 12$ coupling constraints.

We run the distributed algorithm up to asymptotic convergence and obtain a feasible mixed-integer solution $(x_1^\infty, \ldots, x_N^\infty)$. In Figure 4.1 (left), we provide an histogram of the relative restriction magnitude with respect to the total available resource, i.e. $\|\sigma\|/\|b\|$. The picture shows that the magnitude of our restriction is about 1% of $\|b\|$. Moreover, compared to the restriction proposed by [121], our restriction is about 240 times smaller. We were not able to find a proper tuning of the problems parameters to apply the approach proposed by [121]. Indeed, for the generated instances, the approximate Problem (3.2), with their restriction, turns out to be infeasible. In Figure 4.1 (right), we provide an histogram of the relative suboptimality of the solution produced by Algorithm 3, with respect to the dual cost $q^\star$ of MILP (4.11). It can be seen that the computed solutions enjoy around 3.4% suboptimality. Notice that this value represents a worsening estimate of the actual suboptimality since $q^\star \leq J^\star$, where $J^\star$ is the optimal cost of MILP (4.11).

Finally, we show the evolution of Algorithm 3 on one generated instance. Here, we employ an additional restriction equal to 1 to ensure finite-time feasibility (cf. Theorem 3.3) of the computed solution. In Figure 4.2, the evolution of the coupling constraint utilization of $(x_1^t, \ldots, x_N^t)$ is plotted. It can be seen that the mixed-integer solution becomes feasible for the original coupling constraint of MILP (4.11) in about 400 iterations, when the maximum value falls below 0.

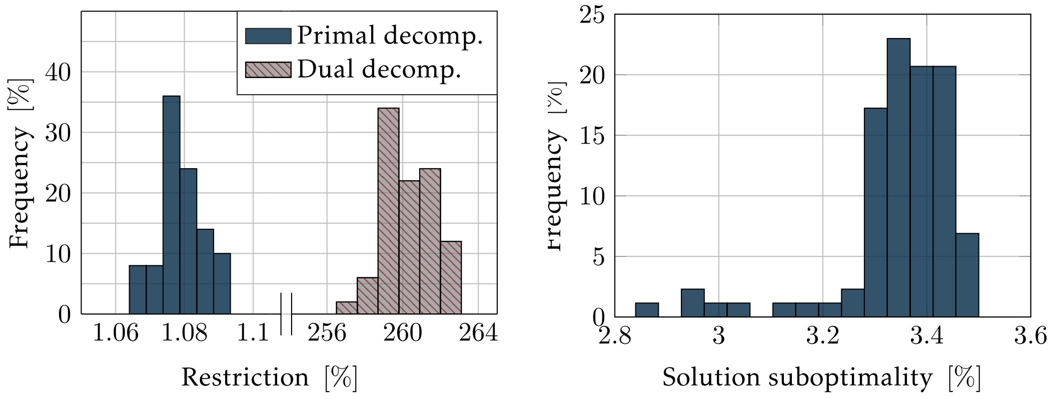


Figure 4.1: Montecarlo simulations on the microgrid problem. Left: relative restriction magnitude with respect to total resource, computed as $\|\sigma\|/\|b\|$, for Algorithm 3 and for the dual decomposition approach in [121]. Right: solution suboptimality, computed as $\big(\sum_{i=1}^{N} c_i^\top x_i^\infty - q^\star\big)/q^\star$, for our approach ([121] is infeasible).

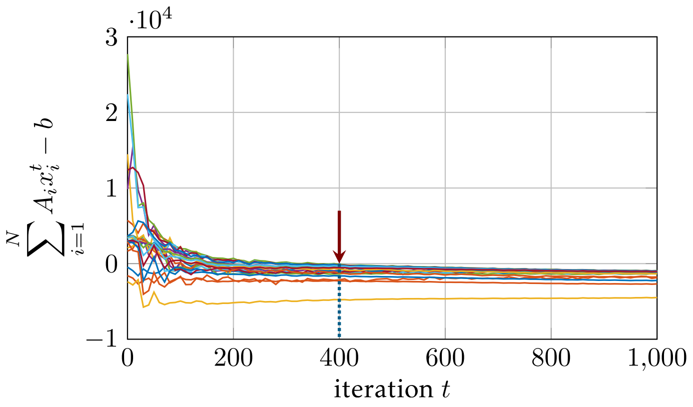


Figure 4.2: Components of the coupling constraint associated to $(x_1^t, \dots, x_N^t)$, generated by our algorithm on an instance of the microgrid problem. The arrow denotes the instant in which the solution becomes globally feasible.

4.3 Distributed Stochastic Mixed-integer Microgrid Control

In this section, we show how to extend the distributed microgrid control framework to the stochastic case. We consider the stochastic formulation of the microgrid control problem discussed in [90]. The microgrid model described in [90] is slightly more complex than the one considered in Section 4.2 as combined heat and power (CHP) capabilities are additionally considered. Without loss of generality, since the main focus of this section is to show how the extend the distributed microgrid control algorithm to the stochastic scenario, we prefer to keep the already introduced microgrid model by only introducing the needed changes due to the stochastic scenario. Indeed, the additional optimization variables and constraints stemming from the extended modeling of [90] can be easily included in our framework.

4.3.1 Stochastic Microgrid Model with Renewable Sources

The microgrid model considered here is similar to the one introduced in Section 4.2.1, except that here we also consider the presence of renewable energy sources, which are treated in a stochastic manner.

Let us now assume that the set of devices $i \in \{1, \dots, N\}$ also contains renewable energy sources with indices $i \in I_{\text{REN}}$. These units only contribute to the power balance constraint (4.9) through their generated power at each time slot k, denoted as $P_i(k)$, and do not have associated cost or constraints. Thus, the power balance constraint becomes

$$u_{I_{\text{GRID}}}(k) = \sum_{i \in I_{\text{STOR}}} u_i(k) - \sum_{i \in I_{\text{GEN}}} u_i(k) + \sum_{i \in I_{\text{CL}}} (1 - \beta_i(k)) D_i(k) + \sum_{i \in I_{\text{LO}}} D_i(k) - \sum_{i \in I_{\text{REN}}} P_i(k), \tag{4.12}$$

for all $k \in \{0, \dots, K-1\}$, where we assumed $P_i(k) \geq 0$. Differently from Section 4.2, here we also have the stochastic variables $P_i(k)$ for $i \in I_{\text{REN}}$. Moreover, note also that differently from (4.9) the constraint (4.12) is an equality.

We consider two types of renewables, namely wind generators and solar generators. Rather than using a physical or dynamical model for these generators, by relying on huge amount of datasets freely available on the internet in this work we prefer to use a predictor to generate realistic power production scenarios. We will employ this technique also to predict the power demand (see Section 4.3.5 for more details).

4.3.2 Distributed Constraint-coupled Stochastic Optimization

Bearing in mind that the constraint (4.12) is stochastic, let us introduce the stochastic optimization problem associated to the new microgrid model. As done in [90], we formulate a so-called two-stage stochastic linear program with simple recourse. This problem is in a sense similar to Problem (4.10), but since some P_i are unknown it is not possible to satisfy the equality a-priori.

Essentially, in this problem scenario we initially choose a set of control actions $u_i(k)$ that minimize a given cost but in general will not satisfy the power balance (4.12) "a posteriori" (i.e. when the value of the random variables is realized). To compensate this infeasibility, *recourse* actions must be taken, which impact on the actual final cost associated to the chosen control actions. In view of this, the considered problem allows for a violation of the power balance constraint *by construction*. For this reason, in the problem there are two kinds of variables. The so-called first-stage variables are the actual control actions $u_i(k)$, while the constraint violations, which are penalized in the cost, are the so-called two-stage variables that model the recourse. Associated to the two-stage variables is a linear objective function, which models the cost associated to recourse actions.

Formally, we denote by Ω the random vector collecting all the renewable energy generation profiles. We assume a finite discrete probability distribution for Ω and we denote by π_r the probability of each ω_r, i.e. $\pi_r = \mathbb{P}(\omega = \omega_i)$ for all $r \in \{1, \dots, R\}$. To keep the notation consistent we denote the renewable energy profile corresponding to ω_r as $P_{ir}(k)$. By following a similar derivation as in Section 4.2.2, we can define the local variables, the local constraints, the cost vectors c_i and the contributions to the coupling constraints. In particular, the power balance (4.12) gives rise to an *equality* coupling constraint of the type $\sum_{i=1}^{N} H_i x_i = h$, where

$$[H_i x_i]_k = u_i(k), \qquad \forall i \in I_{\text{STOR}}, \tag{4.13a}$$

$$[H_i x_i]_k = -u_i(k), \qquad \forall i \in I_{\text{GEN}}, \tag{4.13b}$$

$$[H_i x_i]_k = -\beta_i(k) D_i(k), \qquad \forall i \in I_{\text{CL}}, \tag{4.13c}$$

$$[H_i x_i]_k = -u_i(k), \qquad \forall i \in I_{\text{GRID}}, \tag{4.13d}$$

and

$$h(k) = -\sum_{i \in I_{\text{CL}}} D_i(k) - \sum_{i \in I_{\text{LO}}} D_i(k) + \sum_{i \in I_{\text{REN}}} P_i(k). \tag{4.13e}$$

Note that this equality constraint has K components. Moreover, we point out that the random variables appear only in the vector h. We denote by h_r the realization of h associated to the scenario ω_r. Using these positions, the two-stage stochastic MILP can then be formulated as

$$
\begin{aligned}
\min_{\substack{x_1,\ldots,x_N \\ \eta_+,\eta_-}} \quad & \sum_{i=1}^{N} c_i^\top x_i + \sum_{k=0}^{K-1}\sum_{r=1}^{R} \pi_r (q_+ \eta_{kr+} + q_- \eta_{kr-}) \\
\text{subj. to} \quad & \eta_{r-} \le \sum_{i=1}^{N} H_i x_i - h_r \le \eta_{r+}, \qquad r = 1,\ldots,R \\
& \eta_+, \eta_- \ge \mathbf{0}, \\
& x_i \in X_i^{\text{MILP}}, \qquad\qquad i = 1,\ldots,N,
\end{aligned}
\tag{4.14}
$$

where $x_1,\ldots,x_N$ are the first-stage variables modeling the (a-priori) control actions and η_+, η_- are the two-stage variables modeling the (a-posteriori) recourse actions, which are penalized in the cost with $q_+ \ge 0$ and $q_- \ge 0$, which are the costs related to energy surplus and shortage, respectively. In Problem (4.14), we denoted by η_{kr+} the variable associated with positive recourse for scenario r at time k. We also use the symbol η_{r+} to denote the stack of η_{kr+} for all k. The stack of η_{kr+} for all k and r is denoted by η_+. A similar notation holds for η_-. It is easy to see that the additional term in the cost is the expected value of the cost associated to recourse actions, i.e.

$$
\sum_{k=0}^{K-1}\sum_{r=1}^{R} \pi_r (q_+ \eta_{kr+} + q_- \eta_{kr-}) = \sum_{k=0}^{K-1} \mathbb{E}\left[\Phi\left(\left[\sum_{i=1}^{N} H_i x_i - h\right]_k\right)\right],
$$

where $\Phi(z) = q_+ z$ if $z \ge 0$ and $\Phi(z) = -q_- z$ if $z < 0$.

At a first glance, it may seem that the two-stage problem (4.14) loses the constraint-coupled structure of the deterministic formulation (4.11). However, with a bit a manipulation, it is still possible to arrive at a similar result. We begin by streamlining the notation. Define $\eta \in \mathbb{R}^{2KR}$, $\eta \ge 0$ as the stack of η_+ and η_-, and the vector $d \in \mathbb{R}^{2KR}$ such that $d^\top \eta = \sum_{k=0}^{K-1}\sum_{r=1}^{R} \pi_r (q_+ \eta_{kr+} + q_- \eta_{kr-})$. Moreover, define also $A_i \in \mathbb{R}^{2KR \times n_i}$ and $b \in \mathbb{R}^{2KR}$ with

$$
\begin{aligned}
A_i &= \mathbf{1} \otimes \begin{bmatrix} H_i \\ -H_i \end{bmatrix} = \begin{bmatrix} H_i^\top & -H_i^\top & \cdots & H_i^\top & -H_i^\top \end{bmatrix}^\top, \qquad i = 1,\ldots,N, \\
b &= \begin{bmatrix} h_{i1}^\top & -h_{i1}^\top & \cdots & h_{iR}^\top & -h_{iR}^\top \end{bmatrix}^\top,
\end{aligned}
$$

where $\mathbf{1} \in \mathbb{R}^R$ and $\otimes$ denotes the kronecker product. Thus, Problem (4.14) is equivalent to

$$
\begin{aligned}
\min_{\substack{x_1,\ldots,x_N \\ \eta}} \quad & \sum_{i=1}^{N} c_i^\top x_i + d^\top \eta \\
\text{subj. to} \quad & \sum_{i=1}^{N} A_i x_i - b \le \eta \\
& \eta \ge \mathbf{0}, \;\; x_i \in X_i^{\text{MILP}}, \qquad i = 1, \ldots, N.
\end{aligned}
\tag{4.15}
$$

We would like to apply the results of Section 3.3 to Problem (4.15). However, we must find a way to deal with the additional variable η. By defining $\eta_1, \ldots, \eta_N \in \mathbb{R}^{2RK}$ such that $\sum_{i=1}^{N} \eta_i = \eta$ and each $\eta_i \ge 0$, we see that Problem (4.15) is equivalent to

$$
\begin{aligned}
\min_{\substack{x_1,\ldots,x_N \\ \eta_1,\ldots,\eta_N}} \quad & \sum_{i=1}^{N} (c_i^\top x_i + d^\top \eta_i) \\
\text{subj. to} \quad & \sum_{i=1}^{N} (A_i x_i - \eta_i) \le b, \\
& \eta_i \ge \mathbf{0}, \;\; x_i \in X_i^{\text{MILP}}, \qquad i = 1, \ldots, N,
\end{aligned}
\tag{4.16}
$$

in the sense that any solution of (4.15) can be reconstructed from a solution of (4.16) by using $\eta = \sum_{i=1}^{N} \eta_i$. Problem (4.16) has the constraint-coupled structure (4.11) by defining $\tilde{x}_i = (x_i, \eta_i) \in X_i^{\text{MILP}} \times [0, +\infty)^{2RK}$ as local optimization variables, with cost $\tilde{c}_i = (c_i, d_i)$ and coupling constraint matrices $\tilde{A}_i = [A_i, -I]$, where $I \in \mathbb{R}^{2RK}$ is the identity matrix. Note that Problem (4.16) has an unbounded feasible set (because of the variables η_i) but it always admits an optimal solution due to the terms $d^\top \eta_i$ minimized in the cost (recall that $d \ge \mathbf{0}$).

4.3.3 Distributed Algorithm Description

Using a technique similar to that of Section 3.3, we now propose a distributed algorithm that computes a feasible solution to (4.16). We begin by describing the algorithm and in the next subsection we discuss its theoretical properties.

The basic idea behind the distributed algorithm is the same as in Chapter 3, that is, to compute a mixed-integer solution starting from an optimal solution of the convex relaxation of Problem (4.15) obtained by replacing X_i^{MILP} with their convex hull

$\text{conv}(X_i^{\text{MILP}})$,

$$\begin{aligned} \min_{\substack{z_1,\dots,z_N\\ \eta_1,\dots,\eta_N}} \quad & \sum_{i=1}^{N} (c_i^\top z_i + d^\top \eta_i) \\ \text{subj. to} \quad & \sum_{i=1}^{N} (A_i z_i - \eta_i) \le b, \\ & \eta_i \ge \mathbf{0}, \;\; z_i \in \text{conv}(X_i^{\text{MILP}}), \qquad i = 1, \dots, N, \end{aligned} \tag{4.17}$$

where z_i denotes the continuous counterpart of the mixed-integer variable x_i. To do so, each agent i maintains a local allocation estimate $y_i^t \in \mathbb{R}^{2RK}$ (we recall from Section 2.2.2 that allocation vectors represent utilization of the total resource b of the coupling constraints). At each iteration t, the vector y_i^t is updated according to (4.18)–(4.19). After $T_f > 0$ iterations, the agent computes a tentative mixed-integer solution based on the last computed allocation estimate (cf. (4.20)). Algorithm 8 summarizes the steps from the perspective of agent i.

Algorithm 8 Distributed Stochastic Mixed-integer Microgrid Control

```
**Initialization**: set $T_f > 0$ and $y_i^0$ such that $\sum_{i=1}^N y_i^0 = b$
**Repeat** for $t = 0, 1, \dots, T_f - 1$
  **Compute** $\mu_i^t$ as a Lagrange multiplier of
      $$\begin{aligned} \min_{z_i,\eta_i} \quad & & c_i^\top z_i + d^\top \eta_i \\ \text{subj. to} \quad & \mu_i : & A_i z_i \le y_i^t + \eta_i \\ & & \eta_i \ge \mathbf{0}, \;\; z_i \in \text{conv}(X_i^{\text{MILP}}) \end{aligned} \tag{4.18}$$
  **Receive** $\mu_j^t$ from $j \in \mathcal{N}_i$ and update
      $$y_i^{t+1} = y_i^t + \alpha^t \sum_{j \in \mathcal{N}_i} \big(\mu_i^t - \mu_j^t\big) \tag{4.19}$$
**Return** $(x_i^{T_f}, \eta_i^{T_f})$ as optimal solution of
      $$\begin{aligned} \min_{x_i,\eta_i} \quad & c_i^\top x_i + d^\top \eta_i \\ \text{subj. to} \quad & A_i x_i \le y_i^{T_f} + \eta_i \\ & \eta_i \ge \mathbf{0}, \;\; x_i \in X_i^{\text{MILP}} \end{aligned} \tag{4.20}$$
```

Similarly to the algorithm discussed in Section 3.3.1, the first two steps (4.18)–(4.19) are used to compute an optimal solution of Problem (4.17), while the last step (4.20) reconstructs a mixed-integer solution. Note that Problem (4.18) is an LP and Problem (4.20) is a MILP. From a computational point of view, in order to compute a Lagrange multiplier of Problem (4.18) one can either use a local dual subgradient method (cf. Remark 3.3) or a local cutting-plane method (cf. Remark 3.8), while an optimal solution to Problem (4.20) can be found with any MILP solver. In the following, we do not discuss in every detail Algorithm 8 but only provide a qualitative analysis for those aspects that have been already discussed in Chapter 3. However, there are some novelties with respect to the framework of Chapter 3 for which we do provide formal results. In particular, in the next subsection we will prove a worst-case violation of the power balance constraints.

Remark 4.1. An important fact is that the computed mixed-integer solution always satisfies the power balance constraint of Problem (4.15) at the cost of a possibly high η_i, i.e.

$$\sum_{i=1}^{N}(A_i x_i^{T_f} - \eta_i^{T_f}) \leq \sum_{i=1}^{N} y_i^{T_f} = b,$$

where the inequality follows by construction and the equality follows by Proposition 4.2 (to be formulated in the next subsection). Thus, the algorithm can be stopped at any iteration $T_f \geq 0$ and the resulting solution will be feasible for the two-stage MILP (4.15). The greater the number of iterations, the higher is the optimality of the computed solution. $\triangle$

Before going on with the analysis, let us highlight some similarities and differences with respect to Algorithm 3 in Chapter 3. We begin by noting that here we are not employing the restriction-based mechanism (cf. Section 3.3.3). In Chapter 3 the restriction approach was used to guarantee satisfaction of the coupling constraints, while here this is not necessary since the computed solutions always satisfy the power balance constraint of the stochastic problem. This is a consequence of the fact that in the current chapter we assume that corrections actions (i.e. recourse) will be always necessary. Another difference regards the variables η_i. These vector variables play a role similar to the scalar variables ρ_i and v_i of Chapter 3 (cf. in particular Problems (3.5) and (3.7)), however here we prefer to use a different name to distinguish them from ρ_i and v_i. Indeed, in the cost terms $d^\top \eta_i$ the vector d is given, while in the cost terms $M\rho_i$ the weight M is artificially added and arbitrarily chosen. This is connected to the fact that in Chapter 3 the original objective (cf. Problem (3.1)) is to minimize $c_i^\top x_i$ for all i and to satisfy mandatorily the coupling constraints $\sum_{i=1}^{N} A_i x_i \leq b$, while now the original objective is to *jointly* optimize over x_i and η_i by minimizing $c_i^\top x_i + d^\top \eta_i$ for all i and to achieve only a "soft" satisfaction of the coupling constraints (due to the intrinsic presence of the violation vector η_i). This last consideration also has consequences on the formulation of the distributed algorithm. Indeed, the mixed-integer solution reconstruction is carried out differently from the strategy proposed in Chapter 3. In particular, we recall that Problem (3.7) is equivalent to a two-step procedure in which we first compute the needed (minimal) violation, if any, and then optimize the true cost function $c_i^\top x_i$. Instead, in Problem (4.20) we optimize over x_i and η_i simultaneously in order to faithfully reproduce the purpose of MILP (4.14).

