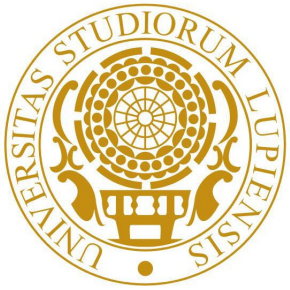
Università del Salento



Department of Engineering for Innovation

Doctoral Programme in Engineering of Complex Systems

Block-Coordinate Methods and Duality for Asynchronous, Big-Data and Constrained Distributed Optimization

Doctoral Dissertation of

Ivano Notarnicola

Tutor:
prof. Giuseppe Notarstefano

Coordinator of the Ph.D. Programme:
prof. Giulio Avanzini

Chair of the Doctoral School:
prof. Antonio Leaci

A.A. 2016-2017 – XXX Cycle

Abstract

In this thesis we consider classes of optimization problems that are of great interest in control, signal processing and learning over networks. We propose and analyze distributed optimization algorithms that tackle specific challenges associated to each considered set-up. In particular, we first consider an optimization problem in which the cost function is the sum of local contributions depending on a common variable and propose asynchronous distributed algorithms that extend the classical distributed dual decomposition. In the proposed approach, agents run their local computation without any time synchronization, based on proximal operators and block-coordinate methods that allow for handling local constraints and fixed step-sizes. Then, we focus on "big-data" problems in which the dimension of the decision variable is "high", possibly depending on the number of agents. We start from a structured optimization set-up where costs and constraints depend only on a small portion of the optimization variable, and propose primal and dual distributed methods for asynchronous networks, based on block-coordinate techniques. Then, we consider a more general big-data set-up (possibly nonconvex) in which each local cost function depends on the entire variable, but nodes can communicate and optimize only single blocks. For this set-up we propose a distributed algorithm that leverages computations and communications performed in a block-wise fashion and requires a block-wise average scheme to enforce consensus on the solution estimates. Finally, we investigate a general set-up that arises in dynamic optimization problems that are strictly related to important applications as, e.g., distributed control. We consider a set-up in which agents in a network want to minimize the sum of local convex cost functions, each one depending on a local variable, subject to local and coupling constraints, with the latter involving all the decision variables. We propose a novel distributed algorithm based on a relaxation of the original problem and an elegant exploration of Lagrangian duality theory which allows agents to recover their portion of an optimal problem solution. For all these set-ups we apply the proposed schemes to learning and control application scenarios to highlight their benefits and performances.

Keywords: Distributed Optimization, Asynchronous Algorithms, Block-Coordinate Methods, Duality, Big-data Optimization, Gradient Tracking, Penalty Approach, Distributed MPC, Big-Data Analytics, Demand-Side Management.

Acknowledgment

The work of this thesis 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

Motivating Scenarios and Literature

In the last years we have witnessed a dramatically increasing interest in designing "smart" objects including phones, cars, drones. With the diffusion of these technological innovations, a number of new challenges have been introduced to transform these objects into smart systems, farms, industries, buildings, cities.

A novel keyword has been introduced to indicate such a scenario, namely Cyber Physical Networks (CPNs). The distinctive feature of CPNs is that a great advantage can be obtained if the interconnected, complex nature of the system under investigation is not belittled. In fact, by explicitly facing this complexity, problems can be tackled by tailored, efficient strategies. A huge number of elements in the system (large-scale) exchange a tremendous amount of information (big-data) by means of peer-to-peer interactions with nearby elements (networked communication). In the last two decades, several research communities have modeled CPNs as "distributed systems": a set of agents (elements of a CPN) cooperates by exchanging their local variables (data information) with only a small group of neighbors in a communication graph (local interactions). With this model at hand, several challenges arising in CPNs as, e.g., estimation in sensor networks, learning in machine learning applications, control of autonomous networked dynamical system, can be stated as optimization problems over networks. A special feature of such optimization problems is that they naturally encode the networked structure in their definition, giving rise to a novel, growing research branch termed *distributed optimization*.

Formally, a common set-up in distributed optimization consists of a network of processors that is asked to solve a *cost-coupled* optimization problem in which each processor is assigned an objective function and (possibly) a local constraint set. The global cost function to minimize is the sum of the local objective functions, and the problem constraint set is the intersection of the local constraints. Following pioneering ideas in [98], the first class of algorithms in peer-to-peer distributed optimization were based on (sub)gradient methods [64, 65]. The proposed algorithms fuse consensus schemes with descent-based iterations. Concurrently, subgradient methods were proposed

in combination with duality theory. Exploiting the separable structure of the cost function and the sparsity of the graph, peer-to-peer versions of primal-dual decomposition methods were proposed in [31, 43], see [103] for a tutorial on peer-to-peer optimization via dual decomposition. Later, these methods were extended to optimization problems including global coupling constraints [25, 106]. In order to induce robustness in the computation and improve convergence in the case of non-strictly convex functions, Alternating Direction Methods of Multipliers (ADMM) have been proposed in the network context. In particular, the standard Lagrangian function is augmented with a quadratic penalty function for linear constraints. This quadratic extension makes ADMM flexible and applicable to most convex optimization problems. A first distributed ADMM algorithm was proposed in [87] in the context of distributed estimation, while a survey on this technique is given in [17].

However, all these methods have focused on a limited distributed optimization setting: *convex cost-coupled* optimization problems, typically *scalar* or *low-dimensional* decision variable, and networks with *synchronous* communication. Thus, novel important challenges need to be faced and have been subject of investigation of this thesis. In peer-to-peer systems the communication network is given and cannot be considered a design parameter as, e.g., in cluster computers. Thus, the algorithms need to take into account more complex schemes that can handle, e.g., asynchronous updates. A second challenge is related to the problem size. In fact, considering all the local information available at each agent in the network leads immediately to "big-data" optimization problems which may be nonconvex. The huge problem dimension and nonconvexity are difficult to handle with standard approaches and call for novel efficient strategies. Finally, it also important to focus on a key feature of distributed optimization algorithms when applied to dynamical networked systems where, usually, it is needed to repeatedly solve optimization problems with coupling among agents, thus having some local information on the global feasibility.

Contributions and Organization

This thesis contributes to the distributed optimization literature by proposing novel algorithms addressing the challenges above, which arise in concrete applications. Specifically, we consider set-ups that have been well-studied in the literature and propose some extensions of existing schemes in order to cope with asynchronous communications. Also, we address new big-data and constrained coupled optimization frameworks and propose novel distributed optimization approaches.

We start by detailing the main contributions of the thesis and then provide the detailed contributions of each chapter.

Summary of Main Contributions

First, we design asynchronous distributed algorithms based on a symmetric event-triggered communication protocol. In this communication set-up, a node is in idle mode until its local timer triggers. When in idle, it continuously collects messages from neighboring nodes that are awake and, if needed, runs some auxiliary computations. When the local timer triggers, it updates local variables and broadcasts them to neighboring nodes. Under mild assumptions on the local timers, the local awakening process results into a uniform random choice of one active node per iteration. Using this property we are able to prove that the distributed algorithms correspond to proper block-coordinate methods. As a result, the distributed algorithms inherit the convergence properties of the centralized schemes. In this thesis we propose asynchronous distributed algorithms for two problem set-ups, namely composite cost-coupled optimization and partitioned optimization. A key feature of our asynchronous schemes is that the step-sizes of the gradient iterations are local and constant, so that it can be chosen independently by each node. We propose an extension of distributed dual decomposition for asynchronous networks in which we are able to handle local constraints and local regularizers. Moreover, we also adopt this asynchronous protocol to design a distributed algorithm to solve nonconvex partitioned problems tackling directly the primal problem formulation.

Second, we take into account "big-data" problems in which the dimension of the decision variable is huge. This distinctive feature represents a twofold challenge in a distributed context: (i) local optimization steps on the entire decision variable cannot be performed at each node since they would be too computationally demanding, and (ii) the communication cost of transmitting a copy of the entire solution estimate would be too expensive. We start by considering several concrete applications in which the dimension of the optimization variable depends on the number of agents, but the nodes are interested in computing only (their own) part of the entire decision vector. These application set-ups can be cast into structured optimization problems in which cost functions and constraints have a partitioned structure, i.e., they depend only on a (small) portion of the (huge) decision vector. This set-up calls for tailored methods that explicitly rely on the partitioned structure. We adopt the asynchronous protocol based on timers also for the partitioned set-up and design both a dual and a primal method. These alternative approaches allow us to derive distributed optimization algorithms that can handle convex programs with local constraints and nonnconvex problems respectively. Second, we consider a more general big-data optimization set-up where the cost is the sum of local (nonconvex) functions depending on a common decision variable which might be huge. Moreover, a (common) convex regularizer and a (common) convex constraint is present. We propose an iterative two-step procedure where

each agent maintains a local estimate of the entire decision variable but, at every iteration, updates and communicates only *one block*. Agents select their block in an uncoordinated fashion by means of an "essentially cyclic rule", thus guaranteeing that all blocks are persistently updated during the algorithmic evolution. Moreover, agents employ a running averaging scheme in order to enforce consensus of the local solution estimates and track the gradient of the global cost function. This running consensus is again implemented in a block-wise fashion, thus extending the existing consensus literature to the distributed big-data scenario.

Finally, we consider problem set-ups typically arising in dynamic optimization contexts such as robotic networks or cooperative estimation. A set of dynamical systems needs to repeatedly solve optimization problems while taking into account explicitly (i) the networked structure of the system and (ii) its dynamical evolution. We formalize this problem as a constraint-coupled optimization problem in which agents in a network need to minimize a local performance index (depending only on a local decision variable) subject to a constraint coupling all the local variables. Such problem set-up represents a first methodological step to study complex dynamical networked systems that need to perform optimization tasks over networks as in, e.g., distributed model predictive control, which is an optimization-based control technique for tackling distributed control problems.

Organization and Chapter Contributions

The organization of the thesis reflects the three major contributions and in the following we detail how each chapter answers the challenges addressed in this thesis.

Chapter 1 introduces the distributed optimization framework by describing the main features of a distributed algorithm, and proposing a classification of the problem set-ups investigated in this thesis, namely cost-coupled and constraint-coupled optimization, along with some motivating concrete examples.

In Chapter 2 we propose a class of distributed optimization algorithms based on proximal gradient methods and duality, to solve optimization problems in which the (possibly nonsmooth) cost function is separable (i.e., the sum of possibly nonsmooth functions sharing a common variable) and can be split into a strongly convex term and a convex one. For a fixed graph topology, we develop a distributed optimization algorithm (based on a centralized dual proximal gradient idea introduced in [6]) to minimize a separable strongly convex cost function. The proposed distributed algorithm is based on a proper choice of primal constraints (suitably separating the graph-induced and node-local constraints), that gives rise to a nice dual problem formulation, namely it has a separable structure when expressed in terms of local conjugate functions. Thus, a proximal gradient applied to

such a dual problem turns out to be a distributed algorithm where each node updates: (i) its primal variable through a local minimization and (ii) its dual variables through a suitable local proximal gradient step. The algorithm inherits the convergence properties of the centralized one and exhibits $O(1/t)$ rate of convergence in objective value. We point out that the algorithm can be accelerated through a Nesterov's scheme obtaining an $O(1/t^2)$ convergence rate. Second, as main contribution for the mentioned asynchronous challenge, we propose an asynchronous algorithm for a symmetric *event-triggered* communication protocol. In this communication set-up, a node is in idle mode until its local timer triggers. When in idle, it continuously collects messages from neighboring nodes that are awake and, if needed, updates a primal variable. When the local timer triggers, it updates local primal and dual variables and broadcasts them to neighboring nodes. Under mild assumptions on the local timers, the whole algorithm results into a uniform random choice of one active node per iteration. Using this property and showing that the dual variables can be stacked into separate blocks, we are able to prove that the distributed algorithm corresponds to a block-coordinate proximal gradient, as the one proposed in [85], performed on the dual problem. Specifically, we are able to show that the dual variables handled by a single node represent a single block-variable, and the local update at each triggered node turns out to be a block-coordinate proximal gradient step (in which each node has its own local step-size). The result is that the algorithm inherits the convergence properties of the block-coordinate proximal gradient. An important property of the distributed algorithms presented in this chapter is that they can solve fairly general optimization problems including both composite cost functions and local constraints. A key distinctive feature of the algorithm analysis is the combination of duality theory, coordinate-descent methods, and properties of the proximal operator when applied to conjugate functions. The results of this chapter are based on [74, 75].