Remark 4.2. In principle, Algorithm 8 can also be applied in a receding horizon fashion. Specifically, at each sample time k, one can run Algorithm 8 for a fixed number of iterations and then apply only the first input of the computed input trajectories. Since the control sampling interval is typically large (e.g. one hour), the number of iterations for which Algorithm 8 is executed can be very large, leading to quasi-asymptotic solutions that almost reach the performance specified in Theorem 4.1. $\triangle$

4.3.4 Theoretical Results

In this subsection, we provide theoretical results on Algorithm 8. The algorithm is similar to the one described in Section 3.3.1, however the previously highlighted differences result in a different kind of properties that can be proven. In particular, we will prove a bound for the worst-case violation of the asymptotically computed mixed-integer solution.

To begin with, we recall the next proposition, where we remind that K denotes the prediction horizon and R is the total number of scenarios in the stochastic problem).[1]

Proposition 4.1 (Lemma 3.1)**.** *Let Problem* (4.17) *be feasible and let* $(\bar{z}_1, \ldots, \bar{z}_N, \bar{\eta}_1, \ldots, \bar{\eta}_N)$ *be any vertex of its feasible set. Then, there exists an index set* $I_\mathbb{Z} \subseteq \{1, \ldots, N\}$*, with cardinality* $|I_\mathbb{Z}| \geq N - 2RK$*, such that* $\bar{z}_i \in X_i^{\text{MILP}}$ *for all* $i \in I_\mathbb{Z}$*.* $\triangle$

As seen in Chapter 3, the consequence of Proposition 4.1 is that at least $N - 2RK$ blocks of the mixed-integer solution computed asymptotically by Algorithm 8 are equal to the corresponding blocks of optimal solution of (4.17). Next we recall convergence of the steps (4.18)–(4.19). To this end, we denote as $(z_1^{\text{LP}}, \ldots, z_N^{\text{LP}}, \eta_1^{\text{LP}}, \ldots, \eta_N^{\text{LP}})$ an optimal solution of Problem (4.17), together with the allocation vector $(y_1^{\text{LP}}, \ldots, y_N^{\text{LP}})$ associated to the primal decomposition master problem (cf. Section 2.2.2), which we recall to be a vector satisfying

$$A_i z_i^{\text{LP}} - \eta_i^{\text{LP}} \leq y_i^{\text{LP}}, \quad \text{for all } i \in \{1, \ldots, N\}, \tag{4.21a}$$

[1]There are some minor differences with respect to the set-up considered in Chapter 3 and in particular between Problem (4.17) and the LP relaxation (3.2). The feasible set of Problem (4.17) is unbounded because of the variables η_i (which were not present in Problem (3.2). This invalidates the standing assumption of compact sets that we did in Chapter 3. However, Proposition 3.1 still applies because we can pretend that artificial constraints $\eta_i \leq M\mathbf{1}$, with $M > 0$ sufficiently large, are added to Problem (4.17) that make the feasible set compact and preserve the optimal solution set.

$$\text{and} \quad \sum_{i=1}^{N} y_i^{\text{LP}} = b. \tag{4.21b}$$

The following assumption is made on the step-size sequence.

Assumption 4.1. *The step-size sequence $\{\alpha^t\}_{t\geq 0}$, with each $\alpha^t \geq 0$, satisfies $\sum_{t=0}^{\infty} \alpha^t = \infty$, $\sum_{t=0}^{\infty} \left(\alpha^t\right)^2 < \infty$.* △

The following proposition summarizes the convergence properties of the steps (4.18)–(4.19). The proof is omitted since it is similar to the derivations of Chapter 2.[2]

Proposition 4.2. *Let Problem* (4.17) *be feasible and let Assumption 4.1 hold. Consider the allocation vector sequence $\{y_1^t, \ldots, y_N^t\}_{t\geq 0}$ generated by steps* (4.18)–(4.19) *of Algorithm 8 with the allocation vectors y_i^0 initialized such that $\sum_{i=1}^{N} y_i^0 = b$. Then,*

(i) $\sum_{i=1}^{N} y_i^t = b$ *for all* $t \geq 0$;

(ii) $\lim_{t\to\infty} \|y_i^t - y_i^{\text{LP}}\| = 0$ *for all* $i \in \{1, \ldots, N\}$. △

Because of Proposition 4.2, from now on we concentrate on the asymptotic mixed-integer solution computed by Algorithm 8. In particular, we denote by $(x_i^\infty, \eta_i^\infty)$ the optimal solution of Problem (4.20) with allocation equal to y_i^{LP}, i.e.

$$\begin{aligned} \min_{x_i, \eta_i} \; & c_i^\top x_i + d^\top \eta_i \\ \text{subj. to} \; & A_i x_i \leq y_i^{\text{LP}} + \eta_i \\ & \eta_i \geq \mathbf{0}, \; x_i \in X_i^{\text{MILP}}. \end{aligned} \tag{4.22}$$

We also define the lower bound of resources

$$\begin{aligned} \ell_i \triangleq \min_{x_i, \eta_i} \; & A_i x_i - \eta_i \\ \text{subj. to} \; & x_i \in \text{conv}(X_i) \\ & \mathbf{0} \leq \eta_i \leq M\mathbf{1}. \end{aligned}$$

where min is vector-wise and $M > 0$ is a sufficiently large number. Thus, it holds $\ell_i \leq y_i$ for all admissible allocations y_i, and in particular $\ell_i \leq y_i^{\text{LP}}$. Operatively, since the constraints on x_i and η_i are disjoint the vector ℓ_i can be computed by replacing $x_i \in \text{conv}(X_i)$ with $x_i \in X_i$.

In the next theorem we formalize the bound on the worst-case violation.

[2] To prove Proposition 4.2, one should apply primal decomposition to Problem (4.17) and then follow the reasonings of Section 2.4 without block randomization. Differently from Section 2.2, here we are not required to apply the relaxation approach, since the constraints $y_i \in Y_i$ are already not present in the master problem.

Theorem 4.1. *Let Problem* (4.17) *be feasible and consider the asymptotic mixed-integer solution* $(x_i^\infty, \eta_i^\infty)$ *computed by each agent* $i \in \{1, \ldots, N\}$*. Then, the worst-case violation of the power balance constraint is*

$$\sum_{i=1}^{N} A_i x_i^\infty - b \leq \sum_{i \in I_{\mathbb{Z}}} \eta_i^{\text{LP}} + \sum_{i \notin I_{\mathbb{Z}}} \frac{c_i^\top (x_i^{\text{L}} - x_i^\infty) + d^\top \eta_i^{\text{L}}}{d^{\text{MIN}}} \mathbf{1},$$

where $d^{\text{MIN}} = \min_{j \in \{1,\ldots,2RK\}} d_j$*,* $I_{\mathbb{Z}}$ *denotes the set of agents (satisfying* $|I_{\mathbb{Z}}| \geq N - 2RK$*) for which* $z_i^{\text{LP}} \in X_i^{\text{MILP}}$ *and* $(x_i^{\text{L}}, \eta_i^{\text{L}})$ *is an optimal solution of Problem* (4.24).

Proof. By the optimality of $(x_i^\infty, \eta_i^\infty)$ for Problem (4.22), it holds

$$c_i^\top x_i^\infty + d^\top \eta_i^\infty \leq c_i^\top x_i + d^\top \eta_i \tag{4.23}$$

for all $x_i \in X_i$ and $\eta_i \geq 0$ such that $A_i x_i^\infty \leq y_i^{\text{LP}} + \eta_i$. One vector satisfying such condition is $(x_i^{\text{L}}, \eta_i^{\text{L}})$ optimal solution of

$$\begin{aligned} \min_{x_i, \eta_i} \; & c_i^\top x_i + d^\top \eta_i \\ \text{subj. to } & \mathbf{0} \leq \eta_i \leq M\mathbf{1}, \;\; x_i \in X_i, \\ & A_i x_i \leq \ell_i + \eta_i, \end{aligned} \tag{4.24}$$

Indeed, it holds $A_i x_i^{\text{L}} \leq \ell_i + \eta_i^{\text{L}} \leq y_i^{\text{LP}} + \eta_i^{\text{L}}$, where the first inequality is by construction and the second one follows by the discussion above on ℓ_i. Thus, by using (4.23) we conclude that

$$d^\top \eta_i^\infty \leq c_i^\top (x_i^{\text{L}} - x_i^\infty) + d^\top \eta_i^{\text{L}}. \tag{4.25}$$

By explicitly writing the scalar product $d^\top \eta_i^\infty$ and by using the fact that $d, \eta \geq \mathbf{0}$ we obtain

$$d^\top \eta_i^\infty = \sum_{j=1}^{2RK} d_j \eta_{ij}^\infty \geq \underbrace{\Big(\min_{j \in \{1,\ldots,2RK\}} d_j \Big)}_{d^{\text{MIN}}} \sum_{j=1}^{2RK} \eta_{ij}^\infty.$$

Moreover, by using the fact that $\eta_{ij}^\infty \leq \sum_{k=1}^{2RK} \eta_{ik}^\infty$ for all k we obtain

$$\eta_i^\infty \leq \frac{d^\top \eta_i^\infty}{d^{\text{MIN}}} \mathbf{1} \leq \frac{c_i^\top (x_i^{\text{L}} - x_i^\infty) + d^\top \eta_i^{\text{L}}}{d^{\text{MIN}}} \mathbf{1}.$$

Let us now compute an upper bound of the coupling constraint value, i.e.

$$\begin{aligned}\sum_{i=1}^{N} A_i x_i^{\infty} - b &\leq \underbrace{\sum_{i=1}^{N} y_i^{\text{LP}}}_{b} + \sum_{i \in I_{\mathbb{Z}}} \eta_i^{\text{LP}} + \sum_{i \notin I_{\mathbb{Z}}} \eta_i^{\infty} - b \\ &= \sum_{i \in I_{\mathbb{Z}}} \eta_i^{\text{LP}} + \sum_{i \notin I_{\mathbb{Z}}} \eta_i^{\infty}. \qquad (4.26)\end{aligned}$$

where we used the fact that, by Proposition 4.1, for $i \in I_{\mathbb{Z}}$ it holds $A_i x_i^{\infty} \leq y_i^{\text{LP}} + \eta_i^{\text{LP}}$, while for $i \notin I_{\mathbb{Z}}$ it holds $A_i x_i^{\infty} \leq y_i^{\text{LP}} + \eta_i^{\infty}$. Thus we finally obtain the bound

$$\sum_{i=1}^{N} A_i x_i^{\infty} - b \leq \sum_{i \in I_{\mathbb{Z}}} \eta_i^{\text{LP}} + \sum_{i \notin I_{\mathbb{Z}}} \frac{c_i^\top (x_i^{\text{L}} - x_i^{\infty}) + d^\top \eta_i^{\text{L}}}{d^{\text{MIN}}} \mathbf{1}.$$

and the proof follows. ∎

Note that, since this bound is the sum of contributions of the agents, it can be computed a posteriori in a distributed way using a consensus scheme. To do so, they first need to detect whether they belong to $I_{\mathbb{Z}}$ or not by computing the primal solution z_i^{LP} of (4.18) and by checking whether it satisfies $z_i^{\text{LP}} \in X_i^{\text{MILP}}$. Then, they run the consensus scheme using as initial condition either $N\eta_i^{\text{LP}}$ (if $z_i^{\text{LP}} \in X_i^{\text{MILP}}$) or $N\frac{c_i^\top (x_i^{\text{L}} - x_i^{\infty}) + d^\top \eta_i^{\text{L}}}{d^{\text{MIN}}}\mathbf{1}$ (if $z_i^{\text{LP}} \notin X_i^{\text{MILP}}$).

4.3.5 Scenario Generation with Deep Generative Adversarial Networks

Let us provide more details regarding the scenario generation for renewable energy sources. Recall that $h \in \mathbb{R}^K$ is a random variable that depends on the total energy produced by the renewables (4.13e). The variable h has its own probability distribution and $h_1, \ldots, h_R \in \mathbb{R}^K$ are randomly drawn samples (cf. (4.14)). In order to generate such samples, we utilize a Generative Adversarial Network trained with real historical data. To train the neural network, we used the historical data provided by Open Power System Data [87], in particular we used the generation data of renewable energy sources in South Italy. We first filtered the dataset by concentrating only on summer months (to guarantee a certain uniformity of the data), removing spurious data and discarding days with missing information. We finally obtained a dataset with a total of 552 samples, where each sample contains information on the power produced during the day with a hourly resolution.

As for the utilized neural networks, the generative networks have a 10-dimensional input with the following layers:

- a dense layer with 1536 units, batch normalization and Leaky ReLU activation function;

- a layer that reshapes the input to the shape $(6, 256)$;
- a transposed convolution layer with 128 output filters, kernel size equal to 5, stride 1, batch normalization and Leaky ReLU activation function;
- a transposed convolution layer with 64 output filters, kernel size equal to 5, stride 2, batch normalization and Leaky ReLU activation function;
- a transposed convolution layer with 1 output filter, kernel size equal to 5, stride 2 and $\tanh$ activation function.

The ouput of the generative network is a 24-dimensional vector containing the power produced by the renewable unit at each time slot of the day. The discriminator networks have a 24-dimensional input with the following layers:

- a convolution layer with 64 output filters, kernel size equal to 5, stride 2 and Leaky ReLU activation function;
- a Dropout layer with rate 0.3;
- a convolution layer with 128 output filters, kernel size equal to 5, stride 2 and Leaky ReLU activation function;
- a Dropout layer with rate 0.3;
- a layer that flattens the input;
- a dense layer with one output unit.

The output of the discriminator networks is a scalar that denotes the probability that the evaluated input is a real one or a generated one.

We used the neural networks to generated samples of solar energy and wind energy. We used Tensorflow 2.4 to model the networks and we performed the training with 10^5 epochs. In Figure 4.3, we show example profiles of solar and wind energy generated by the networks.

4.3.6 Simulation Results

In this subsection, we show simulation results of Algorithm 8 performed with the disropt package [48].

We generated a microgrid control problem with a total of 48 units: 5 generators, 5 storages, 15 controllable loads, 10 critical loads, 10 solar generators, 3 wind generators and the connection to the main grid. We considered $R = 5$ scenarios with a 24-hour prediction horizon and 1-hour sampling time. We used the data provided by [87] for the load profiles and for the daily spot prices, which are shown in Figure 4.4.

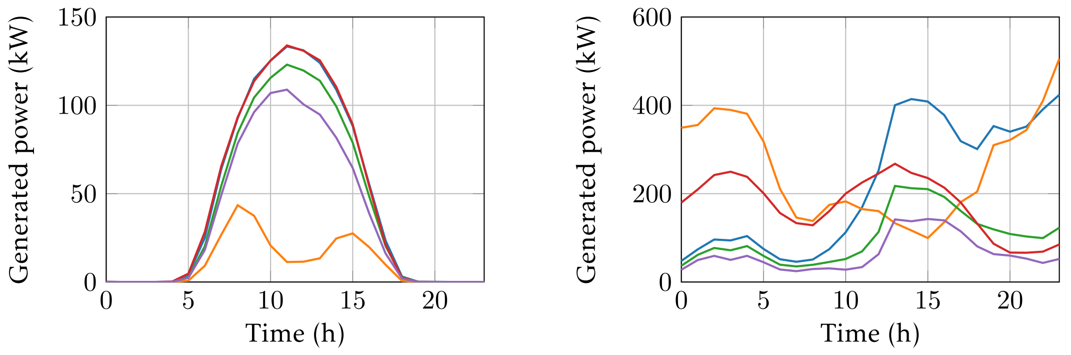


Figure 4.3: Power generation profiles (5 scenarios) generated by the GANs. Left: solar energy, right: wind energy.

We executed Algorithm 8 for 1000 iterations with constant step size $\alpha = 3$. In Figure 4.5, we show the total consumed power and the total curtailed power. In Figure 4.6, we show the total power exchanged with storage units (a positive value means that, overall, the storage units are charging) and the global level of stored power. In Figure 4.7, we show the total power exchanged with the utility grid (a positive value means that power is purchased from the grid). In Figure 4.8, we show where does the total available power comes from. In particular we highlight the fraction of power coming from generators, renewables and the utility grid. In this simulation, the generators did not produce any energy.

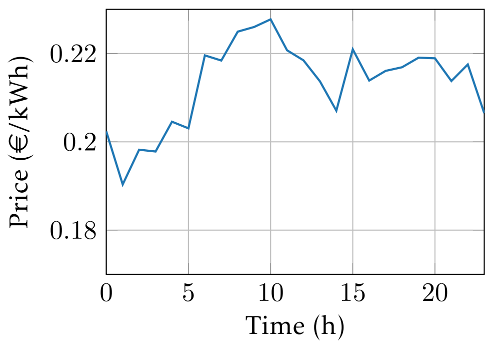


Figure 4.4: Daily spot prices from Open Power System Data [87].

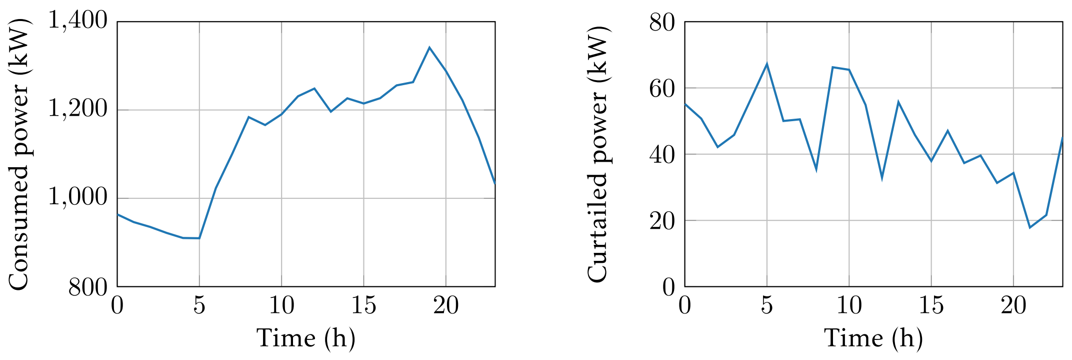


Figure 4.5: Total consumed power (critical and controllable loads) and curtailed power (for controllable loads only).

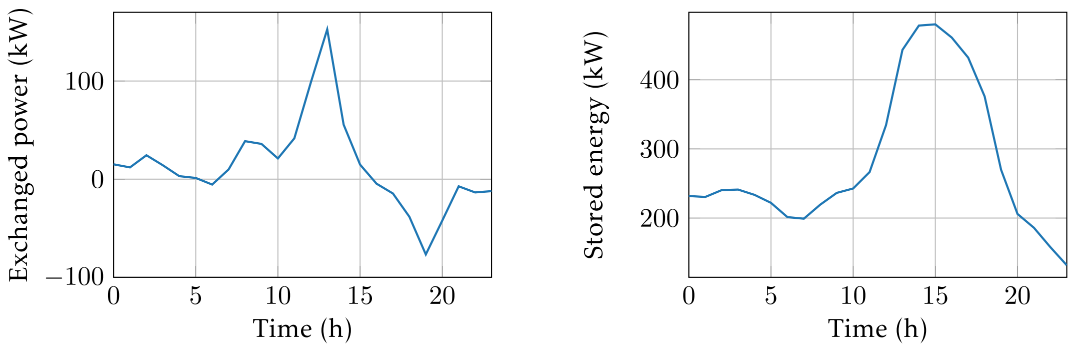


Figure 4.6: Total average power exchanged by storage units (left) and level of total stored power (right).

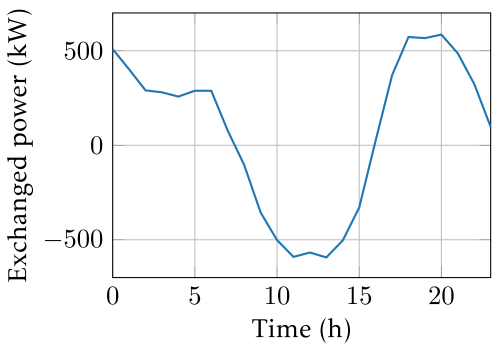


Figure 4.7: Total power exchanged with the utility grid.

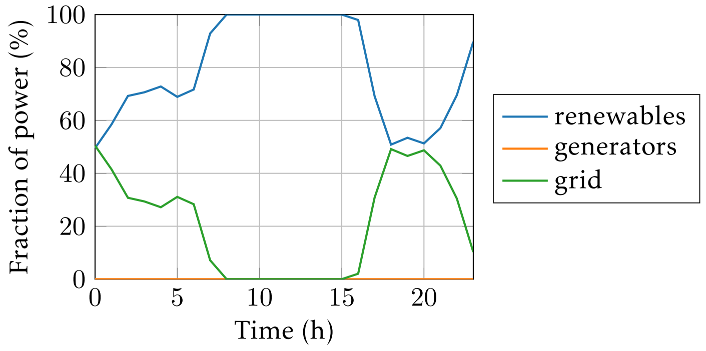


Figure 4.8: Fraction of consumed power coming from generators, renewables and utility grid at each time slot.

Chapter 5

Cooperative Robotics Toolbox and Distributed Vehicle Routing

In this chapter, we show applications of the methods developed in the previous chapters to a distributed multi-robot scenario. To this end, we first develop a software package, named ChoiRbot, which is a general ROS 2 platform for the implementation of cooperative robotics scenarios that can be used for several applications. We describe the architecture of the developed software and provide a set of simulations and experiments to illustrate its usage. Finally, we consider a multi-robot Pickup-and-delivery Vehicle Routing Problem to which we apply the methodologies developed in the previous chapters and show simulation and experimental results. The material of this chapter is based on [28, 48, 115].

5.1 Literature Review

We divide the literature review in two blocks. First, we review literature related to software toolboxes for cooperative robotics. Since its introduction, the Robot Operating System (ROS), [95], has gained popularity among robotics researchers as an open source framework for the development of robotics applications. Nowadays, ROS 2 is extending ROS capabilities, paving the way to real-time control systems and large-scale distributed architectures, [69]. While several theoretical frameworks have been proposed to solve optimization problems over networks of cooperating robots, see, e.g. the survey [86] and references therein, few architectures have been proposed to simulate and run experiments on teams of heterogeneous robots communicating according to arbitrary graphs and aiming at the cooperative solution of complex tasks. Authors in [2] propose a constraint-based task specification for robot controllers, defining a task specification language and the related controller. The framework in [93] allows users to create task plans for collaborative robots. Papers [55] and [73] instead propose architectures to simulate and control UAVs, while [30] allows users to write ROS code on a browser and run it on remote robots. On this purpose, it is worth mentioning the Robotarium, [124], a proprietary platform that allows to test and run control algorithms on robotic teams. Finally, in the recent years, robotics researchers started to develop robotic architectures based on the novel ROS 2 framework. Authors in [42] and [43] consider a framework for collaborative robotics, while the paper in [98] discusses an architecture for self-driving cars. The aforementioned frameworks are typically optimized for a specific task, and most of them focus on single-robot systems. Moreover, the communication in multi-robot networks is often neglected or simulated by means of the resource-demanding all-to-all communication.

Second, we review literature related to vehicle routing problems. Several algorithms have been proposed to solve the problem to (sub)optimality in centralized settings, we refer the reader to the surveys [92, 94, 101, 118] for a comprehensive list of these methods in static and dynamic settings. Online, dynamic approaches have been proposed, see e.g. [36], taking into account collision avoidance constraints among robotic agents, [67]. In order to overcome the drawbacks of centralized approaches, as e.g. the high computational complexity of the problem, a branch of literature analyzes schemes based on master-slave or communication-less architectures. In master-slave approaches, implementations of the parallel auction based algorithm are among the most used strategies, see e.g. [50, 57]. As for communication-less approaches in which agents do not communicate with each other, we refer the reader to [4] for a dynamic pickup and delivery application. Few works address the solution of vehicle routing problems in peer-to-peer networks. Indeed, the majority of the approaches in literature address problems which are approximations or special versions of the Pickup-and-Delivery. Authors in [19, 35, 103] solve unimodular task assignment problems by means of convex optimization techniques. The works [68, 116] address generalized assignment problems, where the order of execution of the tasks does not have impact on the objective. In [117], authors address a multi-agent multi-task assignment problem where agents-to-tasks path are evaluated offline and capacity constraints are not considered. Finally, the distributed auction-based approach, see [17, 113] for recent applications, is often used to address task allocation problems. These approaches do not address more complex models as e.g. capacity and time demands of the tasks. As for distributed approaches to vehicle routing problems, authors in [49] propose a distributed algorithm where agents communicate according to cyclic graph and perform operations one at a time. In [1], a distributed scheme in which agents iteratively solve graph partitioning problems is proposed, and is shown to converge to a suboptimal solution of a dynamic vehicle routing problem.

5.2 ChoiRbot: A ROS 2 Framework for Cooperative Robotics

In this section, we describe ChoiRbot, a toolbox for distributed cooperative robotics based on the novel Robot Operating System (ROS) 2. We also discuss a simple experiment by which we show how the constraint-coupled convex set-up can be used to model a barycenter-constrained position computation scenario, see Section 5.2.6. The material of this section is based on [115].

The package is available at `https://github.com/OPT4SMART/` under the GNU GPL license, with a complete documentation and many examples.

5.2.1 Architecture Description

Let us describe the high-level architecture of ChoiRbot. The toolbox is modular and its blocks are intended to be combined as needed. We first focus on an overall description of the software and then we describe each block separately.

Overview of the Software

ChoiRbot is written in Python and is based on the Robot Operating System (ROS) 2. An important feature of the new ROS 2 architecture is its ability to run programs in a totally distributed fashion, i.e. without a server acting as message broker [69]. This fits perfectly with the goal of ChoiRbot, namely to provide a platform for cooperative robotics over peer-to-peer networks without a central coordinating unit.

The toolbox is structured in a three-layer architecture. Specifically, there is a *Team Guidance* layer, a *RoboPlanning* layer and a *RoboControl* layer. The Team Guidance layer is responsible for taking high-level decisions and for managing the robot lifecycle. The Team Guidance layer uses communication with neighbors in order to perform its tasks. The Roboplanning and Robocontrol layers are responsible for lower-level control actions as driven by the upper layer. Since our main goal is to ease the design and implementation of optimization-based distributed control schemes, the central focus of ChoiRbot is the Team Guidance layer. In particular, the toolbox provides boilerplate code for distributed computation, i.e. communication among robots over a graph, coordination and optimization algorithms. Apart from a few specific scenarios, we do not provide comprehensive RoboPlanning and RoboControl features, which are robot specific and can be already implemented with existing tools.

In ChoiRbot, each robotic agent in the network executes three ROS 2 processes, one for each layer. Each process is also associated to a separate ROS 2 node. To guarantee flexibility and code reusability, layers are implemented as Python classes. In the next we provide details about the three layers. A graphical illustration of the software architecture is represented in Figure 5.1.

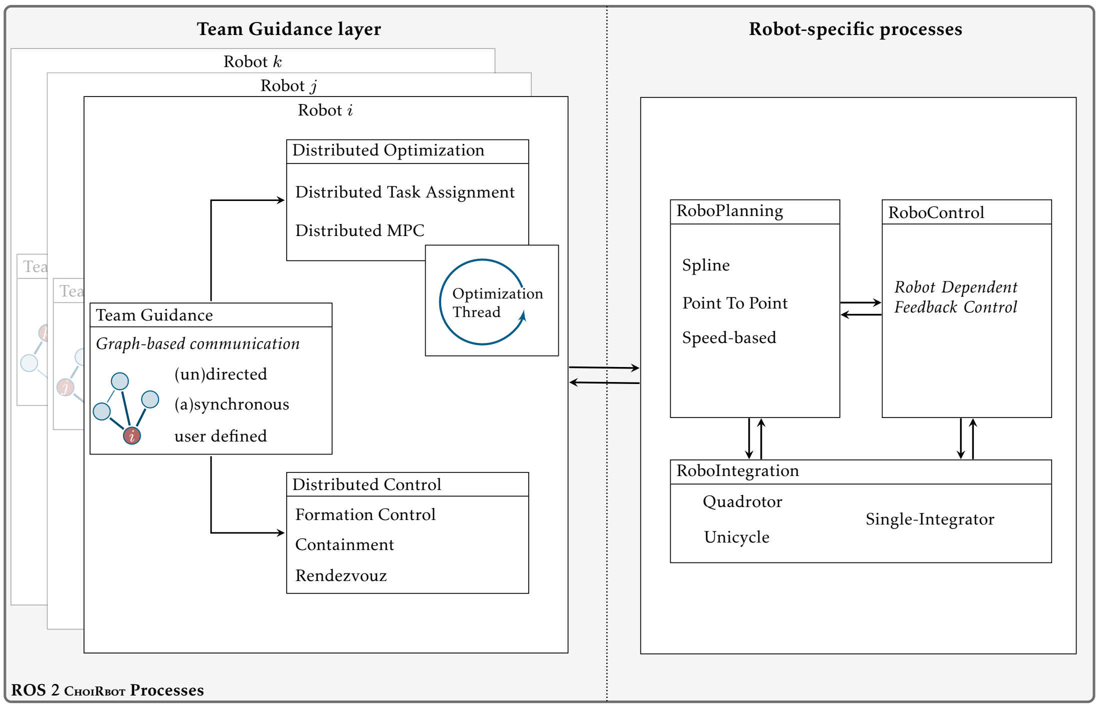


Figure 5.1: ChoiRbot architecture. Each robot communicates with neighboring robotic agents thanks to the Team Guidance Layer. The Team Guidance class handles the communication with planning and control utilities, which are specific for each robot.

Team Guidance Layer

The Team Guidance layer is the main entry point of the package. Here we provide an introductory description, while a more detailed analysis of this part of software is delayed to Section 5.2.2.

The Team Guidance layer is modeled with a `Guidance` class, whose purpose is only to retrieve the basic information regarding the robotic agent (passed as ROS 2 parameters), to create an instance of the `Communicator` class to be described in Section 5.2.2 and to subscribe to the topic where the robot pose is published. In the current version of ChoiRbot, the robot position can be either communicated by a Vicon motion tracking system or by a simulator (such as Gazebo). This basic version of the `Guidance` class is abstract in that it does not implement any guidance logic. However, the package currently provides two possible usable extensions. The first one (implemented as the `OptimizationGuidance` class) also provides optimization-related functionalities and is the starting point for any optimization-based distributed control scheme such as task assignment / allocation, optimal control, model predictive control. The toolbox can also be used to implement simpler distributed feedback laws that do not require the solution of optimization problems. To this end, we also provide a second extension of the `Guidance` class (implemented as the `DistributedControlGuidance` class), where robots repeatedly exchange their current position with their neighbors and compute a control input based on their position and the received positions. The actual form of the control input depends on the specific scenario and is left as an unimplemented method of the class. Currently implemented algorithms are rendezvous, containment, formation control.

RoboPlanning, RoboControl and RoboIntegration

Within the ChoiRbot toolbox, the trajectory planning layer and the control layer do not communicate with the neighboring robotic agents. Instead, each robot has its own planning/control stack, which depends on the dynamics of the single robot and on the chosen control strategy. The base class for trajectory planners is `Planner`. We provide a point-to-point planner (in 2D or 3D), to be used in conjunction with controllers that are able to steer the robot towards a specified point in the space (e.g. for mobile ground robots). We also provide trajectory planners for aerial robots, which generate quad-rotor trajectories either to reach a desired constant speed or to reach a target point. As for controllers, the base class is `Controller`. For mobile ground robots, we provide two unicycle controllers that can be used either to reach a target point or to reach a desired velocity. For quadrotors, we provide a controller to stabilize the trajectories generated by the planners.

The ChoiRbot toolbox is well integrated with simulation environments. In this regard, the `Gazebo` simulator [62] can be used. In case the user does not want to use external tools, we also provide a dynamics integration layer, named *RoboIntegration*, which can e.g. be used in conjunction with `Rviz` for visualization. Currently, there are numerical integrators for point-masses, unicycle robots and quadrotors. New integrators with custom dynamics can be written by extending the `Integrator` class.

5.2.2 Exploring the Team Guidance Layer

As already mentioned, almost all functionalities of ChoiRbot involve the Team Guidance layer. In this section, we explore in more detail this part of software. We begin by analyzing the first important feature of the Team Guidance class, i.e. graph-based communication. Then, we analyze the two essential usages of the Team Guidance layer, represented by the `OptimizationGuidance` and the `DistributedControlGuidance` classes.

Graph-based Communication

At the core of every distributed control scheme is the graph-based communication among the robotic agents. This feature is provided in ChoiRbot at the team guidance layer by the `Communicator` class, which can handle both synchronous/asynchronous and undirected/directed communication among the robots. To achieve this, the `Communicator` class only requires specification of the in- and out-neighbors of the robots and takes care of managing the necessary ROS 2 topics and subscriptions. To maintain the class interface of the class semantically clear, the method names correspond to the specific actions that can be performed: `send`, `receive`, `asynchronous_receive`, `neighbors_exchange` (i.e. simultaneous `send`/`receive`).

An important feature is that it is not necessary to declare the message types to be exchanged among the robots, which is typically needed for all ROS applications. Instead, we fixed the type of message to `std_msgs/ByteMultiArray` and the `Communicator` class handles (de)serialization of the exchanged message to (from) a byte sequence through the Python `dill` package. This allows the user to exchange nearly each type of message (vector, matrices, text, dictionaries, images, etc.) and even to change it at runtime without declaring what type of message must be sent.

Distributed Optimization

A consistent part of the toolbox is devoted to providing optimization-related functions. The main entry point is the `OptimizationGuidance` class. Since numerical optimization algorithms may require a certain number of iterations to converge, these computations are delegated to a separate thread running in the `OptimizationThread` class. This allows the ROS 2 guidance process to continue elaborating callbacks even though an optimization is in progress. This separate thread class, which is started by the `OptimizationGuidance` class, allows the user to start/stop the optimization processes on demand. At the end of an optimization process, the method `optimization_ended` is called. This method is supposed to be overridden by the user with the specific guidance logic, e.g. by retrieving the optimization results and by using them as needed. The actual body of the optimization algorithm is left unimplemented so as to allow the user to implement the desired method. Currently, we provide implementations for distributed task assignment and distributed MPC scenarios, which are analyzed in detail in Section 5.2.4.

As already mentioned, the toolbox supports running both local optimization algorithms at a robot and distributed optimization algorithms involving the whole network. To this end, ChoiRbot integrates with the disropt package [48], which is used to model and to solve both local and distributed optimization problems with several algorithms.

Distributed Feedback Control

The toolbox is designed to support optimization-based distributed control algorithms, however simpler distributed feedback laws can also be implemented by using a subset of the toolbox features. The `DistributedControlGuidance` class implements a general communication and control structure allowing for the implementation of such feedback laws. To give an idea, let us report an excerpt of the main routine of the class, which is executed with a user-chosen frequency:

```
# exchange current position with neighbors
data = self.communicator.neighbors_exchange(
          self.current_pose.position,
          self.in_neighbors,
          self.out_neighbors, False)

# compute input
u = self.evaluate_velocity(data)

# send input to planner/controller
self.send_input(u)
```

In words, the class first exchanges the current position with the neighbors, then computes a velocity profile according to the exchanged data and to the considered feedback law, and finally publishes the control input on a topic. Despite its simplicity, this basic structure can be used for several distributed control algorithms such as containment and formation control. Simulation results can be found in Section 5.2.6.

5.2.3 Distributed Optimization via DISROPT

Distributed optimization schemes can be implemented in ChoiRbot by using the functionalities provided by the disropt package. disropt is a Python toolbox for distributed optimization with MPI-based communication. It provides a set of libraries to model optimization problems (i.e. cost function, constraints) and to run distributed optimization algorithms with a simple syntax. In the package, several distributed optimization algorithms have been implemented for three different set-ups, namely the cost-coupled set-up, the common-cost set-up and the constraint-coupled set-up, see [86]. A description of the package can be found in [48].

In its standalone version, disropt can be used to run distributed optimization algorithms over MPI-based communication networks, modeled with a `Communicator` class. However, the `Communicator` class of ChoiRbot provides the same features by using ROS-based communication and is fully compatible with disropt. Thus, in order to run distributed optimization algorithms over ChoiRbot, one can simply write the code as usual and then rely on the ChoiRbot communication capabilities to execute the distributed algorithm.

5.2.4 Implemented Complex Scenarios

Let us discuss in detail two complex cooperative robotic scenarios that have been implemented in ChoiRbot, i.e. distributed dynamic task assignment and distributed model predictive control. In both scenarios we employ the `OptimizationGuidance` class to leverage optimization capabilities.

Distributed Dynamic Task Assignment

In this scenario, we assume there is a set of tasks that must be performed by a team of robots. In order to choose the final assignment, robots must self-coordinate by using their communication capabilities. A formulation of the problem can be found in Section 1.3.2. We focus on the challenging scenario in which tasks are not known a-priori but arrive dynamically. While robots perform the previously assigned tasks, a new task can arrive and robots re-execute the distributed optimization algorithm to compute the new optimal assignment.

Implementation In ChoiRbot, there is a number of classes to handle the dynamic task assignment scenario. We assume there is a cloud accepting task requests and forwarding them to the robots such as, e.g. in online order and delivery services. This mechanism is implemented with the `TaskTable` class, which is also responsible for maintaining an up-to-date task list and to mark tasks as completed. There is a `TaskGuidance` class that communicates with `TaskTable`, triggers the distributed optimization algorithm in the separate computation thread whenever a new task request is received and maintains a local task queue. While the queue is not empty, tasks are executed in order by calling the `TaskExecutor` class, which implements the specific task execution logic (e.g. to move to a given point and to perform loading/unloading operations). The separate computation thread is implemented with the `TaskOptimizationThread` class, in which methods from the `TaskOptimizer` class are called to handle the distributed optimization process. The `TaskOptimizer` class is responsible for formulating the actual linear programming task assignment problem and to run the distributed simplex algorithm [19] in disropt. In this scenario, communication among neighbors occurs within the `TaskOptimizer` class during the distributed optimization process.

Remarkably, from the point of view of the final user, all is needed is to implement the task execution logic in `TaskExecutor` and to integrate `TaskTable` with the task request service of the specific application scenario. The whole dynamic task assignment mechanism is taken care of by ChoiRbot. In Section 5.2.6, experimental results on this scenario are provided.

Distributed Model Predictive Control

Let us now describe the second complex scenario implemented in ChoiRbot. In this scenario, we assume there are N robots with linear dynamics that aim to apply a distributed Model Predictive Control scheme. A formulation of the problem can be found in Section 1.3.1.

Implementation We implemented the classical distributed MPC algorithm in [99]. The departing point is the `MPCGuidance` class, which implements the actual steps of the distributed MPC algorithm. The problem data (i.e. system matrices, local constraints, coupling constraints, prediction horizon) are provided by the user as class parameters. This class interacts with an `MPCOptimizationThread` class, which calls the `optimize` method from the `MPCOptimizer` class to solve the local optimal control problem at each control iteration. The `MPCOptimizer` class is responsible for formulating the local optimal control problem and for solving it by using the problem modeling and solving capabilities available in disropt. Note that, differently from the dynamic task assignment, in this scenario the solution of optimization problems is completely local. Here, communication among neighbors occurs within the `MPCGuidance` class, as required by the MPC algorithm (see [99] for details).

5.2.5 Basic Usage Example

In this section, we consider a toy example that allows us to show the implementation of a basic distributed cooperative robotics scenario in ChoiRbot. Specifically, we consider containment in leader-follower networks as described in [85]. The mathematical formulation is as follows. Robots communicate according to an undirected connected graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1, \dots, N\}$ is the set of vertices and $\mathcal{E} \subseteq V \times V$ is the set of edges. We denote by $\mathcal{N}_i$ the set of neighbors of each robot i. Robots are partitioned in two groups, namely leaders and followers. The goal for the followers is to converge to the convex hull of the leaders' positions. To this end, the robots implement the dynamics

$$\dot{x}_i(t) = 0 \qquad \text{(leaders)}, \tag{5.1a}$$

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} (x_j(t) - x_t(t)) \qquad \text{(followers)}. \tag{5.1b}$$

Main Software Components

This scenario can be easily implemented in the proposed architecture by means of two components, namely a `ContainmentGuidance` class and a `SingleIntegrator` class. The first one is an extension of the `DistributedControlGuidance` class discussed in Section 5.2.2 and inherits the distributed feedback control structure. Thus, in the `ContainmentGuidance` class one can simply override the `evaluate_velocity` method in order to implement the control law in (5.1) as follows:

```
u = np.zeros(3)
if not self.is_leader:
    for pos_ii in neigh_data.values():
```

```
    u += self.containment_gain*(pos_ii - self.current_pose.position)
return u
```

The `SingleIntegrator` class instead extends the functionalities of the base class `Integrator`. In particular, it is sufficnet to override the (initially empty) `integrate` method to implement the classical forward Euler method for single-integrator dynamics:

```
self.current_pos += self.samp_time * self.u
```

Visualizing the Evolution

Since we are not using an external simulation tool, we need a means to visualize the results graphically. To achieve this, it is possible to rely on the ROS 2 toolbox Rviz, in which we associate each robot with a circle moving on the (x, y) plane. To achieve this, there is a class provided by the toolbox named `RvizSpawner`, which receives the pose published by the `SingleIntegrator` and forwards it to the Rviz visualizer with a user-defined frequency.