In Chapter 3 we investigate partitioned big-data optimization problems by using a dual and a primal approach, covered in the first and in the second part of the chapter, respectively. As for the dual approach, we provide two distributed optimization algorithms based on dual decomposition, with two main distinctive features. First, the algorithms are scalable, in the sense that each node only processes a portion of the decision variable vector. As a result, the information stored and the computation performed by each node does not depend on the network size as long as the node degree is bounded. Second, we design an asynchronous algorithm that works under a communication protocol in which a node wakes-up when triggered by its local timer or by its neighbors, so that no global clock is needed. The distributed algorithms are derived by first writing a suitable equivalent formulation of the original primal optimization problem (which exploits the partitioned struc-

ture). Then its dual problem is derived and solved with suitable algorithms. A scaled gradient applied to the dual problem turns out to be a partitioned version of the distributed dual decomposition (synchronous) algorithm. A randomized ascent method applied to the dual problem allows us to write an asynchronous distributed algorithm that converges in objective value with high probability. As for the primal approach, we propose an asynchronous, distributed algorithm to solve partitioned, big-data nonconvex optimization problems. The proposed algorithm is based on local updates involving the minimization of a strongly convex, quadratic approximation of the objective function. Each node constructs this approximation by exchanging information only with its neighboring nodes. The updates at each node are regulated by a local timer that triggers independently from the ones of the other nodes. We prove the convergence in probability of the distributed algorithm by showing that it is equivalent to a generalized coordinate descent method for the minimization of nonconvex composite functions. The generalized coordinate descent algorithm extends the one proposed in the literature and, thus, represents an interesting side result. The results of this chapter are based on [68, 69].

In Chapter 4 we consider the constrained minimization of the sum of a smooth (possibly) nonconvex function, i.e., the agents' sum-utility, plus a convex (possibly) nonsmooth regularizer. Our interest in this chapter is on big-data problems where there is a large number of variables to optimize. We propose a distributed algorithm to solve (possibly nonconvex) big-data optimization problems over peer-to-peer networks modeled as directed graphs. As opposed to standard optimization, when it comes to big-data problems, local computation and communication requirements need to be explicitly taken into account in the algorithmic design. Specifically, a new twofold challenge arises at each node: (i) local optimization steps on the entire decision variable cannot be performed since they would be too computationally demanding, and (ii) the communication cost of transmitting a copy of the entire solution estimate would be too expensive. To cope with these issues, we propose an iterative two-step procedure, inspired to an existing distributed scheme [92, 93], where each agent maintains a local estimate of the entire decision variable but, at every iteration, updates and communicates only *one block*. Agents select their block in an uncoordinated fashion by means of an "essentially cyclic rule" guaranteeing that all blocks are persistently updated during the algorithmic evolution. Specifically, each agent minimizes a strongly-convex surrogate function with respect to the selected block variable only. The surrogate is based on the agent local cost-function and a local gradient estimate (of the smooth global-cost portion). The optimization step is combined with a *block-wise* consensus step tracking the network average gradient and guaranteeing the asymptotic agreement of the local solution estimates to a common stationary solution of the nonconvex

problem. It is worth noting that the aforementioned new block-wise consensus scheme is of independent interest and represents a contribution per se: it provides a new distributed algorithm to compute the average of big-data vectors over time-varying directed graphs, when each agent can send to neighbors only one block per iteration. The results of this chapter are based on [76, 77].

In Chapter 5, we investigate a problem set-up arising in dynamic optimization scenarios as, e.g., in distributed model predictive control. We propose a novel, distributed optimization algorithm to solve constraint-coupled optimization problems over networks. It is a two-step procedure in which each agent iteratively computes a primal-dual optimal solution pair of a local optimization problem and then updates proper dual variables until convergence. In the proposed protocol, agents exchange only suitable dual variables. They do not need to communicate, and thus disclose, their estimates of local decision variable, cost and constraints. Nevertheless, each agent is guaranteed to asymptotically converge to its portion of an optimal (and, thus, feasible) solution of the original problem. This feature, known in the literature as primal recovery, is obtained for convex costs and without resorting to any running averaging mechanism, but results from the methodology we employed. Despite its clean and simple structure, the proposed distributed algorithm is the result of a sequence of duality steps combined with a proper relaxation in order to guarantee feasibility of the local algorithmic steps at each iteration. Specifically, a relaxation of the original, primal problem is first introduced. This approach is theoretically motivated by a proper restriction of the dual problem and leads to an equivalent primal problem which typically arises in penalty methods. Then, a *tour* through duality is performed by deriving a sequence of equivalent optimization problems. As a final step, a subgradient method applied to the last equivalent problem is reformulated (using again duality) in terms of computations on the primal variables. We also show that without the proposed relaxation approach, the overall algorithmic idea may be formally applied, but several feasibility issues would arise, thus preventing its concrete distributed implementability. Overall, the proposed distributed algorithm enjoys three appealing features: (i) local computations at each node involve only the local decision variable and, thus, scale nicely with respect to the dimension of the decision vector, (ii) privacy is preserved since only dual variables are exchanged among agents, and (iii) an estimate of a primal optimal solution component is computed by each agent without any averaging mechanisms usually needed in dual algorithms. The results of this chapter are based on [70–73].

Complementing the main chapters of the thesis, some preliminaries are reviewed in the appendix.

Chapter 1

Distributed Optimization Framework

In this chapter we introduce the conceptual framework regarding distributed optimization algorithms in peer-to-peer networks. First, we describe a general model of communication network, then we present and motivate the problem set-ups that will be investigated in this thesis.

In a distributed scenario, we consider N units, called *agents* or *processors*, that have both communication and computation capabilities. The communication among agents is modeled by means of graph theory. We consider a graph $\mathcal{G}$ connecting the N units and we say that an agent i can send data to another agent j, called neighbor, when an edge connecting i to j is present in the graph $\mathcal{G}$. Based on the information exchange, agents can update their local states by exploiting their local computational power. In a distributed algorithm, agents initialize their local states to some values and then start an iterative procedure in which communication and computation steps are iteratively performed according to a given scheme.

We are interested in a distributed scenario in which agents cooperatively solve an optimization problem, e.g., they want to agree on a common optimal solution of the problem or, alternatively, each agent wants to compute only its local portion of interest of an optimal solution. The basic assumption we make is that each agent i has only a partial knowledge of the entire problem (which is spread over the network), e.g., only a part of the cost and/or a part of the constraints is locally available.

1.1 Network and Communication Models

In this section we formally define an abstract network model for a distributed algorithm. A communication network is a (possibly time-dependent) directed graph $\mathcal{G}^t = (\{1, \dots, N\}, \mathcal{E}^t)$, where $t \in \mathbb{N}$ is the universal (slotted) time, $\{1, \dots, N\}$ is the (fixed) set of agents and $\mathcal{E}^t \subseteq \{1, \dots, N\} \times \{1, \dots, N\}$, for all $t \in \mathbb{N}$, is the (time-dependent) set of (directed) edges over the vertices

$\{1, \dots, N\}$, called the communication links. A graphical representation of a time-varying network is given in Figure 1.1.

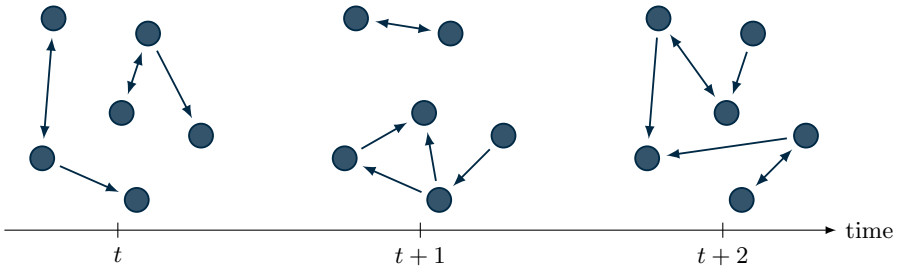


Figure 1.1: A directed time-varying graph of $N = 6$ nodes.

The universal time t determines which communication structure, i.e., which graph $\mathcal{G}^t$, is currently active. The time-varying edge set $\mathcal{E}^t$ models the communication in the sense that at time t there is an edge from node i to node j in $\mathcal{E}^t$ if and only if processor i transmits information to processor j at time t. Given an edge $(i, j) \in \mathcal{E}^t$, i is called *in-neighbor* of j and j is an *out-neighbor* of i at time t. When the edge set $\mathcal{E}^t$ does not depend on t, i.e., $\mathcal{G}^t \equiv \mathcal{G}$ for all t, we say that the network is fixed, otherwise the network is time-varying. Moreover, when for every pair i and j in the network the edge (i, j) and the edge (j, i) are in $\mathcal{E}^t$, then the graph is undirected. An example of a directed and of an undirected graph is depicted in Figure 1.2.

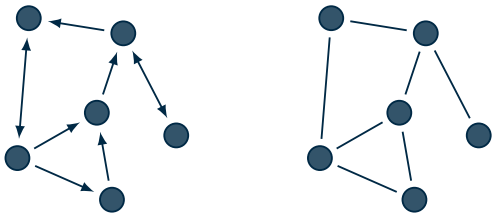


Figure 1.2: A directed (right) and an undirected (left) graph of $N = 6$ nodes.

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. An undirected graph is said to be connected when for every pair of nodes (i, j) there exists a path of edges that goes from i to j. Connectivity properties can be also stated for time-varying topologies. A sequence of time-varying graphs $\{\mathcal{G}^t\}_{t\geq 0}$ is said to be T-strongly connected if there exists a scalar $T > 0$ such that the graph $\mathcal{G}_T^t(\{1, \dots, N\}, \mathcal{E}_T^t)$ with $\mathcal{E}_T^t = \bigcup_{\tau=0}^{T-1} \mathcal{E}^{t+\tau}$, is strongly connected for every $t \geq 0$.

Given a network topology, agents can perform their algorithms according to several communication protocols. In this thesis we will focus on two specific examples. When the steps of the distributed algorithm explicitly

depend on 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 temporal information, i.e., their updates do not depend on t, and we term these algorithms *asynchronous*.

In general, it is not necessary that all processors know the value of the universal time t. In Chapter 2 and Chapter 3 we will consider an asynchronous protocol where each node i has its own concept of time defined by a local timer, which randomly and independently of the other nodes triggers when to awake itself as depicted in Figure 1.3.

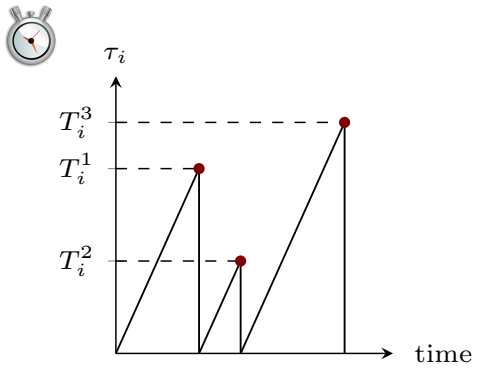


Figure 1.3: Triggering process of timer τ_i; the red circles are the triggering events.

Given a fixed communication network $\mathcal{G}$, each agent i has two modes. Between two triggering events, the node is in an *idle* mode, i.e., it continuously receives messages from its neighbors in $\mathcal{G}$. When a trigger occurs, it switches into an *awake* mode in which it updates its local variables and transmits the updated information to its neighbors.

Formally, the triggering process is modeled by means of a local clock $\tau_i \in \mathbb{R}_{\geq 0}$ and a randomly generated waiting time T_i. As long as $\tau_i < T_i$ the node is in idle mode. When $\tau_i = T_i$ the node switches to the awake mode and, after running the local computation, resets $\tau_i = 0$ and draws a new realization of the random variable T_i. The local waiting times T_i can be used to model that a non-negligible amount of time is needed to perform the local computation. From an external, global perspective the local awakening process results in a proper random process of selection of agents. This mathematical equivalence will be used to analyze the algorithms.

In this thesis, we will design algorithms which work under various network topologies that will be specified in each chapter.

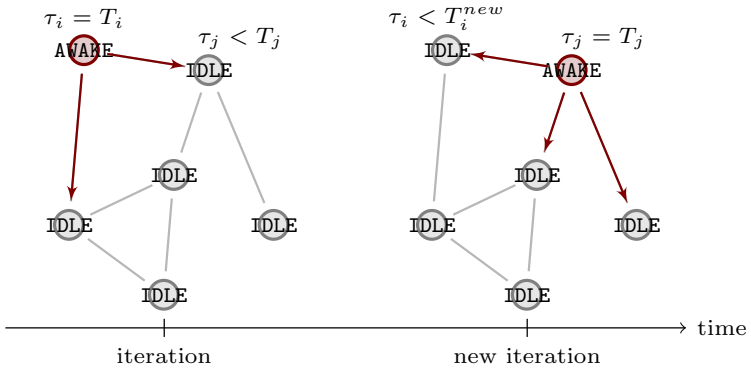


Figure 1.4: An example of awakening process due to the local timers. On the left, agent i is triggered by its timer τ_i, performs its local computation and extracts a new value T_i^{new} and becomes idle. Then τ_j triggers a new iteration of the algorithm.

1.2 Cost-Coupled Optimization

In this section we introduce a common optimization set-up that has been widely investigated in a distributed context. The cost function is expressed as the sum of local contributions depending on a common optimization variable. Formally, the optimization problem is

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^d} \sum_{i=1}^{N} f_i(\mathbf{x}) \\ &\text{subj. to } \mathbf{x} \in X \subseteq \mathbb{R}^d \end{aligned} \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is known only to agent i, for all $i \in \{1, \dots, N\}$.

Next, we discuss how this set-up can be adapted to fit into specific scenarios.

1.2.1 Composite Optimization

Here, we describe a special instance of problem (1.1) where the cost function splits in two terms. We start by a motivating example and, then, we formally state the problem.

Let us consider a separable optimization problem with local constraints

in the form

$$\begin{aligned} &\min_{\mathbf{x}} \ \sum_{i=1}^{N} h_i(\mathbf{x}) \\ &\text{subj. to } \mathbf{x} \in \bigcap_{i=1}^{N} X_i \subseteq \mathbb{R}^d \end{aligned} \tag{1.2}$$

where each h_i is a local function known by agent i and each X_i is a convex set. By using indicator functions I_{X_i} associated to each X_i [1], we can recast problem (1.2) by transforming the constraints into additional terms in the objective function, obtaining

$$\min_{\mathbf{x}\in\mathbb{R}^d} \ \sum_{i=1}^{N} \Big(h_i(\mathbf{x}) + I_{X_i}(\mathbf{x}) \Big).$$

Since each X_i is a convex set, then I_{X_i} is a convex function.

We can generalize the discussion above by considering cost functions that split in two terms. Moreover, by defining these terms as extended real-valued convex functions without smoothness requirements, additional local constraints can be considered in the problem. This gives further flexibility to this set-up as will be clear from the examples in the next section. Formally, the general composite problem is

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^d} \ \sum_{i=1}^{N} \Big(f_i(\mathbf{x}) + g_i(\mathbf{x}) \Big), \\ &\text{subj. to } \mathbf{x} \in X_i, \qquad\qquad i \in \{1,\dots,N\}, \end{aligned} \tag{1.3}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, $g_i : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ and $X_i \subseteq \mathbb{R}^d$ is known only to agent i, for all $i \in \{1,\dots,N\}$.

Notice that here we are not specifying any regularity property on problem (1.3). In fact, we will consider (possibly nonsmooth) strongly convex functions in Chapter 2 and in the first part of Chapter 3, while we will consider smooth nonconvex costs f_i in the second part of Chapter 3 and in Chapter 4.

1.2.2 Partitioned Big-Data Optimization

In this subsection, we start by an observation. In several concrete applications the dimension of the optimization variable depends on the number of agents. Moreover, the nodes are interested in computing only a (small) portion of the entire decision vector, namely only some local variables of

[1] Given a closed, nonempty convex set X, the indicator function of $X \subseteq \mathbb{R}^d$ is defined as $I_X(\mathbf{x}) = 0$ if $\mathbf{x} \in X$ and $I_X(\mathbf{x}) = +\infty$ otherwise.

interest. We model this scenario by letting the decision vector $\mathbf{x} \in \mathbb{R}^d$ be partitioned as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}$$

where each $\mathbf{x}_i \in \mathbb{R}^{n_i}$ with $n_i \in \mathbb{N}$ for all $i \in \{1, \dots, N\}$, and $\sum_{i=1}^N n_i = d$. The sub-vector $\mathbf{x}_i$ represents the relevant information at node i, also referred to as the state of node i. Additionally, let us consider local objective functions and constraints that have the same sparsity as the communication graph, namely, for $i \in \{1, \dots, N\}$, the cost function f_i and the constraint X_i depend only on the state of node i and on its neighbors, that is, on $\{\mathbf{x}_j,\, j \in \mathcal{N}_i \cup \{i\}\}$. Then the problem we aim at solving distributedly is

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_{i=1}^N f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \\ \text{subj. to } & (\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \in X_i, \qquad i \in \{1, \dots, N\}, \end{aligned} \tag{1.4}$$

where the notation $f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})$ means that $f_i : \mathbb{R}^d \to \mathbb{R}$ is, in fact, a function of $\mathbf{x}_i$ and $\mathbf{x}_j$, $j \in \mathcal{N}_i$, and the notation $(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \in X_i$ means that the constraint set X_i involves only the variables $\mathbf{x}_i$ and $\mathbf{x}_j$, $j \in \mathcal{N}_i$.

Remark 1.2.1. *Notice that here the subscript index denotes a component of a decision variable, e.g., $\mathbf{x}_i$ is the i-th component of $\mathbf{x}$. In some parts $\mathbf{x}_i$ might be the copy of $\mathbf{x}$ maintained by agent i. We will clarify, whenever needed, the exact meaning of the subscript during the dissertation to avoid confusion.* □

1.2.3 General Cost-Coupled Big-Data Optimization

Another way to model the big-data optimization set-up consists in considering a generic cost-coupled optimization problem, without any assumption on its partitioned structure. Since the problem is big-data, we explicitly assume that the optimization variable $\mathbf{x}$ has a huge size and, thus, we consider it as a stack of smaller *blocks* given by

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_B \end{bmatrix},$$

with each block $\mathbf{x}_\ell \in \mathbb{R}^d$ for all $\ell \in \{1, \dots, B\}$. Notice that B may also depend on N as in the partitioned set-up.

Then, we consider the following composite optimization problem

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{dB}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}) + \sum_{\ell=1}^{B} r_\ell(\mathbf{x}_\ell) \\ \text{subj. to } & \mathbf{x}_\ell \in \mathcal{K}_\ell, \quad \ell \in \{1,\dots,B\}, \end{aligned} \tag{1.5}$$

where $\mathbf{x}$ is the vector of optimization variables (partitioned in B blocks), each $f_i : \mathbb{R}^{dB} \to \mathbb{R}$ represents the cost function of agent i, $r_\ell : \mathbb{R}^d \to \mathbb{R}$, $\ell \in \{1,\dots,B\}$, is a regularizer; and $\mathcal{K}_\ell$, $\ell \in \{1,\dots,B\}$, is the constraint set. Usually the regularizer term in (1.5) is nonsmooth and is used to promote some extra structure in the solution, such as (group) sparsity.

1.3 Examples of Cost-Coupled Optimization

In this section we provide motivating scenarios for the cost-coupled set-up.

1.3.1 Regularized Constrained Optimization

As mentioned in Section 1.2.1 the algorithmic framework in (1.3) is flexible and allows us to handle, together with local constraints, also a regularization cost through the g_i. Regularizing the solution is a useful technique in many applications such as sparse design, robust estimation in statistics, support vector machine (SVM) in machine learning, total variation reconstruction in signal processing and geophysics, and compressed sensing. In these problems, the cost f_i is a loss function representing how the predictions based on a theoretical model mismatch the real data. Next, we focus on the most widespread choice for the loss function, which is the least square cost, giving rise to the following optimization problem

$$\min_{\mathbf{x}} \ \sum_{i=1}^{N} \|\mathbf{D}_i\mathbf{x} - \mathbf{b}_i\|^2 \tag{1.6}$$

where $\mathbf{D}_i$ are data/regressors and $\mathbf{b}_i$ are labels/observations.

A typical challenge arising in regression problems is due to the fact that problem (1.6) is often ill-posed and standard algorithms easily incur in over-fitting phenomena. A viable technique to prevent over-fitting consists of adding a regularization term/cost and write

$$\min_{\mathbf{x}} \ \sum_{i=1}^{N} \|\mathbf{D}_i\mathbf{x} - \mathbf{b}_i\|^2 + g(\mathbf{x}).$$

Usual choices for g are the ℓ_2-norm, also referred as Tikhonov regularization or ridge regression. Alternatively, also the ℓ_1-norm can be used

and leads to the so-called LASSO (Least Absolute Shrinkage and Selection Operator) problem, i.e.,

$$\min_{\mathbf{x}} \sum_{i=1}^{N} \|\mathbf{D}_i\mathbf{x} - \mathbf{b}_i\|^2 + \lambda\|\mathbf{x}\|_1$$

where λ is a positive scalar which is used to strengthen or weaken the effects of the regularizer.