Running the Simulation

The final simulation can be run by writing a ROS 2 launch file with all the required nodes, i.e. `ContainmentGuidance`, `SingleIntegrator`, `RvizSpawner`. Note that these three nodes must be executed by each of the robotic agents in the network. The launcher file is also responsible for declaring the initial positions of the robots and the communication graph, specified as a binary adjacency matrix.

In order to differentiate the ROS 2 nodes of each robot, robot-specific parameters must be passed as ROS 2 parameters at the time of spawing. For instance, the `ContainmentGuidance` classes are spawned in the launch file as follows

```
def generate_launch_description():
  # .. initialization of adjacency matrix and initial positions

  list_description = []
  for i in range(N):
    # .. initialize in_neighbors, out_neighbors, is_leader

    list_description.append(Node(
      package='choirbot_examples', node_executable='choirbot_containment',
      node_namespace='agent_{}'.format(i),
      parameters=[{'agent_id': i, 'N': N, 'in_neigh': in_neighbors, 'out_neigh':
           out_neighbors, 'is_leader': is_leader}]))

  return LaunchDescription(list_description)
```

A simulation with e.g. $N = 6$ robots can be run by executing the following command

```
ros2 launch choirbot_examples containment.launch.py -n 6
```

5.2.6 Toolbox Validation in Simulations and Experiments

In this section, we provide simulation and experimental results for three different scenarios that can be handled by ChoiRbot. We begin by showing experimental results of the dynamic task assignment scenario described in Section 5.2.4. Then, we show how to the proposed package can be easily interfaced with Gazebo or Rviz and show simulation results for a team of mobile wheeled robots.

Dynamic Task Assignment on Turtlebot3 Mobile Robots

In this experiment, we consider a team of four Turtlebot 3 Burger mobile robots that have to accomplish a set of task scattered in the environment. A task is considered accomplished if the designed robot reaches its position on the $\{x, y\}$ plane. As in real applications, problem data are not completely known a-priori, we consider a dynamic assignment problem where new data arrive during the execution. Thus, robots have to re-optimize and adjust their local planning whenever new information is available. Inspired by the approach in [35], we assume that a new task is revealed as soon as one has been completed. The pose of each robot is retrieved by communicating with a centralized camera infrastructure, in particular a Vicon Motion capture system. In order to interface ChoiRbot with the Vicon system, we developed an ad-hoc ROS 2 package, which is provided in the toolbox repository.

Thanks to the flexibility of ChoiRbot and of ROS 2, one can either execute the ROS nodes on the robots or on an external computed connected to the same Wi-Fi network. In our experiments, since the robots are not equipped with high computational power, we prefer to leave onboard the RoboPlanning and the RoboControl nodes and to execute the Team Guidance layers on a separate computer. Each time a new task is revealed, robots execute the distributed simplex algorithm until convergence, which occurs within less than 1 second. As for the control layer to steer robots to the desired positions, we implemented a dedicated controller node executing the control law in [91]. In Figure 5.2, we report a snapshot from the experiment in which the robots are servicing a set of tasks, while Figure 5.3 reports a Gantt chart of the actual task execution.

Formation Control for Unicycle-Like Robots

In this scenario, the goal is to drive robots to a translationally independent formation in the $\{x, y\}$ plane that satisfies a set of given constraints. These constraints are specified by the communication graph edge set $\mathcal{E}$ and a set $\{d_{ij}\}_{(i,j)\in\mathcal{E}}$ of desired distances between two communicating robots i and j, with the requirement $d_{ij} = d_{ji}$ for all $(i, j) \in \mathcal{E}$. We refer the reader to, e.g. [72] for a detailed description. Let the robots apply the

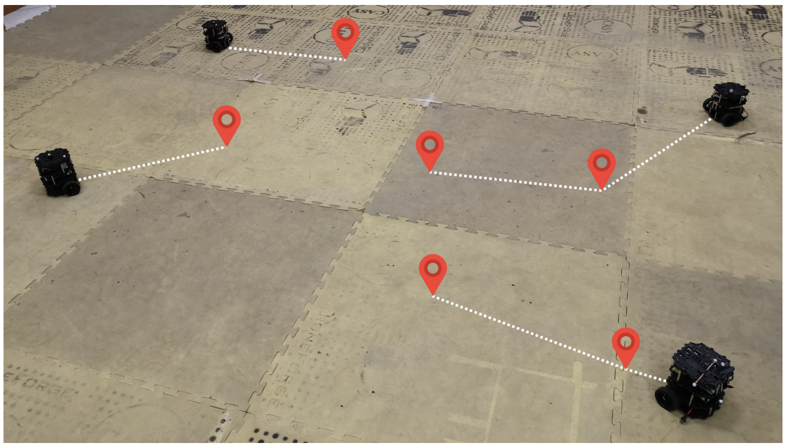


Figure 5.2: Snapshot from the dynamic task assignment experiment. Robots move in order to reach their designed tasks (red markers).

distributed control law

$$u_i(t) = \sum_{j \in \mathcal{N}_i} (\|x_i(t) - x_j(t)\|^2 - d_{ij}^2)(x_j(t) - x_i(t)). \tag{5.2}$$

We extended the `DistributedControlGuidance` and override the `evaluate_velocity` method in order to implement the control law in (5.2). However, this is not directly implementable on unicycle-like robots. Thus, we developed a specific controller, named `UnicycleVelocityController`, that maps the input provided by (5.2) to a suitable set of inputs for wheeled robots. This is performed following the approach described in [124]. The interfacing with Gazebo is straightforward. Robot poses are retrieved directly by the odometry topic maintained by Gazebo, while robot inputs are sent on suitable topics read by Gazebo plugins. In Figure 5.4 a snapshot from a Gazebo simulation in which six Turtlebot 3 Burger robots have to draw an hexagon.

Containment

Finally, let us show simulation results for the containment example outlined in Section 5.2.5. We consider a scenarios with three followers and three leaders spanning a triangle. The evolution of the robot positions is reported in Figure 5.5.

5.2.7 Distributed Primal Decomposition for Mobile Robots

The ChoiRbot platform allows us to show how constraint-coupled optimization can find applications to distributed multi-robot scenarios. Consider $N = 4$ ground robots. Assume the robots aim to compute optimal target positions inside local circular constraints while keeping the barycenter within certain bounds. This task can be modeled

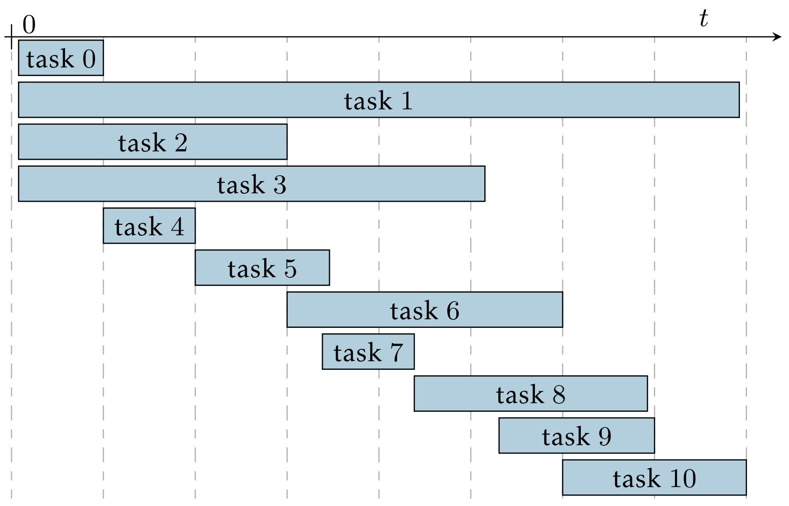


Figure 5.3: Gantt chart of the task execution flow in the dynamic task assignment. The horizontal axis represents time with the actual scale. The left side of each rectangle indicates the beginning of a task, while the right indicates that the task has been serviced by a robot. As described above, when a task is serviced (e.g. task 0) a new one begins (e.g. task 4)

as the convex constraint-coupled problem

$$\begin{aligned} \min_{x_1,\dots,x_4} \quad & \sum_{i=1}^{4} \|x_i - r_i\|_{Q_i}^2 \\ \text{subj. to} \quad & \|x_i\| \leq R_i, \qquad i = 1,\dots,4, \\ & b^{\text{MIN}} \leq \frac{1}{N} \sum_{i=1}^{4} x_i \leq b^{\text{MAX}}, \end{aligned} \tag{5.3}$$

where each $x_i \in \mathbb{R}^2$ is the local decision variable, $r_i \in \mathbb{R}^2$ is the ideal target position, $Q_i \in \mathbb{R}^{2\times 2}$ is a symmetric positive definite cost matrix, $R_i > 0$ is the radius of the local circular constraint, while the vectors b^{MIN} and b^{MAX} define the bounds for the barycenter of the target positions.

We consider a "static" scenario, in which the robots must satisfy the barycenter bounds only at the target position, however in principle this approach can be extended to the case in which the problem is solved in real time (i.e. online). To implement this scenario with ChoiRbot, we implemented a Team Guidance class to let the robots solve Problem (5.3) with the Distributed Primal Decomposition algorithm (cf. Section 2.2.5). Then, we generated a random problem and we executed the algorithm on a team of Turtlebot 3 Burger robots. In Figure 5.6, we show snapshots from the experiment.

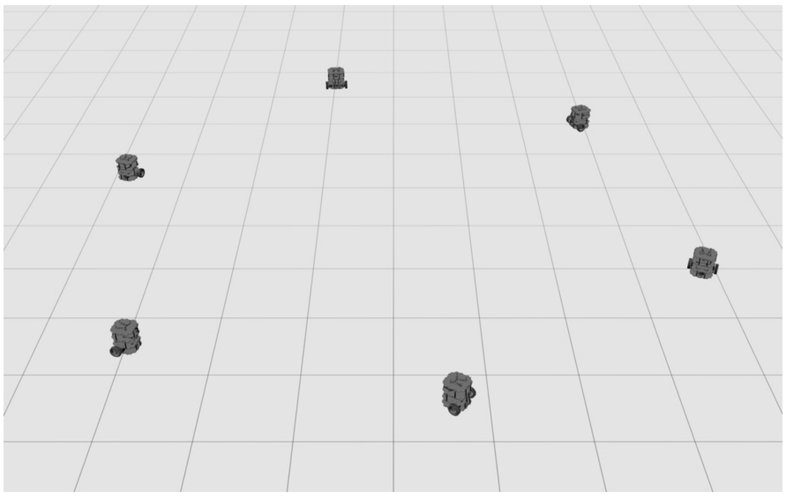


Figure 5.4: Formation control simulation via Gazebo. By leveraging on ChoiRbot functionalities, robots reach the desired hexagonal formation.

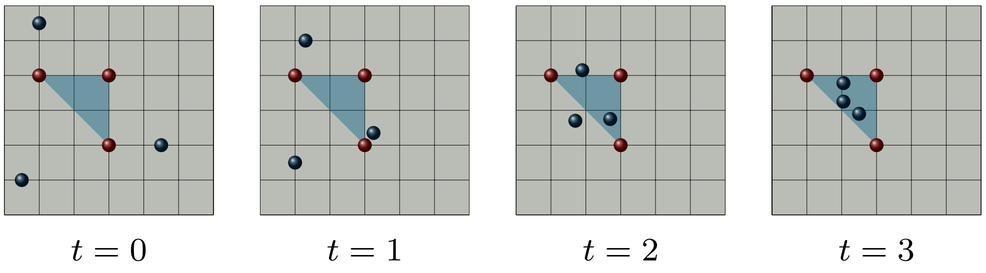


Figure 5.5: Sequence of images from the Rviz toolbox at different subsequent time instants. Red spheres represent leader robots, while the blue ones represent the followers. As time progresses, followers enter the convex hull of the leaders' positions, depiced with the cerulean triangle.

5.3 Distributed Multi-Robot Pickup and Delivery

In this section, we apply the methodology of Chapter 3 to a *Pickup-and-Delivery Vehicle Routing Problem* (PDVRP) to be solved in a distributed multi-robot scenario. The pickup and delivery problem models several applications such as battery exchange in robotic networks, [60], pickup and delivery in warehouses, [56], task scheduling [54] and delivery with precedence constraints [5].

We first provide a mathematical formulation of the optimization problem together with a thorough description of its structure. Then, we apply the results of Chapter 3 and use specific problem properties to simplify the algorithm. Finally, we show simulation and experimental results performed with the ChoiRbot platform. The results of this section are based on [28].

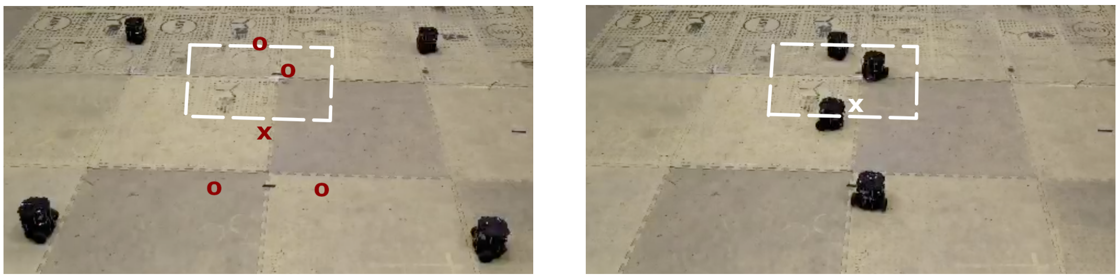


Figure 5.6: Experimental results of constraint-coupled optimization for mobile robots. Left: robots start the mission. The ideal target positions r_i are denoted with red circles and their barycenter with a red cross. The barycenter of the ideal positions is not contained in the bounds. Right: robots determine a feasible set of target positions and reach them. The resulting barycenter is indicated with a white cross, which satisfies the bounds.

5.3.1 Problem Description

The (Multi-Vehicle) Pickup-and-Delivery Vehicle Routing Problem can be described as follows. A group of vehicles has to fulfill a set of transportation requests. Requests consist of picking up goods at some locations and delivering them to other locations. The problem then consists of determining minimal length paths such that all the requests are satisfied. To achieve this, we can assign to each location a label "P" (pickup) or "D" (delivery), and then define a graph of all possible paths that can be traveled by vehicles. In Figure 5.7, we show an example scenario with two vehicles, two pickups and two deliveries. Note that, in order to have a well-defined graph, vehicles must start

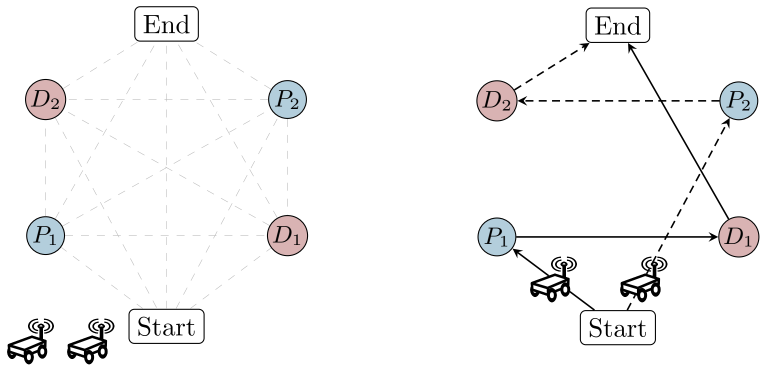


Figure 5.7: Example PDVRP scenario. Left: vehicles begin from the "start" node and must end at the terminal node. There are two pickup requests P_1 and P_2 and two associated deliveries D_1 and D_2. Right: the optimal path consists of the first vehicle traveling through P_1 and D_1 and the second vehicle traveling through P_2 and D_2.

from an initial node representing the initial position (which may be different for each of them) and have to reach a target node. This can be also "virtual" in the sense that, once the last delivery position has been reached, the vehicle stops there and wait for further instructions. In order to determine the optimal path, one typically formulates an associated optimization problem and solves it to optimality. In the next subsection, we introduce all the needed formalism.

5.3.2 Optimization Problem Formulation and Description

We consider a scenario in which N robots, indexed by $\mathbb{I} \triangleq \{1, \ldots, N\}$, have to serve the transportation requests. We denote by $P \triangleq \{1, \ldots, |P|\}$ the index set of pickup requests and by $D \triangleq \{|P+1|, \ldots, 2|P|\}$ the index set of delivery demands (with $P \cap D = \emptyset$). To each pickup location $j \in P$ is associated a delivery location $j \in D$ (with $|P| = |D|$), so that both the requests must be served by the same robot. To ease the notation, we also define a set $R \triangleq P \cup D$ of *all* the transportation requests (independently of their pickup/delivery nature). Each request $j \in R$ is characterized by a service time $d_j \geq 0$, which is the time needed to perform the pickup or delivery operation. Within each request, it is also associated a load $q_j \in \mathbb{R}$, which is positive if $j \in P$ and negative if $j \in D$. Each robot has a maximum load capacity $C_i \geq 0$ of goods that can be simultaneously held. The travel time needed for the i-th vehicle to move from a location $j \in R$ to another location $k \in R$ is denoted by $t_i^{jk} \geq 0$. In order to travel from two locations j, k, the i-th robot incurs a cost $c_i^{jk} \in \mathbb{R}_{\geq 0}$. Finally, two additional locations s and σ are considered. The first one represents the mission starting point, while the second one is a virtual ending point. For this reason, the corresponding demands q^s, q^σ and service times d^s, d^σ are set to 0.

The goal is to construct minimum cost paths satisfying all the transportation requests. To this end, a graph of all possible paths through the transportation requests is defined as follows. Let $\mathcal{G}_A = (V_A, \mathcal{E}_A)$, be the graph with vertex set $V_A = \{s, \sigma\} \cup R$ and edge set $\mathcal{E}_A = \{(j,k) \mid j,k \in V_A, j \neq k \text{ and } j \neq \sigma, k \neq s\}$. Owing to its definition, $\mathcal{E}_A$ contains edges starting from s or from locations in R and ending in σ or other locations in R. For all edges $(j,k) \in \mathcal{E}_A$, let x_i^{jk} be a binary variable denoting whether vehicle $i \in \{1, \ldots, N\}$ is traveling ($x_i^{jk} = 1$) or not ($x_i^{jk} = 0$) from a location j to a location k. Also, let $B_i^j \in \mathbb{R}_{\geq 0}$ be an the optimization variable modeling the time at which vehicle i begins its service at location j. Similarly, let $Q_i^j \in \mathbb{R}_{\geq 0}$ be the load of vehicle i when leaving location j. To keep the notation light, we denote by x the vector stacking x_i^{jk} for all i, j, k and by B, Q the vectors stacking all B_i^j and Q_i^j. The PDVRP can be formulated as the following optimization problem [92],

$$\min_{x,B,Q} \sum_{i=1}^{N} \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} \tag{5.4a}$$

$$\text{subj. to} \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1 \qquad \forall j \in R \tag{5.4b}$$

$$\sum_{k:(s,k)\in\mathcal{E}_A} x_i^{sk} = 1 \qquad \forall i \in \mathbb{I} \tag{5.4c}$$

Table 5.1: List of the main symbols and their definitions

Symbol	Description
$N \in \mathbb{N}_{\geq 0}$	Number of vehicles of the system
$\mathbb{I}$	Set of vehicles
R	Set of all transportation requests
V_A	Set of PDVRP graph vertices
$\mathcal{E}_A$	Set of PDVRP graph edges
$x_i^{jk} \in \{0,1\}$	1 if vehicle i travels arc (j,k), 0 otherwise
$Q_i^j \in \mathbb{R}_{\geq 0}$	Load of vehicle i when leaving vertex j
$B_i^j \in \mathbb{R}_{\geq 0}$	Beginning of service of vehicle i at vertex j
$c_i^{jk} \in \mathbb{R}_{\geq 0}$	Incurred cost if vehicle i travels arc (j,k)
$q^j \in \mathbb{R}$	Demand/supply at location $j \in R$
$C_i \in \mathbb{R}_{\geq 0}$	Capacity of vehicle i
$t_i^{jk} \in \mathbb{R}_{\geq 0}$	Travel time from j to k for vehicle i
$d^j \in \mathbb{R}_{\geq 0}$	Service duration at $j \in V_A$

$$\sum_{j:(j,\sigma)\in\mathcal{E}_A} x_i^{j\sigma} = 1 \qquad \forall i \in \mathbb{I} \tag{5.4d}$$

$$\sum_{j:(j,k)\in\mathcal{E}_A} x_i^{jk} = \sum_{j:(k,j)\in\mathcal{E}_A} x_i^{kj} \qquad \forall i \in \mathbb{I}, k \in R \tag{5.4e}$$

$$\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} = \sum_{k:(j+|P|,k)\in\mathcal{E}_A} x_i^{|P|+j,k} \quad \forall i \in \mathbb{I}, j \in P \tag{5.4f}$$

$$B_i^j \leq B_i^{j+|P|}, \qquad \forall i \in \mathbb{I}, j \in P \tag{5.4g}$$

$$x_i^{jk} = 1 \Rightarrow B_i^k \geq B_i^j + d^j + t_i^{jk} \tag{5.4h}$$

$$x_i^{jk} = 1 \Rightarrow Q_i^k = Q_i^j + q^k \tag{5.4i}$$

$$\underline{Q}^j \leq Q_i^j \leq \overline{Q}_i^j \qquad \forall j \in V_A, i \in \mathbb{I} \tag{5.4j}$$

$$Q_i^s = Q_i^{\text{init}} \qquad \forall i \in \mathbb{I} \tag{5.4k}$$

$$x_i^{jk} \in \{0,1\} \qquad \forall i \in \mathbb{I}, (j,k) \in \mathcal{E}_A, \tag{5.4l}$$

where $\underline{Q}^j = \max\{0, q^j\}$, $\overline{Q}_i^j = \min\{C_i, C_i + q^j\}$ and $Q_i^{\text{init}} \in \mathbb{R}_{\geq 0}$. We make the standing assumption that Problem (5.4) is feasible and admits an optimal solution. Throughout the section, we use the convention that subscripts denote the vehicle index, while superscripts refer to locations. Table 5.1 collects all the relevant symbols.