In some cases (as, e.g., in distributed estimation [44]) one may be interested in having the solution bounded in a given box or leaving in a reduced subspace. This gives rise to the so-called *constrained LASSO* problem (see, e.g., [42, 58, 101])

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i=1}^{N} \|\mathbf{D}_i\mathbf{x} - \mathbf{b}_i\|^2 + \lambda\|\mathbf{x}\|_1 \\ \text{subj. to} \quad & \mathtt{lb} \preceq \mathbf{x} \preceq \mathtt{ub}. \end{aligned}$$

Notice that the regularizer is usually added to enforce sparsity patterns, in the solution of the problem. The most natural choice would be the 0-norm which counts the nonzero entries of a vector. Unfortunately this function is nonconvex, thus researchers have usually employed the 1-norm or the 2-norm which are good convex surrogate of the 0-norm. However other nonconvex surrogates can be used to improve the sparsity pattern, but this would lead to a nonconvex optimization problem. In Chapter 4 we address this class of problems and, in Section 4.6, a numerical example of nonconvex regression can be found.

1.3.2 Partitioned Estimation in Power Networks

Next we consider an application scenario in which the partitioned structure described in 1.2.2 arises naturally. To describe this example we follow the dissertation in [82]. For a power network, the state in a certain time instant consists of the voltage angles and magnitudes at all the system buses. The (static) state estimation problem refers to the procedure of estimating the state of a power network given a set of measurements of the network variables, such as, for instance, voltages, currents, and power flows along the transmission lines. Formally, let $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{z} \in \mathbb{R}^P$ be, respectively, the state and measurements vector. Then, the vectors $\mathbf{x}$ and $\mathbf{z}$ are related by

$$\mathbf{z} = h(\mathbf{x}) + \boldsymbol{\eta}, \tag{1.7}$$

where $h(\cdot)$ is a nonlinear measurement function, and where $\boldsymbol{\eta}$ is the noise measurement, which is traditionally assumed to be a zero mean random vector satisfying $\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^\top] = \boldsymbol{\Sigma} \succ \mathbf{0}$. An optimal estimate of the network state

coincides with the most likely vector $\mathbf{x}^\star$ that solves equation (1.7). This static state estimation problem can be simplified by adopting the approximate estimation model presented in [88], which follows from the linearization around the origin of equation (1.7). Specifically,

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v},$$

where $\mathbf{H} \in \mathbb{R}^{P\times d}$ and where $\mathbf{v}$, the noise measurement, is such that $\mathbb{E}[\mathbf{v}] = \mathbf{0}$ and $\mathbb{E}[\mathbf{v}\mathbf{v}^\top] = \mathbf{\Sigma}$. In this context the static state estimation problem is formulated as the following weighted least-square problem

$$\min_{\mathbf{x}} \ (\mathbf{z} - \mathbf{H}\mathbf{x})^\top \mathbf{\Sigma}^{-1}(\mathbf{z} - \mathbf{H}\mathbf{x}). \tag{1.8}$$

Assume $\ker(\mathbf{H}) = \emptyset$, then the optimal solution of problem (1.8) is given by

$$\mathbf{x}_{\text{wls}} = \left(\mathbf{H}^\top \mathbf{\Sigma}^{-1}\mathbf{H}\right)^{-1} \mathbf{H}^\top \mathbf{\Sigma}^{-1}\mathbf{z}.$$

For simplicity let us assume that $\mathbf{\Sigma} = \mathbf{I}$. For a large power network, the centralized computation of $\mathbf{x}_{\text{wls}}$ might be too costly. A possible solution to address this complexity problem is to distribute the computation of $\mathbf{x}_{\text{wls}}$ among geographically deployed control centers (monitors), say N in a way that each monitor is responsible only for a subpart of the whole network. Precisely let the matrices $\mathbf{H}$ and $\mathbf{\Sigma}$ and the vector $\mathbf{z}$ be partitioned as $[H_{ij}]_{i,j=1}^N$, $\mathbf{x} = \left[\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top\right]^\top$ and $\mathbf{z} = \left[\mathbf{z}_1^\top, \ldots, \mathbf{z}_N^\top\right]^\top$, where $\mathbf{H}_{ij} \in \mathbb{R}^{p_i\times n_j}$, $\mathbf{z}_i \in \mathbb{R}^{p_i}$, $\mathbf{x}_i \in \mathbb{R}^{n_i}$ and $\sum_{i=1}^N n_i = d$, $\sum_{i=1}^N p_i = P$. Observe that, because of the interconnection structure of a power network, the measurement matrix $\mathbf{H}$ is usually sparse, i.e., $\mathbf{H}_{ij} = \mathbf{0}$ for many indices (i, j). Assume monitor i knows $\mathbf{z}_i$ and $\mathbf{H}_{ij}$, $j \in \{1, \ldots, N\}$ and that it is interested only in estimating the sub-state $\mathbf{x}_i$. Moreover let $\mathcal{N}_i = \{j \in \{1, \ldots, N\} \mid \mathbf{H}_{ij} \neq \mathbf{0}\}$. Observe that in general if $\mathbf{H}_{ij} \neq \mathbf{0}$ then also $\mathbf{H}_{ji} \neq \mathbf{0}$. Then by defining

$$f_i\big(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}\big) = \Big(\mathbf{z}_i - \sum_{j\in\mathcal{N}_i} \mathbf{H}_{ij}\mathbf{x}_j\Big)^\top \Big(\mathbf{z}_i - \sum_{j\in\mathcal{N}_i} \mathbf{H}_{ij}\mathbf{x}_j\Big),$$

problem (1.8) can be equivalently rewritten as an unconstrained version of (1.4).

It is worth remarking that there are other significant examples that can be cast as distributed weighted least square problems similarly to the static state estimation in power networks we described in this subsection; see, for instance, distributed localization in sensor networks and map building in robotic networks.

1.3.3 Network Utility Maximization and Resource Allocation

We consider the flow optimization problem, or *Network Utility Maximization (NUM)* problem introduced in [45] and studied in [51] in a distributed

context. A flow network (which is different from a communication network) consists of a set L of unidirectional links with capacities c_ℓ, $\ell \in L$. The network is shared by a set of N sources. Each source has a strongly concave utility function $U_i(x_i)$ in the scalar rate x_i. The goal is to calculate source rates that maximize the sum of the utilities $\sum_{i=1}^N U_i(x_i)$ over x_i subject to capacity constraints. Formally, using a notation consistent with [51], let $L(i) \subseteq L$ be the set of links used by source i and $N(\ell) = \{i \in \{1, \dots, N\} \mid \ell \in L(i)\}$ be the set of sources that use link ℓ. Note that $\ell \in L(i)$ if and only if $i \in N(\ell)$. Also, let $I_i = [\kappa_i, K_i]$, with $0 \leq \kappa_i < K_i$, be the interval of transmission rates allowed to node i. The network flow optimization problem is given by

$$\begin{aligned} \max_{x_1,\dots,x_N} \quad & \sum_{i=1}^{N} U_i(x_i) \\ \text{subj. to} \quad & x_i \in I_i, && i \in \{1, \dots, N\}, \\ & \sum_{j \in N(\ell)} x_j \leq c_\ell, && \ell \in \{1, \dots, |L|\}. \end{aligned} \tag{1.9}$$

Notice that problem (1.9) is well posed and has compact domain.

In Figure 1.5 (left) we graphically represent an example of 5 sources (filled circles) that use (dotted arrows) 3 links (gray stripes). In [51] a distributed optimization algorithm is proposed in which both the sources and the links are computation units. Here we consider a set-up in which only the sources are computation units. In particular, sources have the computation and communication capabilities introduced in the previous subsection. We assume that sources using the same links can communicate and both know the capacity constraint on those links. Formally, we introduce a graph $\mathcal{G}$ having an edge (i, j) connecting source i to j if and only if there exists $\ell \in L$ such that $\ell \in L(i) \cap L(j)$. In Figure 1.5 (right) we show the induced communication graph (solid lines) for the considered example.

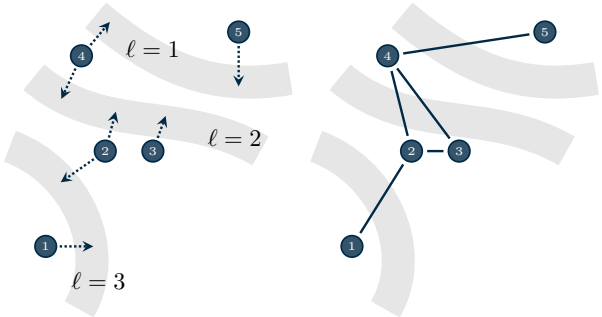


Figure 1.5: Network Utility Maximization problem with 5 sources (filled circles) using 3 links (gray stripes).

Thus, optimization problem (1.9) can be rewritten as

$$\begin{aligned} \max_{x_1,\dots,x_N} \ & \sum_{i=1}^{N} U_i(x_i) \\ \text{subj. to } \ & (x_i, \{x_j\}_{j\in\mathcal{N}_i}) \in \prod_{j\in\mathcal{N}_i\cup\{i\}} I_j,\ i\in\{1,\dots,N\}, \\ & \sum_{j\in\mathcal{N}_i\cup\{i\}} a_{j,\ell}\, x_j \le c_\ell,\ \ \ell\in L(i),\ i\in\{1,\dots,N\}, \end{aligned} \tag{1.10}$$

where $\mathcal{N}_i$ is the neighbor set of i in $\mathcal{G}$, $a_{j,\ell}$ is 1 if agent j can use link ℓ and 0 otherwise. Notice that in problem (1.10) if sources i and j share a link ℓ, then they both have the capacity constraint of link ℓ. Moreover, in order to have compactness of the local constraint set X_i, transmission rate constraints of neighboring nodes are also taken into account. Finally, in order to fulfill the strong convexity assumption on the local costs, two strategies can be used. First, one can assume that each agent knows the utility functions of neighboring agents, so that it sets $f_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) = \sum_{j\in\mathcal{N}_i\cup\{i\}} U_j(x_j)$. Alternatively, one can consider an additional separable (small) regularization term in the NUM problem formulation in (1.9), e.g., $\epsilon\sum_{i=1}^{N} x_i^2$ with $\epsilon > 0$. In this case each agent sets its local cost function to $f_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) = U_i(x_i) + \sum_{j\in\mathcal{N}_i} \epsilon_{ij} x_j^2$, where $\epsilon_{ij} > 0$ are suitable fractions of ϵ. Except for the maximization versus minimization, this problem is partitioned, that is it has the same structure as (1.4).

A problem with this structure can be also found in resource allocation problems, which are of great importance in several research areas. In the context of network systems solving resource allocation problems in a distributed way is a preliminary task to solve several control and estimation problems. Indeed, it is often the case that the agents in the network have some local resource that has to be shared with their neighbors.

Consider a general set-up in which each agent produces a certain amount of resources, which it can share with its neighbors (i.e., neighboring nodes in the communication graph). Each agent has a local strongly concave utility function to maximize. The resulting optimization problem turns out to be

$$\begin{aligned} \max_{x_1,\dots,x_N} \ & \sum_{i=1}^{N} U_i(x_i) \\ \text{subj. to } \ & \sum_{j\in\mathcal{N}_i\cup\{i\}} x_j \le r_i, \qquad i\in\{1,\dots,N\}, \end{aligned}$$

where x_i is the resource produced by node i, r_i is the capacity of node i and we are assuming that the set of neighbors with whom node i can share its resource coincides with the set of neighbors in the communication graph. In other words, agents can share resources only if they can communicate.

1.4 Constraint-Coupled Optimization

In this section, we present another important large-scale optimization set-up amenable to distributed computation. Here the goal is the minimization of the sum of local cost functions, each one depending on a *local* variable, subject to a local constraint for each variable and a coupling constraint involving all the decision variables. This *constraint-coupled* optimization set-up arises in several scenarios of interest in Control and Robotics as well as Communication and Signal Processing. Example set-ups include resource allocation (e.g., in Communication or Cooperative Robotics) or network flow optimization (e.g., in smart grid energy management).

Formally, the constraint-coupled optimization problem can be stated as

$$\begin{aligned} \min_{\mathbf{x}_1,\ldots,\mathbf{x}_N} \ & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to } & \mathbf{x}_i \in X_i, \qquad i \in \{1,\ldots,N\} \\ & \sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}, \end{aligned} \tag{1.11}$$

where $X_i \subseteq \mathbb{R}^{n_i}$, $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ and $\mathbf{g}_i : \mathbb{R}^{n_i} \to \mathbb{R}^S$ are known only to agent i, for all $i \in \{1,\ldots,N\}$. Moreover, we stress that each agent i is interested in computing only the component $\mathbf{x}_i^\star$ of an optimal solution $(\mathbf{x}_1^\star,\ldots,\mathbf{x}_N^\star)$ of (1.11). Notice that a similar set-up has been investigated in the context of resource allocation. We point out, that differently from several existing works, we focus on problems where the coupling can be nonlinear and local constraints are present.

Remark 1.4.1 (Comparison with the cost-coupled set-up)**.** *We notice that problem* (1.11) *can be seen as a special case of problem* (1.1)*, where additional structure is assumed, i.e., each cost function f_i depends on a component only, say $\mathbf{x}_i$, of the decision variable $\mathbf{x}$ and the constraints have a sparse structure. From a* centralized *perspective this is true, however in a distributed scenario this problem set-up introduces novel challenges when agents know only their portion of the cost and the constraints and want to obtain only their portion of the optimal solution. This is a distinctive feature that calls for specialized methodologies and algorithms that are different from the ones developed for cost-coupled problems.* □

A constraint-coupled optimization set-up (1.11) which is particularly relevant in our control community is distributed Model Predictive Control (MPC) in which the goal is to design a feedback control law for a (spatially distributed) network of dynamical systems based on the MPC concept. In such a scheme several optimization problems need to be iteratively solved.

The local decision variable of each agent corresponds to its state-input trajectory, while the local constraints encode its dynamics, which is usually independent of other agents. A constraint that couples agents' states, inputs or outputs needs to be taken into account in order to enforce cooperative tasks as, e.g., formation control, or to take into account common bounds, e.g., due to shared resources. Distributed MPC approaches are mainly classified into non-cooperative and cooperative schemes, see, e.g., [28] for a tutorial. While in non-cooperative schemes the main focus is guaranteeing recursive feasibility and stability without enforcing global optimality at each iteration, in cooperative approaches the agents try to solve in a distributed way the global, constraint-coupled optimization problems arising in each time window.

In the following, we briefly discuss how an instance of a finite horizon optimization problem arising in a distributed MPC scheme fits setup (1.11). Suppose we are given a system with N agents. Each agent i has a discrete-time state-input trajectory over a finite horizon T, i.e., $[z_i(0), \dots, z_i(T), u_i(0), \dots, u_i(T-1)]^\top$, where $z_i(k)$ is the state and $u_i(k)$ is the input. The trajectory must satisfy some local constraints as, e.g., the dynamical model and state-input constraints that we model, for the sake of simplicity, as a linear system and box constraints, respectively. Moreover, additional constraints might be present that enforce some cooperative tasks as, e.g., the sum of the states at each time instant cannot exceed a given value, i.e., $\sum_{i=1}^{N} z_i(k) \leq h$ for all $k \in [0, T]$ and for some $h \in \mathbb{R}$. Finally, agents run an optimization algorithm in order to select a trajectory that minimizes a stage cost $\sum_{k=1}^{T} \ell_i(z_i(k), u_i(k))$ measuring some performance of each system, e.g., distance from a given desired profile of the trajectory. Thus, formally the problem can be stated as

$$\begin{aligned}
\min \ & \sum_{i=1}^{N} \Big(\sum_{k=0}^{T-1} \ell_i(z_i(k), u_i(k)) + V_i(z_i(T)) \Big) \\
\text{subj. to } & z_i(k+1) = A_i z_i(k) + B_i u_i(k), \quad \forall\, k \in [0, T-1],\ i \in \{1, \dots, N\} \\
& c_1 \leq z_i(k) \leq c_2, u_1 \leq u_i(k) \leq d_2, \ \forall\, k \in [1, T-1],\ i \in \{1, \dots, N\} \\
& \sum_{i=1}^{N} z_i(k) \leq h, \qquad \forall\, k \in [0, T],
\end{aligned} \tag{1.12}$$

where $z_i(0)$ is given for all $i \in \{1, \dots, N\}$. It is easy to see that by setting $\mathbf{x}_i$ equal to the state-input trajectory of agent i, collecting the stage cost plus the terminal cost of agent i into $f_i(\mathbf{x}_i)$, defining the contribution of agent i to the coupling constraints as $\mathbf{g}_i(\mathbf{x}_i)$ and encoding the dynamics and the constraints on states and inputs in X_i, then problem (1.12) is an instance of problem (1.11).

1.5 Examples of Constraint-Coupled Optimization

In this section we describe two examples related to distributed Model Predictive Control (MPC) and a Demand-Side Management (DSM) program in smart grid respectively.

1.5.1 Distributed Model Predictive Control of Microgrids

Next, we introduce a microgrid control scenario that we then fit into our constraint-coupled set-up. A microgrid consists of several generators, controllable loads, storage devices and a connection to the main grid. In the following, we use the notational convention that energy generation corresponds to positive variables, while energy consumption corresponds to negative variables. Generators are collected in the set $\texttt{GEN}$. At each time instant τ in a given horizon $[0, T]$, they generate power, denoted by $p^\tau_{\texttt{gen},i}$, that must satisfy magnitude and rate bounds, i.e., for given positive scalars $\underline{p}$, $\bar{p}$, $\underline{r}$ and $\bar{r}$, it must hold, for all $i \in \texttt{GEN}$, $\underline{p} \leq p^\tau_{\texttt{gen},i} \leq \bar{p}$, with $\tau \in [0, T]$, and $\underline{r} \leq p^{\tau+1}_{\texttt{gen},i} - p^\tau_{\texttt{gen},i} \leq \bar{r}$, with $\tau \in [0, T-1]$. The cost to produce power by a generator is modeled as a quadratic function $f^\tau_{\texttt{gen},i} = \alpha_1 p^\tau_{\texttt{gen},i} + \alpha_2 (p^\tau_{\texttt{gen},i})^2$ with α_1 and α_2 positive scalars. Storage devices are collected in $\texttt{STOR}$ and their power is denoted by $p^\tau_{\texttt{stor},i}$ and satisfies bounds and a dynamical constraint given by $-d_{\texttt{stor}} \leq p^\tau_{\texttt{stor},i} \leq c_{\texttt{stor}}$, $\tau \in [0, T]$, $q^{\tau+1}_{\texttt{stor},i} = q^\tau_{\texttt{stor},i} + p^\tau_{\texttt{stor},i}$, $\tau \in [0, T-1]$, and $0 \leq q^\tau_{\texttt{stor},i} \leq q_{\max}$, $\tau \in [0, T]$, where the initial capacity $q^0_{\texttt{stor},i}$ is given and $d_{\texttt{stor}}$, $c_{\texttt{stor}}$ and $q_{\max}$ are positive scalars. There are no costs associated with the stored power. Controllable loads are collected in $\texttt{CONL}$ and their power is denoted by $p^\tau_{\texttt{conl},i}$. A desired load profile $p^\tau_{\texttt{des},i}$ for $p^\tau_{\texttt{conl},i}$ is given and the controllable load incurs in a cost $f^\tau_{\texttt{conl},i} = \beta \max\{0, p^\tau_{\texttt{des},i} - p^\tau_{\texttt{conl},i}\}$, $\beta \geq 0$, if the desired load is not satisfied. Finally, the device $i = N$ is the connection node with the main grid; its power is denoted as $p^\tau_{\texttt{tr}}$ and must satisfy $|p^\tau_{\texttt{tr}}| \leq E$, $\tau \in [0, T]$. The power-trading cost is modeled as $f^\tau_{\texttt{tr}} = -c_1 p^\tau_{\texttt{tr}} + c_2 |p^\tau_{\texttt{tr}}|$, with c_1 and c_2 positive scalars corresponding to the price and a general transaction cost respectively.

The power network must satisfy a given power demand D^τ modeled by a *coupling constraint* among the units

$$\sum_{i \in \texttt{GEN}} p^\tau_{\texttt{gen},i} + \sum_{i \in \texttt{STOR}} p^\tau_{\texttt{stor},i} + \sum_{i \in \texttt{CONL}} p^\tau_{\texttt{conl},i} + p^\tau_{\texttt{tr}} - D^\tau = 0,$$

for all $\tau \in [0, T]$. Reasonably, we assume D^τ to be known only by the connection node $\texttt{tr}$.

In order to cast the microgrid control problem as (1.11), we let each $\mathbf{x}_i$ be the whole trajectory over the prediction horizon $[0, T]$, e.g.,

$$\mathbf{x}_i \triangleq \left[p^0_{\texttt{gen},i}, \ldots, p^T_{\texttt{gen},i}\right]^\top,$$

for all the generators $i \in \texttt{GEN}$ and, consistently, for the other device types. As for the cost functions we can define $f_i(\mathbf{x}_i) \triangleq \sum_{\tau=0}^{T} f^{\tau}_{\texttt{gen},i}(p^{\tau}_{\texttt{gen},i})$ for $i \in \texttt{GEN}$ and, consistently, for the other device types. The local X_i are given by the constraint described above for each unit type.

1.5.2 Distributed Peak Minimization in Smart Grids

Another important optimization scenario in smart grids arises in designing smart generators, accumulators and loads that cooperatively execute Demand Side Management (DSM) programs [3]. Here, the goal is to reduce the hourly and daily variations and peaks of electric demand by optimizing generation, storage and consumption. A widely adopted objective in DSM programs is Peak-to-Average Ratio (PAR), defined as the ratio between peak-daily and average-daily power demands. PAR minimization gives rise to a min-max optimization problem if the average daily electric load is assumed not to be affected by the demand response strategy, [54].

We consider a network of N processors that want to solve a peak-minimization problem in a distributed way. We assume that the interaction occurs according to a connected, undirected graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$, where $\mathcal{E}$ is the set of edges. Moreover, we associate to each processor i a decision vector $\mathbf{x}_i = [x_{i,1}, \dots, x_{i,S}]^\top \in \mathbb{R}^S$, a constraint set $X_i \subseteq \mathbb{R}^S$ and local functions $g_{i,s}$, $s \in \{1, \dots, S\}$. We use the subscript pair (i, s) to indicate the s-th component of a vector in $\mathbb{R}^S$ that belongs to node i. In the peak-minimization problem, S represents the horizon on which we want to reduce the peak. Thus, we set-up the following optimization problem

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \ & \max_{s\in\{1,\dots,S\}} \ \sum_{i=1}^{N} g_{i,s}(x_{i,s}) \\ \text{subj. to } \ & \mathbf{x}_i \in X_i, \qquad\qquad i \in \{1, \dots, N\}, \end{aligned} \tag{1.13}$$

where, for all $i \in \{1, \dots, N\}$, the constraint set $X_i \subseteq \mathbb{R}^S$ and the functions $g_{i,s} : \mathbb{R} \to \mathbb{R}$, $s \in \{1, \dots, S\}$, are known only to agent i.