Notice that, in order to satisfy (5.4j), it is necessary to assume $C_i \geq \max_{j\in R}\{q^j\}$, i.e., the generic vehicle must have sufficient capacity to accomplish any of the pickup/delivery tasks. In order to keep the discussion not too technical, in the following we always maintain this assumption. However, note that the proposed algorithm works also if this assumption is removed. A discussion on this extension is given in Section 5.3.5.

Before going on with the presentation, we note that Problem (5.4) is mixed-integer but not linear. Indeed, the constraints (5.4h)–(5.4i) are nonlinear. However, an equivalent linear formulation of these constraints is always possible (the detailed procedure is outlined in Section 5.3.8). In the following, we refer to Problem (5.4) as being a MILP, with the implicit assumption that the constraints (5.4h)–(5.4i) are replaced by their linear version.

Let us detail the cost and constraints of Problem (5.4). The objective (5.4a) minimizes the total route cost, in particular, the total euclidean distance traveled by robots. Constraint (5.4b) enforces that every location has to be visited at least once. Typically, PDVRP formulations consider this constraint as an equality, however, in the considered case of cost being the total euclidean distance, the solution is the same both with the equality and with the inequality. We stick to the inequality formulation as this allows us to exploit the problem structure and apply the methods of Chapter 3. Constraints (5.4c)–(5.4d) guarantee that every vehicle starts at s and ends its mission at σ. Equality (5.4e) is a flow conservation constraint, meaning that if a vehicle enters a location k it also has to leave it. Constraint (5.4f) ensures that, if a robot i performs a pickup operation, it also has to perform the corresponding delivery. Inequality (5.4g) imposes that deliveries have to occur after pickups. Constraint (5.4h) avoids subtours in each vehicle route (i.e. paths passing from the same location more than once), while inequalities (5.4i)–(5.4j) ensure that the total vehicle capacity is never exceeded. Finally, (5.4k) takes into account the initial load of the i-th robot.

5.3.3 Distributed Primal Decomposition for Pickup and Delivery

Let us now apply the methods developed in Chapter 3. We first show how to recast Problem (5.4) as a constraint-coupled MILP and then describe the distributed algorithm.

Reformulation as Constraint-coupled MILP

We assume robots aim to solve Problem (5.4) in a distributed fashion, i.e. without a central (coordinating) node. In order to solve the problem, we suppose each robot is equipped with its own communication and computation capabilities. Robots can exchange information according to a static communication network modeled as a connected and undirected graph $\mathcal{G} = (\{1, \ldots, N\}, \mathcal{E})$. The graph $\mathcal{G}$ models the communication in the sense that there is an edge $(i, j) \in \mathcal{E}$ if and only if agent i is able to send information to agent j. For each node i, the set of neighbors of i at time t is denoted by $\mathcal{N}_i$ and is the set of j such that there exists an edge $(i, j) \in \mathcal{E}$. We assume that the i-th robot only knows the travelling times t_i^{jk}, the local capacity C_i and the cost entries c_i^{jk}. Reasonably, we assume assume that all the robots know the demand/supply values q_j and service time d_j for each task request $j \in V_A$.

Note that the optimization variables in (5.4) associated with a robot i are all and only the variables with subscript i (i.e. x_i^{jk}, B_i^j and Q_i^j for all j,k). Therefore, it is possible to recast Problem (5.4) as a constraint-coupled MILP of the form (3.1), i.e.

$$\begin{aligned}
\min_{z_1,\ldots,z_N} \quad & \sum_{i=1}^{N} c_i^\top z_i \\
\text{subj. to} \quad & \sum_{i=1}^{N} A_i z_i \leq b \\
& z_i \in Z_i, \qquad i = 1,\ldots,N.
\end{aligned} \tag{5.5}$$

In particular, the vector $z_i \in \mathbb{R}^{n_i}$ is the stack of all the variables x_i^{jk}, B_i^j and Q_i^j for all j,k ($n_i \in \mathbb{N}$ denotes the total number of entries). Moreover, note that the constraints (5.4c)–(5.4l) are repeated for each index i, thus the local mixed-integer sets are

$$Z_i = \Big\{(x_i, B_i, Q_i) \text{ such that (5.4c)–(5.4l) are satisfied}\Big\}, \tag{5.6}$$

and finally note that for suitable A_i and b it is possible to recast the constraint (5.4b) as $\sum_{i=1}^{N} A_i z_i \leq b$. With this shorthands, any route that robot i can implement can be denoted more shortly as $z_i = (x_i, B_i, Q_i) \in Z_i$ and any solution satisfying all the pickup/delivery demands satisfies the constraint $\sum_{i=1}^{N} A_i z_i \leq b$. Within the primal decomposition approach described in Chapter 3, each robot i will aim to compute an allocation $y_i \in \mathbb{R}^{|R|}$ satisfying

$$\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i]_j, \qquad \forall j \in R,$$

where $[y_i]_j$ denotes the j-th component of y_i. As it will be clear from the forthcoming analysis, the j-th entry of the vector y_i immediately determines whether or not robot i must perform task j. In the next subsection, we introduce our distributed algorithm, whose purpose is to coordinate the computation of allocation vectors y_i such that (5.4b) is satisfied.

Distributed Algorithm Description

Let us now show how Algorithm 3 reads for Problem (5.4). Let $t \in \mathbb{N}$ be the iteration index. Each agent i maintains an estimate of the local allocation vector $y_i^t \in \mathbb{R}^{|R|}$. At each iteration, the vector y_i^t is updated according to (5.7)–(5.8). After a finite number of iterations, say $T_f \in \mathbb{N}$, the agents compute a tentative solution to the PDVRP based on the last computed allocation $y_i^{T_f}$ with (5.9)–(5.10). Algorithm 9 summarizes the algorithm as performed by robot i. The symbol $\text{conv}(Z_i)$ denotes the convex hull of the set Z_i, while α^t is a step-size sequence. The algorithm has also some tuning parameters that are reported on the top of the table.

Algorithm 9 Distributed Primal Decomposition for PDVRP

```
Global parameters: $T_f > 0$, $M > 0$, $0 < \delta < 1$.
Initialization: Set $y_i^0 = \delta/N\,\mathbf{1}$.
Repeat for $t = 0, 1, \ldots, T_f - 1$:
    Compute $\mu_i^t$ as Lagrange multiplier of
```

$$\begin{aligned}\min_{x_i, B_i, Q_i, v_i} \quad & \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} + M v_i \\ \text{subj. to} \quad & \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^t]_j - v_i, \qquad \forall j \in R \\ & (x_i, B_i, Q_i) \in \text{conv}(Z_i), \;\; v_i \geq 0\end{aligned} \tag{5.7}$$

```
    Receive $\mu_j^t$ from neighbors $j \in \mathcal{N}_i$ and update
```

$$y_i^{t+1} = y_i^t - \alpha^t \sum_{j\in\mathcal{N}_i} \left(\mu_i^t - \mu_j^t\right) \tag{5.8}$$

```
Perform component-wise thresholding of allocation
```

$$y_i^{\text{END}} = \min\left(y_i^{T_f}, \mathbf{1}\right) \tag{5.9}$$

```
Return $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$ as optimal solution of MILP
```

$$\begin{aligned}\min_{x_i, B_i, Q_i} \quad & \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} \\ \text{subj. to} \quad & \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j, \qquad \forall j \in R \\ & (x_i, B_i, Q_i) \in Z_i\end{aligned} \tag{5.10}$$

The algorithm structure is inherited from Algorithm 3 in Section 3.3. However, the final mixed-integer recovery step has some changes with respect to Algorithm 3. Specifically, the lex-min optimization (3.7) (which we recall from Section 3.3.2 to be equivalent to solving two MILPs) is replaced with a component-wise thresholding of the last computed allocation $y_i^{T_f}$ and the solution of a final MILP (5.10). In Section 5.3.4 we show that such a modification preserves the results of Chapter 3, while a discussion on the choice of the tunable parameters M, T_f and δ can be found in Section 5.3.5.

5.3.4 Algorithm Analysis

In this section, we provide a theoretical study of Algorithm 9. In particular, we will show that the algorithm provides a feasible solution to Problem (5.4) in finite time. There are two important differences with respect to the analysis of the general algorithm in Chapter 3:

(i) Algorithm 9 does not employ a restriction-based mechanism. On the contrary, in Algorithm 9 we imposed the initialization $y_i^0 = \delta/N\mathbf{1}$, with $\delta \in (0,1)$, which means that we actually *enlarged* the right-hand side $\mathbf{1}$ of constraint (5.4b) to $\sum_{i=1}^N y_i^0 = \delta\mathbf{1} < \mathbf{1}$.

(ii) The final lex-min optimization is replaced with the thresholding step and MILP (5.10).

Feasibility of Local Problems

We begin the analysis by proving that the algorithm is well posed. In particular, we show that it is indeed possible to solve Problems (5.7) and (5.10). The next lemma formalizes feasibility of Problem (5.7).

Lemma 5.1. *Consider a robot $i \in \mathbb{I}$ and let y_i^t be allocation computed by Algorithm 9 at an iteration t. Then, Problem* (5.7) *is feasible.*

Proof. Note that Problem (5.7) is the epigraph form of

$$\begin{aligned} \min_{x_i, B_i, Q_i} \; & \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} + M \max\left\{0, \max_{j\in R}\left([y_i^t]_j - \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk}\right)\right\} \\ \text{subj. to} \; & (x_i, B_i, Q_i) \in \text{conv}(Z_i) \end{aligned}$$

Moreover, it holds $\text{conv}(Z_i) \supset Z_i$. Therefore, the proof follows since Z_i is not empty (by assumption). ■

Proving feasibility of Problem (5.10) is more delicate and relies upon the thresholding operation, as formally shown next.

Lemma 5.2. *Consider a robot $i \in \mathbb{I}$ and let y_i^{END} be the final allocation computed by Algorithm 9. Then, Problem* (5.10) *is feasible.*

Proof. Fix a robot i. Because of the thresholding operation (5.9), it holds $[y_i^{\text{END}}]_j \leq 1$ for all $j \in R$. We now show that the feasible set of Problem (5.10) is not empty. Since Problem (5.4) is assumed to be feasible, we know that there exists $z_i = (x_i, B_i, Q_i) \in Z_i$. Thus, we only have to show that the constraint $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j$ for all $j \in R$ can be satisfied by at least one vector $z_i \in Z_i$.

Due to the flow constraints (5.4e), the subtour elimination constraints (5.4h) and the integer constraints (5.4g), for all $j \in R$ the quantity $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk}$ is either equal to 0 or equal to 1, since there can be at most one index k satisfying $(j,k) \in \mathcal{E}_A$ and such that $x_i^{jk} = 1$. Consider the constraint $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j$ and fix a component

$j \in R$. Note that this constraint essentially imposes whether or not robot i must pass through location j. Indeed, on the one hand, if $[y_i^{\text{END}}]_j \leq 0$, the vector z_i can be chosen such that the left-hand side is either equal to 0 (vehicle i does not pass through location j) or equal to 1 (vehicle i passes through location j). In either case, it holds $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 0 \geq [y_i^{\text{END}}]_j$ so that the constraint is satisfied. On the other hand, if $0 < [y_i^{\text{END}}]_j \leq 1$ (recall that $[y_i^{\text{END}}] \leq 1$ for all $j \in R$ and thus there are no other possibilities), then the only way to satisfy the constraint is to have $x_i^{jk} = 1$ for some index k with $(j,k) \in \mathcal{E}_A$, in which this case we would obtain $1 = \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j > 0$. As a consequence, Problem (5.10) admits as feasible solution any vector $(x_i, B_i, Q_i) \in Z_i$ representing a path passing through all the locations $j \in R$ and satisfying $[y_i^{\text{END}}]_j > 0$. ■

Convergence of Distributed Allocation Scheme

We now focus on the first logic block of Algorithm 9, namely steps (5.7)–(5.8). As shown in Chapter 2, these two iterative steps can be used to obtain an optimal allocation associated to the convex relaxation

$$\begin{aligned}
\min_{x,B,Q} \quad & \sum_{i=1}^{N} \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} \\
\text{subj. to} \quad & \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq \delta \qquad && \forall j \in R \\
& (x_i, B_i, Q_i) \in \text{conv}(Z_i), && \forall\, i \in \mathbb{I}.
\end{aligned} \tag{5.11}$$

Denote by $(z_1^{\text{CONV}}, \ldots, z_N^{\text{CONV}})$ an optimal solution of Problem (5.11) with each $z_i = (x_i, B_i, Q_i)$, and by $\{y_1^t, \ldots, y_N^t\}_{t\geq 0}$ the allocation vector sequence produced by (5.7)–(5.8). We now recall from Chapter 3 the convergence result of steps (5.7)–(5.8).

Assumption 5.1. *The step-size sequence $\{\alpha^t\}_{t\geq 0}$, with each $\alpha^t \geq 0$, satisfies $\sum_{t=0}^{\infty} \alpha^t = \infty$, $\sum_{t=0}^{\infty} \big(\alpha^t\big)^2 < \infty$.* △

Proposition 5.1. *Let Assumption 5.1 hold and recall that $y_i^0 = \delta/N\mathbf{1}$ for all $i \in \mathbb{I}$. Moreover, let $(y_1^{\text{CONV}}, \ldots, y_N^{\text{CONV}}) \in \mathbb{R}^{N|R|}$ be an optimal allocation associated to Problem (5.11), i.e. a vector satisfying $\sum_{i=1}^{N} y_i^{\text{CONV}} = \delta\mathbf{1}$ and $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{CONV}}]_j$ for all $j \in R$ and $i \in \mathbb{I}$. Then, for a sufficiently large $M > 0$, the distributed algorithm (5.7)–(5.8) generates a sequence $\{y_1^t, \ldots, y_N^t\}_{t\geq 0}$ such that*

(i) $\sum_{i=1}^{N} y_i^t = \delta\mathbf{1}$, *for all $t \geq 0$;*

(ii) $\lim_{t\to\infty} \|y_i^t - y_i^{\text{CONV}}\| = 0$ *for all $i \in \mathbb{I}$.* △

Intermediate Results

Before formalizing our main result, we provide some preparatory lemmas, in which essentially prove that the restriction vector (cf. Section 3.3.3) needed by Problem (5.4) is zero. We denote by **0** the vector of zeros of appropriate dimension. The first lemma justifies the use of δ (an arbitrarily small positive number) in place of the original right-hand side 1 in the coupling constraints (5.4b).

Lemma 5.3. *For all $i \in \mathbb{I}$, let $(x_i, B_i, Q_i) \in Z_i$ such that $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} > 0$ for all $j \in R$. Then, it holds $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1$ for all $j \in R$.*

Proof. For each component $j \in R$, by assumption we have

$$\sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} > 0. \tag{5.12}$$

Note that, since each $x_i^{jk} \in \{0,1\}$, the quantity $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk}$ is either zero or at least equal to 1. Therefore, because of the assumption, we have $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1$ for all $j \in R$. ■

The next two lemmas will used in the sequel to characterize an optimal allocation associated to Problem (5.11).

Lemma 5.4. *For all $i \in \mathbb{I}$, let $\tilde{y}_i \in \mathbb{R}^{|R|}$ and $(\tilde{x}_i, \tilde{B}_i, \tilde{Q}_i) \in \text{conv}(Z_i)$ such that $\sum_{k:(j,k)\in\mathcal{E}_A} \tilde{x}_i^{jk} > [\tilde{y}_i]_j$ for all $j \in R$. Then, there exists $(\bar{x}_i, \bar{B}_i, \bar{Q}_i) \in Z_i$ satisfying $\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} > [\tilde{y}_i]_j$ for all $j \in R$.*

Proof. Fix a robot $i \in \mathbb{I}$ and note that, because of the flow constraints (5.4e) and the subtour elimination constraints (5.4h), for all $j \in R$ it holds

$$\max_{(x_i,B_i,Q_i)\in Z_i} \left[\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \right] = 1. \tag{5.13}$$

Moreover, note that it is possible to choose a local solution passing through all the locations (possibly at a high cost), i.e. there exists $(\bar{x}_i, \bar{B}_i, \bar{Q}_i) \in Z_i$ such that $\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} = 1$ for all $j \in R$. Therefore, for all $j \in R$ it holds

$$\begin{aligned}
\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} &= \max_{(x_i,B_i,Q_i)\in Z_i} \left[\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \right] \\
&\overset{(a)}{=} \max_{(x_i,B_i,Q_i)\in \text{conv}(Z_i)} \left[\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \right] \\
&\geq \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \qquad \text{for all } (x_i,B_i,Q_i)\in\text{conv}(Z_i),
\end{aligned} \tag{5.14}$$

where *(a)* follows by linearity of the cost. In particular, the previous inequality holds with $(x_i, B_i, Q_i) = (\tilde{x}_i, \tilde{B}_i, \tilde{Q}_i)$ and thus for all $j \in R$ we have $\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} \geq \sum_{k:(j,k)\in\mathcal{E}_A} \tilde{x}_i^{jk} > [\tilde{y}_i]_j$. ∎

Lemma 5.5. *Let* $(y_1^{\text{CONV}}, \ldots, y_N^{\text{CONV}}) \in \mathbb{R}^{N|R|}$ *be an optimal allocation associated to Problem* (5.11), *i.e. a vector satisfying* $\sum_{i=1}^N y_i^{\text{CONV}} = \delta\mathbf{1}$ *and* $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{CONV}}]_j$ *for all* $j \in R$ *and* $i \in \mathbb{I}$. *Then,* $y_i^{\text{CONV}} \leq \mathbf{1}$ *for all* $i \in \mathbb{I}$.

Proof. By contradiction, suppose that there is a component $j \in R$ for which $[y_i^{\text{CONV}}]_j > 1$. By assumption, we have $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{CONV}}]_j$. Using Lemma 5.4, we conclude that there exists $(\bar{x}, \bar{B}, \bar{Q}) \in Z_i$ such that

$$\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} \geq [y_i^{\text{CONV}}]_j > 1,$$

which contradicts (5.13). ∎

Main Result

The next theorem summarizes the finite-time feasibility result.

Theorem 5.1. *Let Assumption 5.1 hold and let* $0 < \delta < 1$. *Then, for a sufficiently large* $M > 0$, *there exists a time* $T_\delta > 0$ *such that the vector* $(z_1^{\text{END}}, \ldots, z_N^{\text{END}})$, *the aggregate output of Algorithm 9, with each* $z_i^{\text{END}} = (x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$, *is a feasible solution to Problem* (5.4), *provided that the total iteration count satisfies* $T_f \geq T_\delta$.