Using a standard approach for min-max problems, we introduce an auxiliary variable P to write the so-called epigraph representation of problem (1.13), given by

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N,P} \ & P \\ \text{subj. to } \ & \mathbf{x}_i \in X_i, \qquad\qquad i \in \{1, \dots, N\}, \\ & \sum_{i=1}^{N} g_{i,s}(x_{i,s}) \le P, \quad s \in \{1, \dots, S\}. \end{aligned} \tag{1.14}$$

It is worth noticing that this problem has a particular structure, which gives rise to interesting challenges in a distributed set-up. First of all,

two types of couplings are present, which involve simultaneously the N agents and the S components of each decision variable $\mathbf{x}_i$. Specifically, for a given slot s, the constraint $\sum_{i=1}^{N} g_{i,s}(x_{i,s}) \leq P$ couples all the vectors $\mathbf{x}_i$, $i \in \{1, \ldots, N\}$. At the same time, for a given agent $i \in \{1, \ldots, N\}$, the constraint X_i couples all the components $x_{i,1}, \ldots, x_{i,S}$ of $\mathbf{x}_i$. Figure 1.6 provides a nice graphical representation of this interlaced coupling. The horizontal

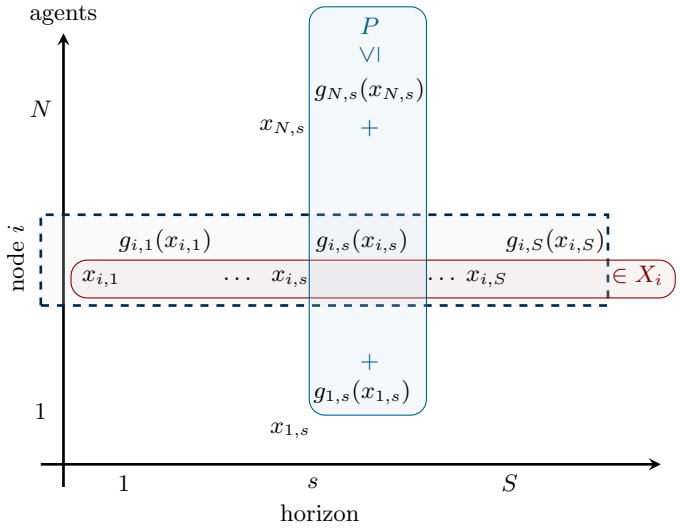


Figure 1.6: Graphical representation of interlaced constraints.

axis describes the local coupling at each node, while the vertical dimension models the coupling among agents. Moreover, the problem is both large-scale and big-data. That is, both the number of decision variables and the number of constraints depend on N (and thus scale badly with the number of agents in the network). Also, the dimension of the coupling constraint, S, can be large. Therefore, common approaches as reaching a consensus among the nodes on an optimal solution and/or exchanging constraints are not computationally affordable.

Chapter 2

Asynchronous Distributed Optimization via Randomized Dual Proximal Gradient

In this chapter we consider composite convex optimization problems introduced in Section 1.2.1 and design distributed algorithms based on duality. In particular, we generalize the classical distributed dual decomposition in two ways: (i) we handle composite costs with local constraints, and (ii) we use local, constant step-sizes in the ascent steps. First, we propose and prove the convergence of a synchronous distributed algorithm based on a proper proximal gradient method applied to a dual problem formulation. Second, we extend the synchronous algorithm to two asynchronous versions in which agents perform their local computation based on local timers (see Section 1.1), without any centralized synchronization by exploiting a block-coordinate proximal method. The results of this chapter are based on [74, 75].

2.1 Literature Review

Early references on distributed optimization algorithms involved primal and dual subgradient methods and Alternating Direction Method of Multipliers (ADMM), designed for synchronous communication protocols over fixed graphs. More recently time-varying versions of these algorithmic ideas have been proposed to cope with more realistic peer-to-peer network scenarios. A Newton-Raphson consensus strategy is proposed in [105] to solve unconstrained, convex optimization problems under asynchronous, symmetric gossip communications. In [89] a primal, synchronous algorithm, called EXTRA, is proposed to solve smooth, unconstrained optimization problems. In [41] the authors propose accelerated distributed gradient methods for unconstrained optimization problems over symmetric, time-varying networks. In order to deal with time-varying and directed graph topologies, in

[60] a push-sum algorithm for average consensus is combined with a primal subgradient method in order to solve unconstrained convex optimization problems. The work presented in [1] extends this algorithm to online distributed optimization over time-varying, directed networks. In [46] a novel class of continuous-time, gradient-based distributed algorithms is proposed both for fixed and time-varying graphs and conditions for exponential convergence are provided. A distributed (primal) proximal-gradient method is proposed in [27] to solve (over time-varying, balanced communication graphs) optimization problems with a separable cost function including local differentiable components and a common non-differentiable term. In [97] experiments of a dual averaging algorithm are run for separable problems with a common constraint on time-varying and directed networks.

For general constrained convex optimization problems, in [47] the authors propose a distributed random projection algorithm, that can be used by multiple agents connected over a (balanced) time-varying network. In [57] the author proposes (primal) randomized block-coordinate descent methods for minimizing multi-agent convex optimization problems with linearly coupled constraints over networks. In [99] an asynchronous ADMM-based distributed method is proposed for a separable, constrained optimization problem. The algorithm is shown to converge at the rate $O(1/t)$ (being t the iteration counter). In [40] the ADMM approach is proposed for a more general framework, thus yielding a continuum of algorithms ranging from a fully centralized to a fully distributed. In [13] a method, called ADMM+, is proposed to solve separable, convex optimization problems with a cost function written as the sum of a smooth and a nonsmooth term.

Successive block-coordinate updates are proposed in [33, 86] to solve separable optimization problems in a parallel big-data setting. Another class of algorithms exploits the exchange of active constraints among the network nodes to solve constrained optimization problems [78]. This idea has been combined with dual decomposition and cutting-plane methods to solve robust convex optimization problems via polyhedral approximations [20]. These algorithms work under asynchronous, directed and unreliable communication.

2.2 Dual Decomposition Approach Development

We start by introducing some useful definitions that will be used later. Given a closed, nonempty convex set X, the indicator function of $X \subseteq \mathbb{R}^d$ is defined as $I_X(\mathbf{x}) = 0$ if $\mathbf{x} \in X$ and $I_X(\mathbf{x}) = +\infty$ otherwise.

Let $\varphi : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, its conjugate function $\varphi^* : \mathbb{R}^d \to \mathbb{R}$ is defined as $\varphi^*(\mathbf{y}) := \sup_{\mathbf{x}} \left\{\mathbf{y}^\top \mathbf{x} - \varphi(\mathbf{x})\right\}$. Let $\varphi : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper, convex function and α a positive scalar, the proximal operator $\mathbf{prox}_{\alpha\varphi} : \mathbb{R}^d \to \mathbb{R}^d$ is defined by $\mathbf{prox}_{\alpha\varphi}(\mathbf{v}) := \operatorname{argmin}_{\mathbf{x}} \left\{\varphi(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x}-\mathbf{v}\|^2\right\}$. We also

introduce a generalized version of the proximal operator. Given a positive definite matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$, we define

$$\mathbf{prox}_{\mathbf{W},\varphi}(\mathbf{v}) := \operatorname*{argmin}_{\mathbf{x}} \Big\{ \varphi(\mathbf{x}) + \frac{1}{2}\big\|\mathbf{x}-\mathbf{v}\big\|^2_{\mathbf{W}^{-1}} \Big\}.$$

We consider the optimization problem introduced in Section 1.2 and we report it here for easing the discussion

$$\min_{\mathbf{x}} \ \sum_{i=1}^{N} \Big(f_i(\mathbf{x}) + g_i(\mathbf{x}) \Big), \tag{2.1}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed and strongly convex extended real-valued function with strong convexity parameter $\sigma_i > 0$ and each $g_i : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex extended real-valued functions.

Note that, given an optimization problem, the split of f_i and g_i may be non-unique and may depend on the problem structure. Roughly speaking, on f_i an easy minimization step can be performed (e.g., by division free operations, [79]), while g_i has an easy expression of its proximal operator.

Remark 2.2.1 (Min-max problems)**.** *Our set-up does not require f_i to be differentiable, thus one can also consider each strongly convex function f_i given by $f_i(\mathbf{x}) := \max_{j\in\{1,\dots,m_i\}} f_{ij}(\mathbf{x})$, $m_i \in \mathbb{N}$, where $\{f_{ij}(\mathbf{x}) \mid j \in \{1,\dots,m_i\}\}$ is a nonempty collection of strongly convex functions.* □

Since we will work on a dual formulation of problem (2.1), we introduce the next standard assumption which guarantees that a dual approach is applicable, i.e. the dual problem is equivalent to the primal (strong duality).

Assumption 2.2.2 (Constraint qualification)**.** *The intersection of the relative interior of* $\mathbf{dom} \sum_{i=1}^{N} f_i$ *and the relative interior of* $\mathbf{dom} \sum_{i=1}^{N} g_i$ *is non-empty.* □

We want the optimization problem (2.1) to be solved in a distributed way by a network of peer processors without a central coordinator. Each processor has a local memory, a local computation capability and can exchange information with neighboring nodes. We assume that the communication can occur among nodes that are neighbors in a given fixed, undirected and connected graph $\mathcal{G} = (\{1,\dots,N\}, \mathcal{E})$, where $\mathcal{E} \subseteq \{1,\dots,N\} \times \{1,\dots,N\}$ is the set of edges. That is, the edge (i,j) models the fact that node i and j can exchange information. We recall that $\mathcal{N}_i$ denotes the set of *neighbors* of node i in the fixed graph $\mathcal{G}$, i.e., $\mathcal{N}_i := \{j \in \{1,\dots,N\} \mid (i,j) \in \mathcal{E}\}$, and by $|\mathcal{N}_i|$ its cardinality.

To exploit the sparsity of the graph we introduce copies of $\mathbf{x}$ and their coherence (consensus) constraint, so that the optimization problem (2.1) can

be equivalently rewritten as

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \ & \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + g_i(\mathbf{x}_i) \Big) \\ \text{subj. to } & \mathbf{x}_i = \mathbf{x}_j, \qquad\qquad (i,j) \in \mathcal{E}, \end{aligned} \tag{2.2}$$

with $\mathbf{x}_i \in \mathbb{R}^d$ for all $i \in \{1,\dots,N\}$. Since $\mathcal{G}$ is connected, then the equivalence is guaranteed.