Proof. First, note that, by Lemmas 5.1 and 5.2, the algorithm is well posed. Moreover, by construction it holds $z_i^{\text{END}} = (x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}}) \in Z_i$ for all $i \in \mathbb{I}$ (indeed z_i^{END} is selected as an optimal solution of Problem (5.10)). Therefore, all the local constraints (5.4c) to (5.4l) are satisfied by $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$ and we only need to show that there exists $T_\delta > 0$ such that constraint (5.4b) is satisfied by $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$ if $T_f \geq T_\delta$. By Lemma 5.3, it suffices to prove that $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} > 0$ for all $j \in R$.

Consider the auxiliary sequence $\{y_1^t, \ldots, y_N^t\}_{t\geq 0}$ generated by Algorithm 9. By Proposition 5.1, this sequence converges to the vector $(y_1^{\text{CONV}}, \ldots, y_N^{\text{CONV}})$. By definition of limit (using the infinity norm), there exists $T_\delta > 0$ such that $\|y_i^t - y_i^{\text{CONV}}\|_\infty < \delta/N$ (and thus $y_i^t < y_i^{\text{CONV}} + \delta/N\mathbf{1}$) for all $i \in \mathbb{I}$ and $t \geq T_\delta$.

Let us define a vector $\rho_i \in \mathbb{R}^{|R|}$ representing the mismatch between $y_i^{T_f}$ and its thresholded version y_i^{END},

$$\rho_i = y_i^{T_f} - y_i^{\text{END}}, \qquad \text{for all } i \in \mathbb{I}. \tag{5.15}$$

By definition (5.9), it holds $\rho_i \geq 0$. Then, for all $j \in R$, it holds

$$\sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{\text{END}jk} \geq \sum_{i=1}^{N} [y_i^{\text{END}}]_j = \underbrace{\sum_{i=1}^{N} [y_i^{T_f}]_j}_{\delta} - \sum_{i=1}^{N} [\rho_i]_j = \delta - \sum_{i=1}^{N} [\rho_i]_j$$

Let us temporarily assume that $\rho_i < \delta/N\mathbf{1}$ for all $i \in \mathbb{I}$. Then, we obtain the desired statement

$$\sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{\text{END}jk} \geq \delta - \sum_{i=1}^{N} [\rho_i]_j > \left(\delta - \sum_{i=1}^{N} \delta/N\right) = 0.$$

It remains to show that $\rho_i < \delta/N\mathbf{1}$ for all $i \in \mathbb{I}$. Fix an agent i and consider a component $j \in R$ of the vector ρ_i. Owing to the definition (5.9) of y_i^{END}, either the j-th component is equal to $[y_i^{T_f}]_j$ or it is equal to 1. In the former case, we have $[\rho_i]_j = 0 < \delta/N$. In the latter case, there is a non-negative mismatch $[\rho_i]_j = [y_i^{T_f}]_j - 1 \geq 0$. Now, using the fact $y_i^t < y_i^{\text{CONV}} + \delta/N\mathbf{1}$ for all $t \geq T_\delta$, we have

$$[\rho_i]_j = [y_i^{T_f}]_j - 1 < [y_i^{\text{CONV}}]_j - 1 + \delta/N \leq \delta/N \tag{5.16}$$

provided that $T_f \geq T_\delta$, where in the last inequality we applied Lemma 5.5. The proof follows. ■

5.3.5 Discussion and Extensions

In this section, we provide guidelines for the choice of the algorithm parameters and we discuss possible extensions of Algorithm 9 to more general settings.

On the Choice of the Parameters

As already mentioned in Section 5.3.3, there are a few parameters that must be appropriately set in order for Algorithm 9 to work correctly. The basic requirements for the parameters are summarized in Theorem 5.1 and are recalled here: *(i)* $M > 0$ must be sufficiently large, *(ii)* δ is any number in the open interval $(0, 1)$, *(iii)* the total number of iterations $T_f > 0$ must be sufficiently large, *(iv)* the step-size sequence $\{\alpha^t\}_{t\geq 0}$ must satisfy Assumption 5.1.

We only discuss the parameters δ and T_f as a discussion on the parameter M can be found in 2.3.4 and possible values of the step sizes are discussed in Section 2.3. The purpose of the parameter δ is to enlarge the constraint (5.4b) and is linked to the minimum value of T_f. There is an inherent tradeoff between δ and the minimal T_f. For δ close to 1, precedence is given to feasibility and T_f may become smaller, while for δ close to 0, solution optimality is prioritized, possibly at the cost of a higher T_f.

Indeed, for δ close to 1, the time T_δ in the proof of Theorem 5.1 may be smaller, and therefore a smaller number of iterations T_f may be sufficient to attain feasibility of the solution. Instead, for δ close to 0, a greater number of iterations may be required to obtain feasibility. However, in the latter case there could be less agents for which the components of $y_i^{T_f}$ are positive (because, by Proposition 5.1, it must hold $\sum_{i=1}^N y_i^{T_f} = \delta\mathbf{1}$ with a small positive δ), thus forcing less agents to pass through the same location.

Extension to Heterogeneous PDVRP Graphs

Let us outline a possible extension of Problem (5.4) that can be handled by Algorithm 9. Recall from Section 5.3.2 that Problem (5.4) has the implicit assumption that $C_i \geq \max_{j \in R}\{q^j\}$ for all $i \in \mathbb{I}$, where C_i is the capacity of vehicle i and q^j is the demand/supply at location j. In real scenarios, while there may be vehicles potentially capable of performing all the pickup/delivery requests, it is often the case that many vehicles are small sized and can only accomplish a subset of the task requests. This means that the assumption $C_i \geq \max_{j \in R}\{q^j\}$ may not hold for some robots.

The general case just outlined can be handled with minor modifications in the formulation of Problem (5.4) and in the algorithm. Indeed, if $C_i < q^j$ for some robot i and some location j, Problem (5.4) would be infeasible by construction. Thus, for each robot $i \in \mathbb{I}$ it is necessary to define the largest *local* set of requests $R_i \subseteq R$ such that $C_i \geq \max_{j \in R_i}\{q^j\}$. As a consequence, owing to the description in Section 5.3.2, the sets R_i will now induce *local* graphs $\mathcal{G}_{Ai} = (V_{Ai}, \mathcal{E}_{Ai})$ with vertex set $V_{Ai} = \{s, \sigma\} \cup R_i$ and edge set $\mathcal{E}_{Ai} = \{(j,k) \mid j,k \in V_{Ai}, j \neq k \text{ and } j \neq \sigma, k \neq s\}$. Thus, each robot i defines only the optimization variables x_i^{jk} with $j,k \in V_{Ai}$ (instead of $j,k \in V_A$) and similarly for B_i and Q_i. The optimization problem is formulated similarly to Problem (5.4) with obvious modifications (by dropping all the references to non-existing optimization variables). The resulting modified version of Problem (5.4) is now feasible as long as each task can be performed by at least one robot.

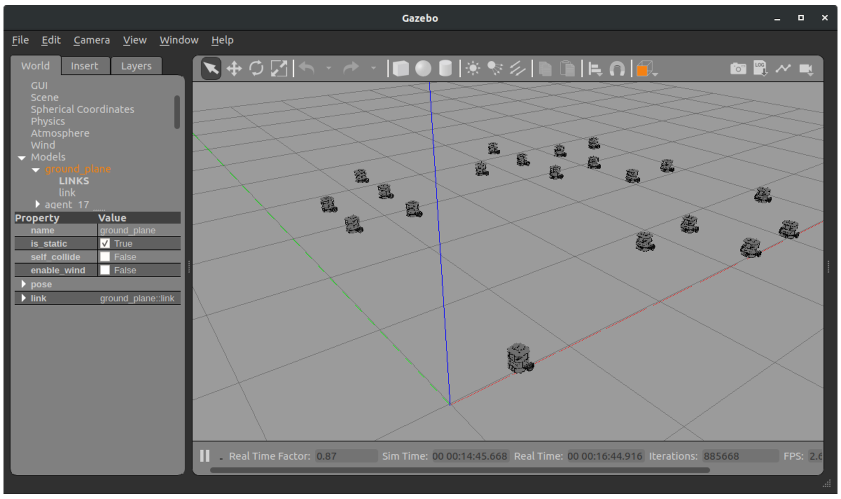
The distributed algorithm also requires minor modifications. In particular, the summations in problems (5.7) and (5.10) are performed using $\mathcal{E}_{Ai}$ in place of $\mathcal{E}_A$. Moreover, the thresholding operation (5.9) is replaced by the following one

$$[y_i^{\text{END}}]_j = \begin{cases} \min\left([y_i^{T_f}]_j, 1\right) & \text{if } j \in R_i \\ \min\left([y_i^{T_f}]_j, 0\right) & \text{otherwise} \end{cases}$$

for all $j \in R$. Finally, the sets Z_i must be replaced by the new version of constraints (5.4c)–(5.4l). It is possible to follow essentially the same line of proof outlined in Section 5.3.4 (precautions are only necessary in Lemmas 5.2 and 5.4), so that Theorem 5.1 holds with no changes.

5.3.6 Simulations on Gazebo

In this section, we provide simulation results on teams of ground robots that have to serve a set of pickup and delivery requests scattered in the environment. Thanks to the realistic simulations provided by Gazebo, the proposed results are comparable to experimental results on a real team of robots. In Figure 5.8, we show a snapshot of one of the simulations addressed in the following.



Property	Value
name	ground_plane
is_static	✓ True
self_collide	False
enable_wind	False
pose	
link	ground_plane::link

Real Time Factor: 0.87 Sim Time: 00 00:14:45.668 Real Time: 00 00:16:44.916 Iterations: 885668 FPS: 2.6

Figure 5.8: Snapshot of the initial condition of one of the Gazebo simulations.

Implementation Details

To implement the distributed algorithm for PDVRP, we developed a modified version of the ChoiRbot's Team Guidance layer of Dynamic Task Assignment 5.2.4 in which we encoded the problem formulation (5.4) and the distributed algorithm by using the disropt layer. The RoboPlanning layer and the RoboControl layer are left unchanged. Note that ROS 2 handles inter-process communications via the TCP/IP stack, meaning that the proposed simulations are based on the same network communications needed to run the pickup and delivery algorithm on a team of real robots connected over a wireless network. At the beginning of the simulation, each optimization node gathers the information on the pickup and delivery requests and evaluates the cost vector c_i (i.e. the robot-to-task distances) and the local constraint sets Z_i. We stress that these computations are performed independently for each robot on different processes, without having access to the other robot information. After the initialization, robots start communicating and performing the steps of the distributed algorithm proposed in Section 5.3.3. As soon as the distributed optimization procedure completes, robots start moving towards the assigned tasks. Due to constraint (5.4b), suboptimal solutions of problem (5.4) may lead more than one robot to perform the same task. Thus, the robot communicates to a so-called Auth node that it wants to start a particular task. If the task has been already taken care of by another robot, the Auth node denies authorization and the robot performs the next one. The target positions are then communicated to the local trajectory planning layer and then fed to the low-level controllers to steer the robots over the requests positions. The controller nodes interact with Gazebo, which simulates the robot dynamics and provides the pose of each robot.

We have experimentally found that a satisfying tuning of the distributed algorithm is as follows. The robots perform 250 iterations with local allocation initialized as in Algorithm 9. For the first 125 iterations they use the diminishing step size $\alpha^t = 0.005/(t+1)$, then they use a constant step size (equal to the last computed one). Because of the constant step size, the final allocation fed to the thresholding operation (5.9) is the running average computed from iteration 126 on, i.e.

$$\Big(\sum_{\tau=126}^{250} \alpha^\tau y_i^\tau\Big) \Big/ \Big(\sum_{\tau=126}^{250} \alpha^\tau\Big)$$

This particular tweaking allows the robots to quickly converge to a good-quality solution.

Results

We performed three Monte Carlo simulations on random instances of problem (5.4) on the described platform with Turtlebot3 robots.

First simulation. To begin with, we test the optimality of the computed solution while varying the number of robots N. We perform 50 Monte Carlo trials for each value of N and we fix $\delta = 0.9$ to prioritize feasibility over optimality (cf. Section 5.3.5). For each trial, 10 pickup requests and 10 corresponding deliveries are randomly generated on the plane. In Figure 5.9, we show the cost error of the solution actuated by robots after 250 iterations of the distributed algorithm, compared to the cost of a centralized solver, with varying number of robots. The figure highlights that the distributed algorithm is able to achieve an average 20% suboptimality.

Second simulation. Now, we assess the behavior of the cost error while varying the total number of requests. Specifically, we consider a team of 20 robots and we let the number of requests $|R|$ vary from 4 to 24 (with $\delta = 0.9$). For each of these values of $|R|$, we perform 50 trials and we let the robots implement the solution after 250 iterations of the distributed algorithm. The results are depicted in Figure 5.10 and Figure 5.11. Notably, as it can be seen from Figure 5.10, the mean relative error remains constant while increasing the number of requests. This is an appealing feature of the proposed strategy considering the fact that, as depicted also in Figure 5.11, the global optimal cost increases with the number of tasks.

Third simulation. Finally, we perform simulations to determine the number of

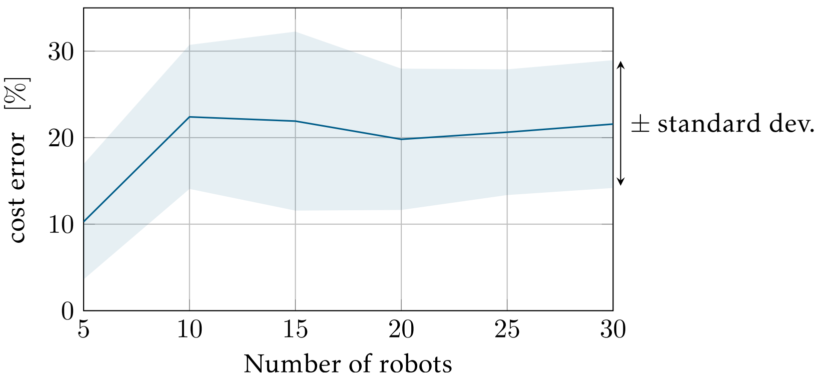


Figure 5.9: Cost error in Monte Carlo simulations on Gazebo for varying number of robots.

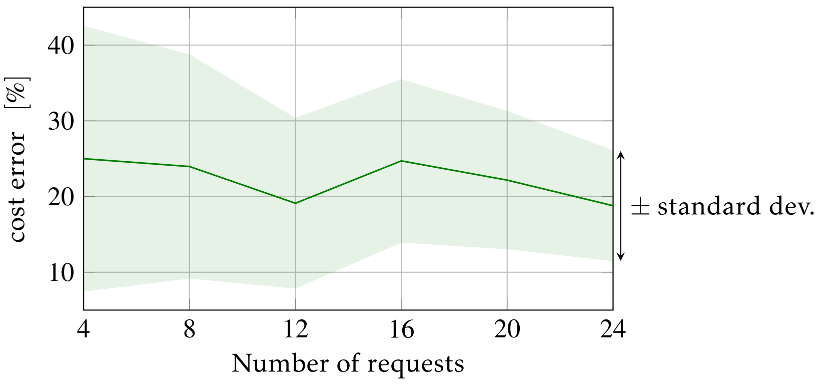


Figure 5.10: Cost error in Monte Carlo simulations on Gazebo for varying number of requests.

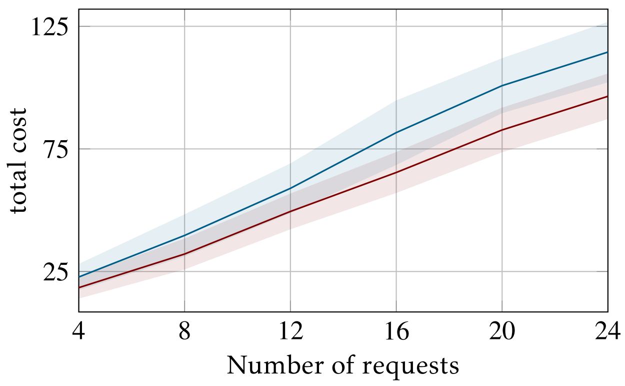


Figure 5.11: Comparison between the centralized optimal solution (red) and the one found by the proposed distributed strategy (blue).

iterations needed to achieve finite-time feasibility while varying the number of robots N and the value of δ. We employed three values of δ, namely $0.1, 0.5, 0.9$, and performed 50 trials for each of the values $N = 5, 10, 15, 20, 25, 30$ and for each value of δ. In each trial, we performed 250 iterations of the algorithm and recorded the value of the coupling constraint. Then we determined the following quantity

$$\begin{aligned} \min \;\; & t \\ \text{such that} \;\; & \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk^\tau} \geq 1 \qquad \text{for all } \tau \geq t, \end{aligned}$$

which is essentially an empirical value of T_δ appearing in Theorem 5.1. Interestingly, we found out that, for all the trials, the empirical value of T_δ is zero, which means that in all the simulated scenarios the algorithm provides a feasible solution to the PDVRP (5.4) since the first iteration.

5.3.7 Experiments

To conclude, we show experimental results on real teams of robots solving PDVRP instances. We consider heterogeneous teams composed by Crazyflie nano-quadrotors and Turtlebot 3 Burger mobile robots. Tasks are generated randomly in the space. In particular, we split the experiment area in two halves. Pickup requests are located in the right half, while deliveries are in the left half. To simulate the pickup/delivery procedure, each robot waits over the request location a random service time d^j between 3 and 5 seconds. Each robot can serve a subset of the pickup/delivery requests. This is decided randomly at the beginning of the experiment. The capacity C_i of each robot and the demand/supply q^j of tasks are drawn from uniform distributions. The velocity of robots (both ground and aerial) is approximately $0.2\,\mathrm{m/s}$. The solution mechanism is the same used in the simulations.

As regards the low-level controllers, we proceed as follows. Turtlebot3 robots are controlled using a linear state feedback for single integrators. In this way, we can handle collision among robots via barrier functions using the approach described in [124]. Then, in order to get the unicycle inputs, we utilize a near-identity diffeomorphism (see [124]). As for nano-quadrotors, a hierarchical controller has been considered. Specifically, a flatness-based position controller generates desired angular rates that are then actuated with a low-level PID control loop. The position controller receives as input a sufficiently smooth position trajectory, which is computed as a polynomial spline.

We performed two different experiments. In the first one, there are 3 ground robots and 2 aerial robots that must serve a total of 5 pickups and 5 deliveries. In Figure 5.12, we show a snapshot from the experiment. Then we performed a second, larger experiment with 7 ground robots and 2 aerial robots that must serve 10 pickups and 10 deliveries.

In Figure 5.13, we show snapshots from the second experiment.

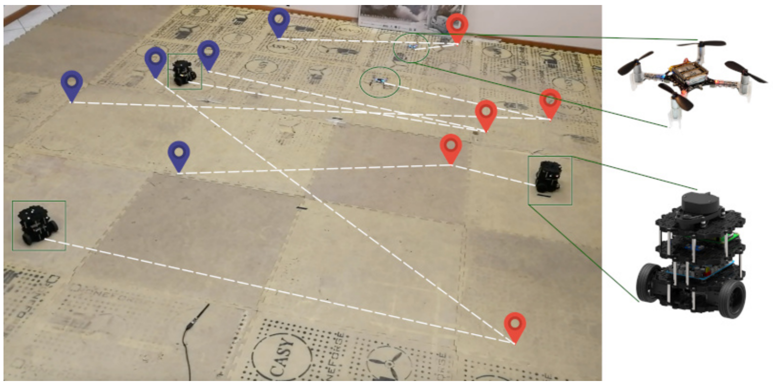


Figure 5.12: First experiment with ground and aerial robots for the PDVRP problem. Ground robots are indicated with a square, while aerial robots are delimited with circles. The red pins represent pickups, while the blue ones represent deliveries. The paths travelled by robots are depicted as dashed lines.