Since we propose distributed dual algorithms, next we derive the dual problem and characterize its properties. To obtain the desired separable structure of the dual problem, we set-up an equivalent formulation of problem (2.2) by adding new variables $\mathbf{z}_i$, $i \in \{1,\dots,N\}$, i.e.,

$$\begin{aligned} \min_{\substack{\mathbf{x}_1,\dots,\mathbf{x}_N \\ \mathbf{z}_1,\dots,\mathbf{z}_N}} \ & \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + g_i(\mathbf{z}_i) \Big) \\ \text{subj. to } & \mathbf{x}_i = \mathbf{x}_j \qquad (i,j) \in \mathcal{E}, \\ & \mathbf{x}_i = \mathbf{z}_i \qquad i \in \{1,\dots,N\}. \end{aligned} \tag{2.3}$$

Let $\mathbf{X} = [\mathbf{x}_1^\top \ \dots \ \mathbf{x}_N^\top]^\top$ and $\mathbf{Z} = [\mathbf{z}_1^\top \ \dots \ \mathbf{z}_N^\top]^\top$, the Lagrangian of primal problem (2.3) is given by

$$\begin{aligned} \mathcal{L}(\mathbf{X},\mathbf{Z},\boldsymbol{\Lambda},\boldsymbol{\mu}) &= \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + g_i(\mathbf{z}_i) + \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ij}^\top(\mathbf{x}_i - \mathbf{x}_j) + \boldsymbol{\mu}_i^\top(\mathbf{x}_i - \mathbf{z}_i) \Big) \\ &= \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ij}^\top(\mathbf{x}_i - \mathbf{x}_j) + \boldsymbol{\mu}_i^\top \mathbf{x}_i + g_i(\mathbf{z}_i) - \boldsymbol{\mu}_i^\top \mathbf{z}_i \Big), \end{aligned}$$

where $\boldsymbol{\Lambda}$ and $\boldsymbol{\mu}$ are respectively the vectors of the Lagrange multipliers $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^d$, $(i,j) \in \mathcal{E}$, and $\boldsymbol{\mu}_i \in \mathbb{R}^d$, $i \in \{1,\dots,N\}$, and in the last line we have separated the terms containing $\mathbf{x}_i$ and $\mathbf{z}_i$. Since $\mathcal{G}$ is undirected, the Lagrangian can be rearranged as

$$\mathcal{L}(\mathbf{X},\mathbf{Z},\boldsymbol{\Lambda},\boldsymbol{\mu}) = \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \Big(\sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) + \boldsymbol{\mu}_i \Big) + g_i(\mathbf{z}_i) - \mathbf{z}_i^\top \boldsymbol{\mu}_i \Big).$$

The dual function is

$$\begin{aligned} q(\boldsymbol{\Lambda}, \boldsymbol{\mu}) &= \min_{\mathbf{X},\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\Lambda}, \boldsymbol{\mu}) \\ &= \min_{\mathbf{X}} \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \Big(\sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) + \boldsymbol{\mu}_i \Big) \Big) \\ &\quad + \min_{\mathbf{Z}} \sum_{i=1}^{N} \Big(g_i(\mathbf{z}_i) - \mathbf{z}_i^\top \boldsymbol{\mu}_i \Big) \\ &= \sum_{i=1}^{N} \min_{\mathbf{x}_i} \Big(f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \Big(\sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) + \boldsymbol{\mu}_i \Big) \Big) \\ &\quad + \sum_{i=1}^{N} \min_{\mathbf{z}_i} \Big(g_i(\mathbf{z}_i) - \mathbf{z}_i^\top \boldsymbol{\mu}_i \Big), \end{aligned}$$

where we have used the separability of the Lagrangian with respect to each $\mathbf{x}_i$ and each $\mathbf{z}_i$. Then, by using the definition of conjugate function,the dual function can be expressed as

$$q(\boldsymbol{\Lambda}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \Big(- f_i^* \Big(- \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) - \boldsymbol{\mu}_i \Big) - g_i^*(\boldsymbol{\mu}_i) \Big).$$

The dual problem of (2.3) consists in maximizing the dual function with respect to dual variables $\boldsymbol{\Lambda}$ and $\boldsymbol{\mu}$, i.e.,

$$\max_{\boldsymbol{\Lambda}, \boldsymbol{\mu}} \sum_{i=1}^{N} \Big(- f_i^* \Big(- \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) - \boldsymbol{\mu}_i \Big) - g_i^*(\boldsymbol{\mu}_i) \Big). \tag{2.4}$$

Under Assumption 2.2.2 the dual problem (2.4) is feasible (see [9, Chapter 5]) and strong duality holds, so that (2.4) can be equivalently solved to get a solution of (2.3).

In order to have a more compact notation for problem (2.4), we stack the dual variables as $\mathbf{y} = [\mathbf{y}_1^\top \ \dots \ \mathbf{y}_N^\top]^\top$, where

$$\mathbf{y}_i = \begin{bmatrix} \boldsymbol{\Lambda}_i \\ \boldsymbol{\mu}_i \end{bmatrix} \in \mathbb{R}^{d|\mathcal{N}_i|+d} \tag{2.5}$$

with $\boldsymbol{\Lambda}_i \in \mathbb{R}^{d|\mathcal{N}_i|}$ a vector whose block-component associated to neighbor j is $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^d$. Thus, changing sign to the cost function, dual problem (2.4) can be restated as

$$\min_{\mathbf{y}} \Gamma(\mathbf{y}) = F^*(\mathbf{y}) + G^*(\mathbf{y}), \tag{2.6}$$

where

$$F^*(\mathbf{y}) = \sum_{i=1}^{N} f_i^*\Big(- \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) - \boldsymbol{\mu}_i\Big),$$
$$G^*(\mathbf{y}) = \sum_{i=1}^{N} g_i^*(\boldsymbol{\mu}_i).$$

2.3 Distributed Dual Proximal Algorithms

In this section we derive the proposed distributed optimization algorithms based on dual proximal gradient methods.

2.3.1 Distributed Dual Proximal Gradient (DDPG)

We begin by deriving a synchronous algorithm on a fixed graph. We assume that all the nodes share a common clock. At each time instant $t \in \mathbb{N}$, every node communicates with its neighbors in the graph $\mathcal{G} = (\{1,\dots,N\}, \mathcal{E})$ and updates its local variables.

First, we provide an informal description of the distributed optimization algorithm. Each node $i \in \{1,\dots,N\}$ stores a set of local dual variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$, and $\boldsymbol{\mu}_i$, updated through a local proximal gradient step, and a primal variable $\mathbf{x}_i^\star$, updated through a local minimization. Each node uses a properly chosen, *local* step-size α_i for the proximal gradient step. Then, the updated primal and dual values are exchanged with the neighboring nodes. The local dual variables at node i are initialized as $\boldsymbol{\lambda}_{ij0}$, $j \in \mathcal{N}_i$, and $\boldsymbol{\mu}_{i0}$. A pseudo-code of the local update at each node is given in Distributed Algorithm 1.

Remark 2.3.1. *In order to start the algorithm, a preliminary communication step is needed in which each node i receives from each neighbor j its $\boldsymbol{\lambda}_{ji0}$ (to compute $\mathbf{x}_i^0$) and the convexity parameter σ_j of f_j (to set α_i, as it will be clear from the analysis in Section 2.4.2). This step can be avoided if the nodes agree to set $\boldsymbol{\lambda}_{ij0} = \mathbf{0}$ and know a bound for α_i. Also, it is worth noting that, differently from other existing algorithms, in general $\boldsymbol{\lambda}_{ij}^t \neq -\boldsymbol{\lambda}_{ji}^t$.* □

We point out once more that to run the DDPG algorithm the nodes need to have a common clock. Also, it is worth noting that, as it will be clear from the analysis in Section 2.4.2, to set the local step-size each node needs to know the number of nodes, N, in the network. In the next subsections we present two asynchronous distributed algorithms which overcome these limitations.

Distributed Algorithm 1 DDPG

Processor states: $\mathbf{x}_i$, $\boldsymbol{\lambda}_{ij}$ for all $j \in \mathcal{N}_i$ and $\boldsymbol{\mu}_i$

Initialization: $\boldsymbol{\lambda}_{ij}^0 = \boldsymbol{\lambda}_{ij0}$ for all $j \in \mathcal{N}_i$, $\boldsymbol{\mu}_i^0 = \boldsymbol{\mu}_{i0}$ and

$$\mathbf{x}_i^0 = \operatorname*{argmin}_{\mathbf{x}_i} \left\{ \mathbf{x}_i^\top \left(\sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij0} - \boldsymbol{\lambda}_{ji0}) + \boldsymbol{\mu}_{i0} \right) + f_i(\mathbf{x}_i) \right\}$$

Evolution:

FOR: $t = 1, 2, \ldots$ DO

receive $\mathbf{x}_j^t$ for each $j \in \mathcal{N}_i$, update

$$\begin{aligned} \boldsymbol{\lambda}_{ij}^{t+1} &= \boldsymbol{\lambda}_{ij}^t + \alpha_i (\mathbf{x}_i^t - \mathbf{x}_j^t) \\ \tilde{\boldsymbol{\mu}}_i &= \boldsymbol{\mu}_i^t + \alpha_i \, \mathbf{x}_i^t \\ \boldsymbol{\mu}_i^{t+1} &= \tilde{\boldsymbol{\mu}}_i - \alpha_i \, \mathbf{prox}_{\frac{1}{\alpha_i} g_i} \left(\frac{\tilde{\boldsymbol{\mu}}_i}{\alpha_i} \right) \end{aligned}$$

receive $\boldsymbol{\lambda}_{ji}^{t+1}$ for each $j \in \mathcal{N}_i$ and compute

$$\mathbf{x}_i^{t+1} = \operatorname*{argmin}_{\mathbf{x}_i} \left\{ \mathbf{x}_i^\top \left(\sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{t+1} - \boldsymbol{\lambda}_{ji}^{t+1} \right) + \boldsymbol{\mu}_i^{t+1} \right) + f_i(\mathbf{x}_i) \right\} \tag{2.7}$$

2.3.2 Asynchronous Node-based DDPG (A-DDPG)

In this subsection, we propose a node-based asynchronous algorithm. We consider an asynchronous protocol where each node has its own concept of time defined by a local timer, which randomly and independently of the other nodes triggers when to awake itself. Between two triggering events the node is in an *idle* mode, i.e., it continuously receives messages from neighboring nodes. When a trigger occurs, it switches into an *awake* mode in which it updates its local (primal and dual) variables and transmits the updated information to its neighbors.

Formally, the triggering process is modeled by means of a local clock $\tau_i \in \mathbb{R}_{\geq 0}$ and a randomly generated waiting time T_i. As long as $\tau_i < T_i$ the node is in idle mode. When $\tau_i = T_i$ the node switches to the awake mode and, after running the local computation, resets $\tau_i = 0$ and draws a new realization of the random variable T_i. We make the following assumption on the local waiting times T_i.

Assumption 2.3.2 (Exponential i.i.d.[1] local timers)**.** *The waiting times between consecutive triggering events are i.i.d. random variables with the same exponential distribution.* □

[1] Independent and Identically Distributed.