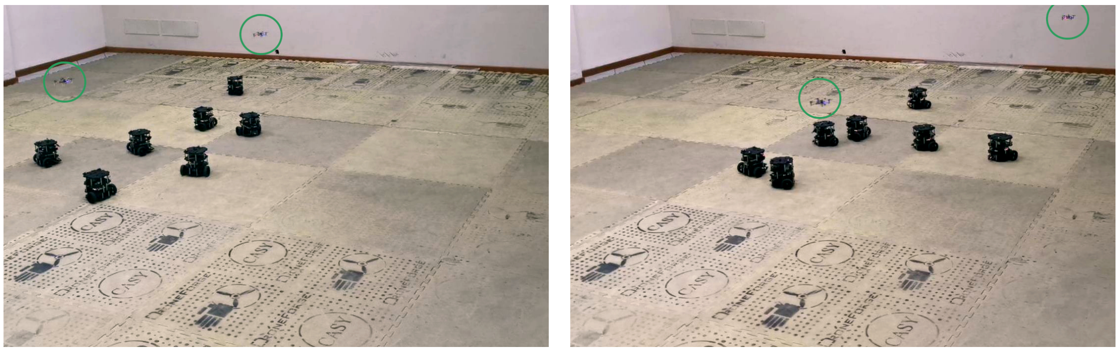


Figure 5.13: Snapshots from the second experiment. Left: robots have reached the pickup positions and perform the loading operation (simulated). Right: robots have reached the delivery positions and have terminated the mission.

5.3.8 Supplement: Conversion to Mixed-Integer Linear Program

Problem (5.4) is almost a mixed-integer linear program, except for the fact that the constraints (5.4h) and (5.4i) are nonlinear. However, from a computational point of view, the constraints (5.4h) and (5.4i) can be readily recast as linear ones. To achieve this, we use a standard procedure (see [7, 38]) that can be summarized as follows.

First, we introduce for each i, j, k the constraints $0 \leq B_i^j \leq \overline{B}_i$, where $\overline{B}_i \geq 0$ is any conservative upper bound on the total travel time of vehicle i selected so as to preserve the solutions of the original problem.[1] While this operation does not affect the problem, it introduces a bound on the value of each B^j. After defining for all i, j, k the scalars

[1]A simple possibility is to select $\overline{B}_i$ as the sum of all possible travel times t_i^{jk} from all i to all j, plus the service times d^j for all j.

$M_i^{jk} = \overline{B}_i + d^j + t_i^{jk}$ (or any larger number), the nonlinear constraint (5.4h) can be replaced with the linear one

$$B_i^k \geq B_i^j + d_j + t_i^{jk} - M_i^{jk}(1 - x_i^{jk}). \tag{5.17}$$

Equivalence of the constraint (5.4h) with (5.17) can be verified by noting that, for $x_i^{jk} = 1$, we obtain the desired constraint $B_i^k \geq B_i^j + d_j + t_i^{jk}$, while for $x_i^{jk} = 0$ the constraint (5.17) becomes $B_i^k \geq B_i^j + d_j + t_i^{jk} - M_i^{jk}$ (which is already implied by the constraints $B_i^k \geq 0$ and $B_i^j \geq 0$).

A similar reasoning can be applied to turn the constraint (5.4i) into a linear one. Let us define $\overline{W}_i^{jk} = \overline{Q}^j + q^k$ and $\underline{W}_i^{jk} = \overline{Q}^k - q^k - \underline{Q}^j$ (or any larger number), then constraint (5.4i) can be equivalently replaced with the pair of linear constraints

$$Q_i^k \geq Q_i^j + q^k - \overline{W}_i^{jk}(1 - x_i^{jk}), \tag{5.18a}$$

$$Q_i^k \leq Q_i^j + q^k + \underline{W}_i^{jk}(1 - x_i^{jk}). \tag{5.18b}$$

After introducing the additional constraints $0 \leq B_i^j \leq \overline{B}_i$ for all i, j, k and replacing (5.4h)–(5.4i) with their equivalent versions (5.17)–(5.18), problem (5.4) becomes a MILP.

Conclusions

In this thesis, we analyzed several optimization problems arising in peer-to-peer networks of agents that enjoy the constraint-coupled structure. These problems pose major challenges when the size of the network is large, when the network is time-varying or when convexity assumptions are dropped.

The algorithmic frameworks proposed in the thesis are based on the primal decomposition approach. Hence, we first covered the case of convex constraint-coupled problems in order to tackle structural challenges such as time-variability of the network and a-priori unknown cost functions. Then, we dropped the convexity assumption and considered more challenging NP-hard problems. We identified the mixed-integer constraint-coupled set-up as a prominent distributed optimization scenario that captures numerous decision problems of interest and developed innovative distributed strategies to compute high-quality solutions (as highlighted by Monte Carlo simulations), both asymptotically and in finite-time. These approaches can be of great interest in distributed decision contexts with privacy concerns. We demonstrated that the proposed distributed techniques can be successfully applied to highly relevant practical scenarios, such as stochastic microgrid control and distributed multi-robot pickup and delivery. This thesis represents a starting point toward more complex applications. In particular, the distributed primal decomposition approach, together with the restriction and violation principle, pave the way to the development of novel approaches for distributed control problems of interest that are intrinsically large-scale and nonconvex.

Future research directions include the extension to a general nonconvex constraint-coupled set-up, for which we only provided a first attempt. Moreover, it could be interesting to quantify the number of iterations needed for feasibility for the algorithms tackling mixed-integer problems. This ultimately depends on the convergence rate of the distributed subgradient-based algorithms for convex problems, for which a "feasibility convergence rate" is still unavailable. Such a contribution would find applications to distributed receding horizon control, for which controlled finite-time feasibility would imply recursive feasibility of the scheme with a a-priori fixed number of iterations. Finally, further developments of the distributed robotics toolbox may be useful to handle more general robotics contexts.

Appendix A

Optimization Basics

A.1 Lagrangian Duality

Consider a constrained optimization problem, addressed as primal problem, having the form

$$\begin{aligned} \min_{x \in X} \ & f(x) \\ \text{subj. to } \ & g(x) \leq \mathbf{0}, \end{aligned} \tag{A.1}$$

where $X \subseteq \mathbb{R}^d$ is a convex, compact set, $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function and $g : \mathbb{R}^d \to \mathbb{R}^S$ is such that each component $g_s : \mathbb{R}^d \to \mathbb{R}$, $s = 1, \dots, S$, is a convex (scalar) function. By adding an optimization variable, problem (A.1) can be equivalently written in the so-called *epigraph form*

$$\begin{aligned} \min_{x \in X, \rho} \ & \rho \\ \text{subj. to } \ & g(x) \leq \mathbf{0}, \\ & \rho \geq f(x), \end{aligned} \tag{A.2}$$

therefore a convex problem can be always assumed to have linear cost without loss of generality. For problem (A.1) we define the Lagrangian function as

$$\mathcal{L}(x, \mu) = f(x) + \mu^\top g(x). \tag{A.3}$$

The following optimization problem

$$\begin{aligned} \max_{\mu} \ & q(\mu) \\ \text{subj. to } \ & \mu \geq \mathbf{0} \end{aligned} \tag{A.4}$$

is called the dual of problem (A.1), where $q : \mathbb{R}^S \to \mathbb{R}$ is the so-called dual function,

$$q(\mu) = \inf_{x \in X} \mathcal{L}(x, \mu). \tag{A.5}$$

It can be shown that the domain of q (i.e. the set of μ such that $q(\mu) > -\infty$) is convex and that q is concave on its domain. A vector $\bar{\mu} \in \mathbb{R}^S$ is said to be a Lagrange multiplier if it holds $\bar{\mu} \geq \mathbf{0}$ and

$$\inf_{x \in X} \mathcal{L}(x, \bar{\mu}) = \inf_{x \in X \,:\, g(x) \leq \mathbf{0}} f(x).$$

Denote by $f^\star$ the optimal cost of the primal problem and by $q^\star$ the optimal cost of the dual problem It can be shown that the following inequality holds [10],

$$q^\star \leq f^\star, \tag{A.6}$$

which is called weak duality. When in (A.6) the equality holds, then we say that strong duality holds and, thus, solving the primal problem (A.1) is equivalent to solving its dual formulation (A.4).

A.2 Convergence Rates

Let $\{x^t\}_{t \in \mathbb{N}}$ be a sequence of vectors in $\mathbb{R}^n$. Assume the sequence converges to some $\bar{x} \in \mathbb{R}^n$. We say that the sequence converges *linearly* to $\bar{x}$ if there exists a number $\eta \in (0, 1)$ such that

$$\lim_{t \to \infty} \frac{\|x^{t+1} - \bar{x}\|}{\|x^t - \bar{x}\|} = \eta. \tag{A.7}$$

If $\eta = 0$ we say that the sequence converges *superlinearly* to $\bar{x}$, while if $\eta = 1$ it converges *sublinearly* to x. It is common to denote the rate of convergence using the big-O notation. For instance, a sequence that goes to zero as $O(1/t)$ converges sublinearly, while a sequence that goes to zero as $O(\lambda^t)$, with $\lambda \in (0, 1)$, converges linearly.

A.3 Linear Programs and Mixed-Integer Linear Programs

A Linear Program (LP) is an optimization problem with linear cost function and linear constraints:

$$\begin{aligned} &\min_{x} \; c^\top x \\ &\text{subj. to } \; a_k^\top x \leq b_k, \qquad k = 1, \ldots, K, \end{aligned} \tag{A.8}$$

where $c \in \mathbb{R}^d$ is the cost vector and $a_k \in \mathbb{R}^d$ and $b_k \in \mathbb{R}$ describe K inequality constraints. In the subsequent discussion, we assume that $d \leq K$. The feasible set $\mathcal{X}$ of problem (A.8) is the set of vectors satisfying all the constraints, i.e.

$$\mathcal{X} \triangleq \{x \in \mathbb{R}^d \mid a_k^\top x \leq b_k \text{ for all } k \in \{1, \ldots, K\}\}.$$

Note that $\mathcal{X}$ is a polyhedron, for which the following definition of vertex can be given.

Definition A.1. *A vector $\tilde{x} \in \mathbb{R}^d$ is a vertex of $\mathcal{X}$ if there exists some $c \in \mathbb{R}^d$ such that $c^\top \tilde{x} < c^\top x$ for all $x \in \mathcal{X}$ with $x \neq \tilde{x}$.* $\triangle$

If problem (A.8) admits an optimal solution, it can be shown that there exists an optimal vertex, i.e. a vertex which is an optimal solution of the problem (see e.g. [13, Theorem 2.7]). Let $x^\star$ be an optimal vertex of problem (A.8). Then, it is a standard result in linear programming theory that there exists an index set $\{\ell_1, \ldots, \ell_d\} \subset \{1, \ldots, K\}$, with cardinality d, such that $x^\star$ is the unique optimal vertex of the problem

$$\begin{aligned} \min_x \; & c^\top x \\ \text{subj. to } \; & a_{\ell_h}^\top x \leq b_{\ell_h}, \qquad h = 1, \ldots, d, \end{aligned}$$

which is a relaxed version of problem (A.8) in which only d constraints are considered. In addition, the vectors a_{ℓ_h}, $h = 1, \ldots, d$, are linearly independent, so that they form a basis of $\mathbb{R}^d$. By analogy, the constraints $a_{\ell_h}^\top x \leq b_{\ell_h}$, $h = 1, \ldots, d$, are called a *basis* of the point $x^\star$. Due to the optimality of $x^\star$, we call it also a basis of problem (A.8). To compactly denote such basis, we introduce a matrix $P \in \mathbb{R}^{d \times d}$, obtained by stacking the row vectors $a_{\ell_h}^\top$, and a vector $q \in \mathbb{R}^d$, obtained by stacking the scalars b_{ℓ_h}, i.e.

$$P = \begin{bmatrix} a_{\ell_1}^\top \\ \vdots \\ a_{\ell_d}^\top \end{bmatrix}, \qquad q = \begin{bmatrix} b_{\ell_1} \\ \vdots \\ b_{\ell_d} \end{bmatrix}.$$

Then, $x^\star = P^{-1} q$, and we say that the tuple (P, q) is a basis of (A.8).

A.3.1 Dual Degeneracy and Lexicographic Ordering

If problem (A.8) has multiple optimal solutions, we say that the LP is *dual degenerate*. In presence of dual degeneracy, it is not trivial to guarantee convergence of distributed algorithms to the same optimal solution. In order to overcome this issue, it is possible to rely on the lexicographic ordering of vectors. We now give some definitions.

Definition A.2. *A vector $v \in \mathbb{R}^n$ is said to be* lexicographically positive *(or* lex-positive*)* *if $v \neq \mathbf{0}$ and the first non-zero component of v is positive. In symbols:*

$$u \overset{\text{LEX}}{>} \mathbf{0}.$$

A vector $u \in \mathbb{R}^n$ is said to be lexicographically larger *(resp.* smaller*) than another vector $v \in \mathbb{R}^n$ if $u - v$ is lex-positive (resp. $v - u$ is lex-positive), or, equivalently, if $u \neq v$ and the first nonzero component of $u - v$ is positive (resp., negative). In symbols:*

$$u \overset{\text{LEX}}{>} v \quad \text{or} \quad u \overset{\text{LEX}}{<} v.$$

Given a set of vectors $\{v_1, \dots, v_r\}$, the lexicographic minimum is the element v_i such that $v_j \overset{\text{LEX}}{>} v_i$ for all $j \neq i$. In symbols:

$$v_i = \text{lex-min}\{v_1, \dots, v_r\}. \qquad \triangle$$

Now, consider the optimal solution set of problem (A.8), i.e. $\mathcal{X}^\star \triangleq \{x \in \mathcal{X} \mid c^\top x \leq c^\top x' \text{ for all } x' \in \mathcal{X}\} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the feasible set of problem (A.8). Among all the optimal solutions in $\mathcal{X}^\star$, it is possible to compute the lexicographically minimal one, i.e. $\text{lex-min}(\mathcal{S}^\star)$. It turns out that finding $\text{lex-min}(\mathcal{S}^\star)$ is equivalent to finding the (unique) optimal solution to a modified (non dual-degenerate) version of the original problem (A.8), where the cost vector c is perturbed to $c' = c + \Delta$, with Δ a lexicographic perturbation vector:

$$\Delta^\top = [\Delta_0 \ \Delta_0^2 \ \dots \ \Delta_0^d],$$

for a sufficiently small $\Delta_0 > 0$ (see [59]). Therefore, the lex-optimal solution of problem (A.8) is the *unique* optimal solution of the problem with perturbed cost

$$\begin{aligned} \min_x \ & (c + \Delta)^\top x \\ \text{subj. to } & a_k^\top x \leq b_k, \qquad k = 1, \dots, K. \end{aligned} \tag{A.9}$$

Thus, the lex-optimal solution of problem (A.8) exists if and only if problem (A.9) admits an optimal solution. Moreover, the optimal solution of (A.9) is attained at a vertex of (A.8), therefore it is an optimal vertex of problem (A.8).

A.3.2 Mixed-Integer Linear Programs and Duality

A Mixed-Integer Linear Program (MILP) is a Linear Program where some of the variables take on integer values only, i.e.

$$\begin{aligned} \min_x \ & c^\top x \\ \text{subj. to } \ & a_k^\top x \le b_k, \qquad && k = 1, \dots, K, \\ & x_j \in \mathbb{Z}, && \forall j \in \mathcal{J}, \end{aligned} \tag{A.10}$$

where $\mathcal{J} \subseteq \{1, \dots, d\}$ is the index set of integer variables. MILPs are NP-hard problems, thus enumeration techniques (such as branch and bound) are required to compute an optimal solution (see e.g. [13]). Integer variables can be used to model a wide range of situations, including logic constraints, switching conditions, etc.

Denote by P_I the feasible set of problem (A.10), which is a so-called mixed-integer polyhedron. The optimal cost of problem (A.10) is equal to the optimal cost of the following convex problem

$$\begin{aligned} \min_x \ & c^\top x \\ \text{subj. to } \ & x \in \text{conv}(P_I), \end{aligned} \tag{A.11}$$

where $\text{conv}(P_I)$ denotes the convex hull of P_I. This property is important for the development of cutting-plane algorithms to solve problem (A.10).

Another important property is related to the application of Lagrange duality to MILPs. Consider a MILP of the form

$$\begin{aligned} \min_x \ & c^\top x \\ \text{subj. to } \ & x \in X, \\ & Ax \le b, \end{aligned} \tag{A.12}$$

where $X \subset \mathbb{R}^d$ is a compact mixed-integer polyhedron, $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. Consider also the following convexification of problem (A.12),

$$\begin{aligned} \min_x \ & c^\top x \\ \text{subj. to } \ & x \in \text{conv}(X). \\ & Ax \le b, \end{aligned} \tag{A.13}$$

Note that the constraint set of problem (A.13) is *not* the convex hull of the constraint set of MILP (A.12). Indeed, the constraint $Ax \le b$ has been kept separate from the computation of the convex hull. The dual problem of (A.12) is

$$\max_{\mu \geq 0} \Big[\min_{x \in X} \big(c^\top x + \mu^\top (Ax - b) \big) \Big] = \max_{\mu \geq 0} \Big[\min_{x \in \text{conv}(X)} \big(c^\top x + \mu^\top (Ax - b) \big) \Big], \qquad \text{(A.14)}$$

where the equality follows by linearity of the cost in the inner minimization. Thus, problems (A.12) and (A.13) have the same dual problem. Denote by $q^\star$ the dual optimal cost and denote by $f^\star_{\text{MI}}$ the optimal cost of problem (A.12). In general, only weak duality can be assessed, thus $q^\star \leq f^\star_{\text{MI}}$. However, strong duality holds for problem (A.13), therefore its optimal cost is equal to $q^\star$ [52].

Ringraziamenti personali

Il primo ringraziamento va senza alcun dubbio al mio tutor Giuseppe, innanzitutto perché in questi tre anni mi ha aiutato a maturare scientificamente e mi ha continuamente fornito tantissimi stimoli. Lo ringrazio anche perché mi ha dato un sacco di opportunità, a partire dalla partecipazione alle conferenze, ai corsi di dottorato, ma anche per avermi dato la possibilità di poter lavorare in un ambiente di eccellenza.

Ringrazio i miei colleghi con cui ogni giorno ho avuto il piacere di lavorare. Sicuramente il percorso di dottorato non sarebbe stato lo stesso senza di voi!

Un grazie speciale va alla Comunità Emmanuel di Lecce, che dal 2009 è ormai la mia seconda casa, e a tutte le persone che ne fanno parte.

Ringrazio anche Katia, Marco, Salvatore, Mariangela ed Enzo, che sono stati sempre presenti in questi tre anni, e tutti i miei amici leccesi che anche se sono a distanza sono sempre vicini.

Il ringraziamento più grande va però a mia moglie Scharon, che mi ha sostenuto nei momenti di sconforto e ha gioito insieme a me nei momenti di felicità. Molti dei miei successi sono dovuti in gran parte a lei e al suo esserci per me.

Bibliography

[1] L. Abbatecola, M. P. Fanti, G. Pedroncelli, and W. Ukovich, *A distributed cluster-based approach for pick-up services*, IEEE Transactions on Automation Science and Engineering **16** (2018), no. 2, 960–971.

[2] E. Aertbeliën and J. De Schutter, *eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 1540–1546.

[3] S. A. Alghunaim, K. Yuan, and A. H. Sayed, *Dual coupled diffusion for distributed optimization with affine constraints*, IEEE Conference on Decision and Control, 2018, pp. 829–834.

[4] A. Arsie, K. Savla, and E. Frazzoli, *Efficient routing algorithms for multiple vehicles with no explicit communications*, IEEE Transactions on Automatic Control **54** (2009), no. 10, 2302–2317.

[5] X. Bai, M. Cao, W. Yan, and S. S. Ge, *Efficient routing for precedence-constrained package delivery for heterogeneous vehicles*, IEEE Transactions on Automation Science and Engineering **17** (2019), no. 1, 248–260.

[6] A. Beck and L. Tetruashvili, *On the convergence of block coordinate descent type methods*, SIAM Journal on Optimization **23** (2013), no. 4, 2037–2060.

[7] A. Bemporad and M. Morari, *Control of systems integrating logic, dynamics, and constraints*, Automatica **35** (1999), no. 3, 407–427.

[8] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Academic Press, 1982.

[9] D. P Bertsekas, *Network optimization: continuous and discrete models*, Citeseer, 1998.

[10] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, 1999.

[11] ———, *Convex optimization algorithms*, Athena Scientific, 2015.

[12] D. P Bertsekas, A. Nedić, A. E Ozdaglar, et al., *Convex analysis and optimization*, Athena Scientific, 2003.

[13] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, Vol. 6, Athena Scientific Belmont, MA, 1997.

[14] P. Bianchi and J. Jakubowicz, *Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization*, IEEE Transactions on Automatic Control **58** (2013), no. 2, 391–405.

[15] S. Bolognani and S. Zampieri, *A distributed control strategy for reactive power compensation in smart microgrids*, IEEE Transactions on Automatic Control **58** (2013), no. 11, 2818–2833.

[16] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[17] N. Buckman, H.-L. Choi, and J. P. How, *Partial replanning for decentralized dynamic task allocation*, Aiaa scitech 2019 forum, 2019, pp. 0915.

[18] M. Bürger, G. Notarstefano, and F. Allgöwer, *A polyhedral approximation framework for convex and robust distributed optimization*, IEEE Transactions on Automatic Control **59** (2014), no. 2, 384–395.

[19] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, *A distributed simplex algorithm for degenerate linear programs and multi-agent assignments*, Automatica **48** (2012), no. 9, 2298–2304.

[20] A. Camisa, A. Benevento, and G. Notarstefano, *Constraint-coupled optimization with unknown costs: A distributed primal decomposition approach*, arXiv preprint arXiv:2104.06341 (2021).

[21] A. Camisa, F. Farina, I. Notarnicola, and G. Notarstefano, *Distributed constraint-coupled optimization over random time-varying graphs via primal decomposition and block subgradient approaches*, IEEE Conference on Decision and Control, 2019, pp. 6374–6379.

[22] ______, *Distributed constraint-coupled optimization via primal decomposition over random time-varying graphs*, Automatica (2021). (to appear).

[23] A. Camisa, I. Notarnicola, and G. Notarstefano, *A primal decomposition method with suboptimality bounds for distributed mixed-integer linear programming*, IEEE Conference on Decision and Control, 2018, pp. 3391–3396.

[24] ______, *Distributed primal decomposition for large-scale MILPs*, IEEE Transactions on Automatic Control (2021).

[25] A. Camisa and G. Notarstefano, *A distributed primal decomposition scheme for nonconvex optimization*, IFAC-PapersOnLine **52** (2019), no. 20, 315–320.

[26] ______, *Primal decomposition and constraint generation for asynchronous distributed mixed-integer linear programming*, IEEE European Control Conference, 2019, pp. 77–82.

[27] ______, *A distributed mixed-integer framework to stochastic optimal microgrid control*, arXiv preprint arXiv:2104.06346 (2021).

[28] A. Camisa, A. Testa, and G. Notarstefano, *Multi-robot pickup and delivery via distributed resource allocation*, arXiv preprint arXiv:2104.02415 (2021).

[29] R. Carli and M. Dotoli, *Distributed alternating direction method of multipliers for linearly constrained optimization over a network*, IEEE Control Systems Letters **4** (2020), no. 1, 247–252.

[30] G. A Casan, E. Cervera, A. A Moughlbay, J. Alemany, and P. Martinet, *ROS-based online robot programming for remote education and training*, IEEE International Conference on Robotics and Automation, 2015, pp. 6101–6106.

[31] M. Chamanbaz, G. Notarstefano, and R. Bouffanais, *Randomized constraints consensus for distributed robust linear programming*, IFAC-PapersOnLine **50** (2017), no. 1, 4973–4978.

[32] M. Chamanbaz, G. Notarstefano, F. Sasso, and R. Bouffanais, *Randomized constraints consensus for distributed robust mixed-integer programming*, IEEE Transactions on Control of Network Systems (2020).

[33] T.-H. Chang, M. Hong, and X. Wang, *Multi-agent distributed optimization via inexact consensus ADMM*, IEEE Transactions on Signal Processing **63** (2014), no. 2, 482–497.

[34] T.-H. Chang, A. Nedić, and A. Scaglione, *Distributed constrained optimization by consensus-based primal-dual perturbation method*, IEEE Transactions on Automatic Control **59** (2014), no. 6, 1524–1538.

[35] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, *A distributed version of the hungarian method for multirobot assignment*, IEEE Transactions on Robotics **33** (2017), no. 4, 932–947.

[36] B. Coltin and M. Veloso, *Online pickup and delivery planning with transfers for mobile robots*, Workshops at the twenty-seventh aaai conference on artificial intelligence, 2013.

[37] S. R. Cominesi, M. Farina, L. Giulioni, B. Picasso, and R. Scattolini, *A two-layer stochastic model predictive control scheme for microgrids*, IEEE Transactions on Control Systems Technology **26** (2017), no. 1, 1–13.

[38] J.-F. Cordeau, *A branch-and-cut algorithm for the dial-a-ride problem*, Operations Research **54** (2006), no. 3, 573–586.

[39] C. D. Dang and G. Lan, *Stochastic block mirror descent methods for nonsmooth and stochastic optimization*, SIAM Journal on Optimization **25** (2015), no. 2, 856–881.

[40] P. Di Lorenzo and G. Scutari, *NEXT: in-network nonconvex optimization*, IEEE Transactions on Signal and Information Processing over Networks **2** (2016), no. 2, 120–136.

[41] J. C. Duchi, A. Agarwal, and M. J. Wainwright, *Dual averaging for distributed optimization: convergence analysis and network scaling*, IEEE Transactions on Automatic Control **57** (2012), no. 3, 592–606.

[42] E. Erős, M. Dahl, K. Bengtsson, A. Hanna, and P. Falkman, *A ROS2 based communication architecture for control in collaborative and intelligent automation systems*, Procedia Manufacturing **38** (2019), 349–357.

[43] E. Erős, M. Dahl, A. Hanna, A. Albo, P. Falkman, and K. Bengtsson, *Integrated virtual commissioning of a ROS2-based collaborative and intelligent automation system*, IEEE International Conference on Emerging Technologies and Factory Automation, 2019, pp. 407–413.

[44] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, *Dual decomposition for multi-agent distributed optimization with coupling constraints*, Automatica **84** (2017), 149–158.

[45] A. Falsone, K. Margellos, and M. Prandini, *A distributed iterative algorithm for multi-agent MILPs: finite-time feasibility and performance characterization*, IEEE Control Systems Letters **2** (2018), no. 4, 563–568.

[46] ———, *A decentralized approach to multi-agent MILPs: finite-time feasibility and performance guarantees*, Automatica **103** (2019), 141–150.

[47] A. Falsone, I. Notarnicola, G. Notarstefano, and M. Prandini, *Tracking-ADMM for distributed constraint-coupled optimization*, Automatica **117** (2020), 108962.

[48] F. Farina, A. Camisa, A. Testa, I. Notarnicola, and G. Notarstefano, *DISROPT: a Python framework for distributed optimization*, IFAC world congress, 2020.

[49] A. Farinelli, A. Contini, and D. Zorzi, *Decentralized task assignment for multi-item pickup and delivery in logistic scenarios*, International conference on autonomous agents and multiagent systems, 2020, pp. 1843–1845.

[50] M. H. F. b. M. Fauadi, S. H. Yahaya, and T. Murata, *Intelligent combinatorial auctions of decentralized task assignment for AGV with multiple loading capacity*, IEEJ Transactions on electrical and electronic Engineering **8** (2013), no. 4, 371–379.

[51] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, *Gossip algorithms for heterogeneous multi-vehicle routing problems*, Nonlinear Analysis: Hybrid Systems **10** (2013), 156–174.

[52] A. M. Geoffrion, *Lagrangean relaxation for integer programming*, Mathematical programming study, 1974, pp. 82–114.

[53] P. Giselsson and A. Rantzer, *Large-scale and distributed optimization*, Vol. 2227, Springer, 2018.

[54] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, *Fast scheduling of robot teams performing tasks with temporospatial constraints*, IEEE Transactions on Robotics **34** (2018), no. 1, 220–239.

[55] V. Grabe, M. Riedel, H. H Bülthoff, P. R. Giordano, and A. Franchi, *The TeleKyb framework for a modular and extendible ROS-based quadrotor control*, European Conference on Mobile Robots, 2013, pp. 19–25.

[56] A. Ham, *Drone-based material transfer system in a robotic mobile fulfillment center*, IEEE Transactions on Automation Science and Engineering **17** (2019), no. 2, 957–965.

[57] B. Heap and M. Pagnucco, *Repeated sequential single-cluster auctions with dynamic tasks for multi-robot task allocation with pickup and delivery*, German conference on multiagent system technologies, 2013, pp. 87–100.

[58] D. Jakovetić, J. Xavier, and J. M. Moura, *Fast distributed gradient methods*, IEEE Transactions on Automatic Control **59** (2014), no. 5, 1131–1146.

[59] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski, *Lexicographic perturbation for multiparametric linear programming with applications to control*, Automatica **43** (2007), no. 10, 1808–1816.

[60] N. Kamra, T. K. S. Kumar, and N. Ayanian, *Combinatorial problems in multirobot battery exchange systems*, IEEE Transactions on Automation Science and Engineering **15** (2017), no. 2, 852–862.

[61] S.-J. Kim and G. B. Giannakis, *Scalable and robust demand response with mixed-integer constraints*, IEEE Transactions on Smart Grid **4** (2013), no. 4, 2089–2099.

[62] N. Koenig and A. Howard, *Design and use paradigms for gazebo, an open-source multi-robot simulator*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, pp. 2149–2154.

[63] P. O. Kriett and M. Salani, *Optimal control of a residential microgrid*, Energy **42** (2012), no. 1, 321–330.

[64] Y. Kuwata and J. P How, *Cooperative distributed robust trajectory optimization using receding horizon MILP*, IEEE Transactions on Control Systems Technology **19** (2010), no. 2, 423–431.

[65] H. Lakshmanan and D. P. De Farias, *Decentralized resource allocation in dynamic networks of agents*, SIAM Journal on Optimization **19** (2008), no. 2, 911–940.

[66] S. Liang, L. Y. Wang, and G. Yin, *Distributed smooth convex optimization with coupled constraints*, IEEE Transactions on Automatic Control **65** (2020), no. 1, 347–353.

[67] M. Liu, H. Ma, J. Li, and S. Koenig, *Task and path planning for multi-agent pickup and delivery*, International conference on autonomous agents and multiagent systems, 2019, pp. 1152–1160.

[68] L. Luo, N. Chakraborty, and K. Sycara, *Distributed algorithms for multirobot task assignment with task deadline constraints*, IEEE Transactions on Automation Science and Engineering **12** (2015), no. 3, 876–888.

[69] Y. Maruyama, S. Kato, and T. Azumi, *Exploring the performance of ROS2*, International Conference on Embedded Software, 2016, pp. 1–10.

[70] M. Marzband, M. Ghadimi, A. Sumper, and J. L. Domínguez-García, *Experimental validation of a real-time energy management system using multi-period gravitational search algorithm for microgrids in islanded mode*, Applied Energy **128** (2014), 164–174.

[71] D. Mateos-Núnez and J. Cortés, *Distributed saddle-point subgradient algorithms with Laplacian averaging*, IEEE Transactions on Automatic Control **62** (2017), no. 6, 2720–2735.

[72] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*, Vol. 33, Princeton University Press, 2010.

[73] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, *Comprehensive simulation of quadrotor UAVs using ROS and gazebo*, International Conference on Simulation, Modeling, and Programming for Autonomous Robots, 2012, pp. 400–411.

[74] R. R. Meyer, *On the existence of optimal solutions to integer and mixed-integer programming problems*, Mathematical Programming **7** (1974), no. 1, 223–235.

[75] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, *D-ADMM: a communication-efficient distributed algorithm for separable optimization*, IEEE Transactions on Signal Processing **61** (2013), no. 10, 2718–2723.

[76] I Necoara, Y Nesterov, and F Glineur, *A random coordinate descent method on large-scale optimization problems with linear constraints*, 2014.

[77] I. Necoara, *Random coordinate descent algorithms for multi-agent convex optimization over networks*, IEEE Transactions on Automatic Control **58** (2013), no. 8, 2001–2012.

[78] I. Necoara and V. Nedelcu, *On linear convergence of a distributed dual gradient algorithm for linearly constrained separable convex problems*, Automatica **55** (2015), 209–216.

[79] A. Nedić and J. Liu, *Distributed optimization for control*, Annual Review of Control, Robotics, and Autonomous Systems **1** (2018), 77–103.

[80] A. Nedić and A. Ozdaglar, *Approximate primal solutions and rate analysis for dual subgradient methods*, SIAM Journal on Optimization **19** (2009), no. 4, 1757–1780.

[81] ———, *Distributed subgradient methods for multi-agent optimization*, IEEE Transactions on Automatic Control **54** (2009), no. 1, 48–61.

[82] T. A. Nguyen and M. L. Crow, *Stochastic optimization of renewable-based microgrid operation incorporating battery operating cost*, IEEE Transactions on Power Systems **31** (2015), no. 3, 2289–2296.

[83] I. Notarnicola and G. Notarstefano, *Constraint-coupled distributed optimization: a relaxation and duality approach*, IEEE Transactions on Control of Network Systems **7** (2019), no. 1, 483–492.

[84] G. Notarstefano and F. Bullo, *Distributed abstract optimization via constraints consensus: theory and applications*, IEEE Transactions on Automatic Control **56** (2011), no. 10, 2247–2261.

[85] G. Notarstefano, M. Egerstedt, and M Haque, *Containment in leader–follower networks with switching communication topologies*, Automatica **47** (2011), no. 5, 1035–1040.

[86] G. Notarstefano, I. Notarnicola, and A. Camisa, *Distributed optimization for smart cyber-physical networks*, Foundations and Trends® in Systems and Control **7** (2019), no. 3, 253–383.

[87] OPSD, *Open Power System Data Time Series*, 2020. https://data.open-power-system-data.org/time_series, October 6th, 2020.

[88] A. Ouammi, H. Dagdougui, L. Dessaint, and R. Sacile, *Coordinated model predictive-based power flows control in a cooperative network of smart microgrids*, IEEE Transactions on Smart grid **6** (2015), no. 5, 2233–2244.

[89] A. Parisio, E. Rikos, and L. Glielmo, *A model predictive control approach to microgrid operation optimization*, IEEE Transactions on Control Systems Technology **22** (2014), no. 5, 1813–1827.

[90] ———, *Stochastic model predictive control for economic/environmental operation management of microgrids: an experimental case study*, Journal of Process Control **43** (2016), 24–37.

[91] J. J. Park and B. Kuipers, *A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment*, IEEE International Conference on Robotics and Automation, 2011, pp. 4896–4902.

[92] S. N. Parragh, K. F. Doerner, and R. F. Hartl, *A survey on pickup and delivery models part II: transportation between pickup and delivery locations*, Journal für Betriebswirtschaft **58** (2008), no. 2, 81–117.

[93] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D Hager, *CoSTAR: Instructing collaborative robots with behavior trees and vision*, IEEE International Conference on Robotics and Automation, 2017, pp. 564–571.

[94] V. Pillac, M. Gendreau, C. Guéret, and A. L Medaglia, *A review of dynamic vehicle routing problems*, European Journal of Operational Research **225** (2013), no. 1, 1–11.

[95] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y Ng, *ROS: an open-source Robot Operating System*, ICRA workshop on open source software, 2009, pp. 5.

[96] J. B Rawlings and D. Q Mayne, *Model predictive control: theory and design*, Nob Hill Pub. Madison, Wisconsin, 2009.

[97] M. Razaviyayn, M. Hong, and Z.-Q. Luo, *A unified convergence analysis of block successive minimization methods for nonsmooth optimization*, SIAM Journal on Optimization **23** (2013), no. 2, 1126–1153.

[98] M. Reke, D. Peter, J. Schulte-Tigges, S. Schiffer, A. Ferrein, T. Walter, and D. Matheis, *A self-driving car architecture in ROS2*, International SAUPEC/RobMech/PRASA Conference, 2020, pp. 1–6.

[99] A. Richards and J. P. How, *Robust distributed model predictive control*, International Journal of control **80** (2007), no. 9, 1517–1531.

[100] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming **144** (2014), no. 1-2, 1–38.

[101] U. Ritzinger, J. Puchinger, and R. F Hartl, *A survey on dynamic and stochastic vehicle routing problems*, International Journal of Production Research **54** (2016), no. 1, 215–231.

[102] G. Scutari and Y. Sun, *Distributed nonconvex constrained optimization over time-varying digraphs*, Mathematical Programming **176** (2019), no. 1-2, 497–544.

[103] A. Settimi and L. Pallottino, *A subgradient based algorithm for distributed task assignment for heterogeneous mobile robots*, Ieee conference on decision and control, 2013, pp. 3665–3670.

[104] T. W. Sherson, R. Heusdens, and W B. Kleijn, *On the distributed method of multipliers for separable convex optimization problems*, IEEE Transactions on Signal and Information Processing over Networks **5** (2019), no. 3, 495–510.

[105] W. Shi, Q. Ling, G. Wu, and W. Yin, *EXTRA: an exact first-order algorithm for decentralized consensus optimization*, SIAM Journal on Optimization **25** (2015), no. 2, 944–966.

[106] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, *On the linear convergence of the ADMM in decentralized consensus optimization*, IEEE Transactions on Signal Processing **62** (2014), no. 7, 1750–1761.

[107] E. Shirazi and S. Jadid, *Cost reduction and peak shaving through domestic load shifting and DERs*, Energy **124** (2017), 146–159.

[108] G. J Silverman, *Primal decomposition of mathematical programs by resource allocation: I – basic theory and a direction-finding procedure*, Operations Research **20** (1972), no. 1, 58–74.

[109] A. Simonetto, *Smooth strongly convex regression*, IEEE European Signal Processing Conference, 2021, pp. 2130–2134.

[110] A. Simonetto and H. Jamali-Rad, *Primal recovery from consensus-based dual decomposition for distributed convex optimization*, Journal of Optimization Theory and Applications **168** (2016), no. 1, 172–197.

[111] R. M Starr, *Quasi-equilibria in markets with non-convex preferences*, Econometrica: Journal of the Econometric Society (1969), 25–38.

[112] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, *A simple effective heuristic for embedded mixed-integer quadratic programming*, International Journal of Control (2017), 1–11.

[113] Z. Talebpour and A. Martinoli, *Adaptive risk-based replanning for human-aware multi-robot task allocation with local perception*, IEEE Robotics and Automation Letters **4** (2019), no. 4, 3790–3797.

[114] A. B. Taylor, J. M. Hendrickx, and F. Glineur, *Smooth strongly convex interpolation and exact worst-case performance of first-order methods*, Mathematical Programming **161** (2017), no. 1-2, 307–345.

[115] A. Testa, A. Camisa, and G. Notarstefano, *ChoiRbot: A ROS 2 toolbox for cooperative robotics*, IEEE Robotics and Automation Letters **6** (2021), no. 2, 2714–2720.

[116] A. Testa and G. Notarstefano, *Generalized assignment for multi-robot systems via distributed branch-and-price*, arXiv preprint arXiv:2004.11857 (2020).

[117] A. Testa, A. Rucco, and G. Notarstefano, *Distributed mixed-integer linear programming via cut generation and constraint exchange*, IEEE Transactions on Automatic Control **65** (2019), no. 4, 1456–1467.

[118] P. Toth and D. Vigo, *The vehicle routing problem*, SIAM, 2002.

[119] Q. Tran-Dinh, I. Necoara, and M. Diehl, *A dual decomposition algorithm for separable nonconvex optimization using the penalty function framework*, IEEE Conference on Decision and Control, 2013, pp. 2372–2377.

[120] F. Vanderbeck, *Implementing mixed integer column generation*, Column generation, 2005, pp. 331–358.

[121] R. Vujanic, P. M. Esfahani, P. J Goulart, S. Mariéthoz, and M. Morari, *A decomposition method for large scale MILPs, with performance guarantees and a power system application*, Automatica **67** (2016), 144–156.

[122] H.-T. Wai, J. Lafond, A. Scaglione, and E. Moulines, *Decentralized Frank–Wolfe algorithm for convex and nonconvex problems*, IEEE Transactions on Automatic Control **62** (2017), no. 11, 5522–5537.

[123] Z. Wang and C. J. Ong, *Distributed model predictive control of linear discrete-time systems with local and global constraints*, Automatica **81** (2017), 184–195.

[124] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, *The robotarium: globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems*, IEEE Control Systems Magazine **40** (2020), no. 1, 26–44.

[125] A. Zakariazadeh, S. Jadid, and P. Siano, *Smart microgrid energy and reserve scheduling with demand response using stochastic optimization*, International Journal of Electrical Power & Energy Systems **63** (2014), 523–533.

[126] Y. Zhang and M. M Zavlanos, *A consensus-based distributed augmented lagrangian method*, IEEE Conference on Decision and Control, 2018, pp. 1763–1768.

[127] M. Zhu and S. Martínez, *On distributed convex optimization under inequality and equality constraints*, IEEE Transactions on Automatic Control **57** (2012), no. 1, 151–164.

[128] M. Zhu and S. Martínez, *An approximate dual subgradient algorithm for multi-agent non-convex optimization*, IEEE Transactions on Automatic Control **58** (2013), no. 6, 1534–1539.