When a node i is in idle, it continuously receives messages from awake neighbors. If the local timer τ_i triggers or new dual variables $\boldsymbol{\lambda}_{ji}$ are received, it wakes up. When a node i wakes up, its primal variable $\mathbf{x}_i$ through a local minimization. Moreover, if the transition was due to the local timer triggering, then it also updates and broadcasts its local dual variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$ and $\boldsymbol{\mu}_i$ by a local proximal gradient step. The *local* step-size of the proximal gradient step for node i is denoted by α_i. In order to highlight the difference between updated and old variables at node i during the "awake" phase, we denote the updated ones as $\boldsymbol{\lambda}_{ij}^+$ and $\boldsymbol{\mu}_i^+$ respectively. Notice that we tacitly assume that agents are able to complete their local computation and communication before another triggering in the network occurs. It is worth noting that, being the algorithm asynchronous, there is no common clock as in the synchronous version. The algorithm is reported Distributed Algorithm 2 from the perspective of agent i.

Distributed Algorithm 2 Node-based A-DDPG

Processor states: $\mathbf{x}_i$, $\boldsymbol{\lambda}_{ij}$ for all $j \in \mathcal{N}_i$ and $\boldsymbol{\mu}_i$

Initialization: $\boldsymbol{\lambda}_{ij} = \boldsymbol{\lambda}_{ij,0}$ for all $j \in \mathcal{N}_i$, $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i0}$ and
$\mathbf{x}_i = \underset{\mathbf{x}_i}{\operatorname{argmin}} \Big\{ \mathbf{x}_i^\top \Big(\sum_{j \in \mathcal{N}_i} \Big(\boldsymbol{\lambda}_{ij,0} - \boldsymbol{\lambda}_{ji,0} \Big) + \boldsymbol{\mu}_{i0} \Big) + f_i(\mathbf{x}_i) \Big\}$
set $\tau_i = 0$ and get a realization T_i

Evolution:
IDLE:
WHILE: $\tau_i < T_i$ DO:
Receive $\mathbf{x}_j$ and/or $\boldsymbol{\lambda}_{ji}$ from each $j \in \mathcal{N}_i$.
IF: dual variables are received, THEN: go to *AWAKE*.

AWAKE:
Compute and broadcast

$$\mathbf{x}_i^+ = \underset{\mathbf{x}_i}{\operatorname{argmin}} \left\{ \mathbf{x}_i^\top \left(\sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^+ - \boldsymbol{\lambda}_{ji} \right) + \boldsymbol{\mu}_i^+ \right) + f_i(\mathbf{x}_i) \right\} \tag{2.8}$$

IF: $\tau_i = T_i$, THEN: update and broadcast

$$\begin{aligned} \boldsymbol{\lambda}_{ij}^+ &= \boldsymbol{\lambda}_{ij} + \alpha_i \big(\mathbf{x}_i^+ - \mathbf{x}_j \big), \qquad \text{for all } j \in \mathcal{N}_i \\ \tilde{\boldsymbol{\mu}}_i &= \boldsymbol{\mu}_i + \alpha_i \, \mathbf{x}_i^+ \\ \boldsymbol{\mu}_i^+ &= \tilde{\boldsymbol{\mu}}_i - \alpha_i \, \mathbf{prox}_{\frac{1}{\alpha_i} g_i} \left(\frac{\tilde{\boldsymbol{\mu}}_i}{\alpha_i} \right) \end{aligned}$$

set $\tau_i = 0$, get a new realization T_i and go to *IDLE*.

2.3.3 Asynchronous Edge-based DDPG

In this subsection we present a variant of the A-DDPG in which an edge becomes active uniformly at random, rather than a node. In other words, we assume that timers are associated to edges, rather than to nodes, that is a waiting time T_{ij} is extracted for edge (i, j). The processor states and their initialization stay the same except for the timers. Notice that here we are assuming that nodes i and j have a common waiting time T_{ij} and, consistently, a local timer τ_{ij}. Each waiting time T_{ij} satisfies Assumption 2.3.2. In Distributed Algorithm 3 we report (only) the evolution for this modified scenario. In this edge-based algorithm the dual variable $\boldsymbol{\mu}_i$ (associated to the constraint $\mathbf{x}_i = \mathbf{z}_i$) cannot be updated every time an edge (i, j), $j \in \mathcal{N}_i$, becomes active (otherwise it would be updated more often than the variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$). Thus, we identify one special neighbor $j_{\boldsymbol{\mu}_i} \in \mathcal{N}_i$ and update $\boldsymbol{\mu}_i$ only when the edge $(i, j_{\boldsymbol{\mu}_i})$ is active.

Distributed Algorithm 3 Edge-Based Formulation of Distributed Algorithm 2 (evolution)

IDLE: WHILE $\tau_{ij} < T_{ij}$ DO: nothing
go to *AWAKE*.
AWAKE:
send $\mathbf{x}_i$ to j and receive $\mathbf{x}_j$ from j
update and send to j, $\boldsymbol{\lambda}_{ij}^+ = \boldsymbol{\lambda}_{ij} + \alpha_i(\mathbf{x}_i - \mathbf{x}_j)$
IF: $j = j_{\boldsymbol{\mu}_i}$ THEN: update $\boldsymbol{\mu}_i^+ = \mathbf{prox}_{\alpha_i g_i^*}(\boldsymbol{\mu}_i + \alpha_i\, \mathbf{x}_i)$
compute

$$\mathbf{x}_i^+ = \operatorname*{argmin}_{\mathbf{x}_i} \left\{ \mathbf{x}_i^\top \left(\sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^+ - \boldsymbol{\lambda}_{ji} \right) + \boldsymbol{\mu}_i^+ \right) + f_i(\mathbf{x}_i) \right\}$$

set $\tau_{ij} = 0$, deal on a new realization T_{ij} and go to *IDLE*.

2.4 Analysis of the Distributed Algorithms

To start the analysis we first introduce a more general version of (centralized) proximal gradient methods, namely by introducing a weight matrix in the proximal step.

2.4.1 Weighted Proximal Gradient Methods

Consider the following composite optimization problem

$$\min_{\mathbf{y} \in \mathbb{R}^d} \; \Gamma(\mathbf{y}) := \Phi(\mathbf{y}) + \Psi(\mathbf{y}), \tag{2.9}$$

where $\Phi : \mathbb{R}^d \to \mathbb{R}$ and $\Psi : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ are convex functions.

We decompose the decision variable as $\mathbf{y} = [\mathbf{y}_1^\top, \dots, \mathbf{y}_N^\top]^\top$ and, consistently, we decompose the space $\mathbb{R}^d$ into N subspaces as follows. Let $\mathbf{U} \in \mathbb{R}^{d\times d}$ be a column permutation of the $d \times d$ identity matrix and, further, let $\mathbf{U} = [\mathbf{U}_1\ \mathbf{U}_2\ \dots\ \mathbf{U}_N]$ be a decomposition of $\mathbf{U}$ into N submatrices, with $\mathbf{U}_i \in \mathbb{R}^{d\times d_i}$ and $\sum_i d_i = d$. Thus, any vector $\mathbf{y} \in \mathbb{R}^d$ can be uniquely written as $\mathbf{y} = \sum_i \mathbf{U}_i\mathbf{y}_i$ and, vice-versa, $\mathbf{y}_i = \mathbf{U}_i^\top\mathbf{y}$.

We let problem (2.9) satisfy the following assumptions.

Assumption 2.4.1 (**Block Lipschitz continuity of $\nabla\Phi$**)**.** *The gradient of Φ is block coordinate-wise Lipschitz continuous with positive constants $L_1, \dots, L_N$. That is, for all $\mathbf{y} \in \mathbb{R}^d$ and $\mathbf{s}_i \in \mathbb{R}^{d_i}$ it holds*

$$\|\nabla_i\Phi(\mathbf{y} + U_i\mathbf{s}_i) - \nabla_i\Phi(\mathbf{y})\| \le L_i\|\mathbf{s}_i\|,$$

where $\nabla_i\Phi(\mathbf{y})$ is the i-th block component of $\nabla\Phi(\mathbf{y})$. □

Assumption 2.4.2 (**Separability of Ψ**)**.** *The function Ψ is block-separable, i.e., it can be decomposed as $\Psi(\mathbf{y}) = \sum_{i=1}^N \psi_i(\mathbf{y}_i)$, with each $\psi_i : \mathbb{R}^{d_i} \to \mathbb{R} \cup \{+\infty\}$ a proper, closed convex extended real-valued function.* □

Assumption 2.4.3 (**Feasibility**)**.** *The set of minimizers of problem (2.9) is non-empty.* □

Deterministic descent

We first show how the standard proximal gradient algorithm can be generalized by using a weighted proximal operator.

Following the same line of proof as in [6], we can prove that a generalized proximal gradient iteration, given by

$$\begin{aligned}\mathbf{y}^{t+1} &= \mathbf{prox}_{\mathbf{W},\Psi}\Big(\mathbf{y}^t - \mathbf{W}\nabla\Phi\big(\mathbf{y}^t\big)\Big)\\ &= \operatorname*{argmin}_{\mathbf{y}}\left\{\Psi(\mathbf{y}) + \frac{1}{2}\Big\|\mathbf{y} - \Big(\mathbf{y}^t - \mathbf{W}\nabla\Phi\big(\mathbf{y}^t\big)\Big)\Big\|^2_{\mathbf{W}^{-1}}\right\},\end{aligned} \tag{2.10}$$

converges in objective value to the optimal solution of (2.9) with rate $O(1/t)$.

In order to extend the proof of [6, Theorem 3.1] we need to use a generalized version of [6, Lemma 2.1]. To this end we can use a result by Nesterov, given in [66], which is recalled in the following lemma for completeness.

Lemma 2.4.4 (Generalized Descent Lemma)**.** *Let Assumption 2.4.1 hold, then for all $\mathbf{s} \in \mathbb{R}^N$*

$$\Phi(\mathbf{y} + \mathbf{s}) \le \Phi(\mathbf{y}) + \mathbf{s}^\top\nabla\Phi(\mathbf{y}) + \frac{1}{2}\|\,\mathbf{s}\,\|^2_{\mathbf{W}^{-1}},$$

where $\mathbf{W} := \operatorname{diag}(w_1, \dots, w_N)$ satisfies $w_i \le \frac{1}{NL_i}$ for all $i \in \{1, \dots, N\}$. □

Tighter conditions than the one given above can be found in [58, 86].

Theorem 2.4.5. *Let Assumption 2.4.1 and 2.4.3 hold and let* $\{\mathbf{y}^t\}_{t\geq 0}$ *be the sequence generated by iteration* (2.10) *applied to problem* (2.9)*. Then for any* $t \geq 1$*, it holds*

$$\Gamma(\mathbf{y}^t) - \Gamma(\mathbf{y}^\star) \leq \frac{\left\|\mathbf{y}_0 - \mathbf{y}^\star\right\|^2_{\mathbf{W}^{-1}}}{2t},$$

where $\mathbf{W} := \operatorname{diag}(w_1, \dots, w_N)$ *with* $w_i \leq \frac{1}{NL_i}$*,* $\mathbf{y}_0$ *is the initial condition and* $\mathbf{y}^\star$ *is any minimizer of problem* (2.9)*.*

Proof. The theorem is proven by following the same arguments as in [6, Theorem 3.1], but using Lemma 2.4.4 in place of [6, Lemma 2.1]. □

Randomized block-coordinate descent

Next, we present a randomized version of the weighted proximal gradient, proposed in the literature in [85, Algorithm 2] and called Uniform Coordinate Descent for Composite functions (UCDC). It is reported in Algorithm 4.

Algorithm 4 UCDC

Initialization: $\mathbf{y}^0 = \mathbf{y}_0$

FOR: $t = 0, 1, 2, \dots$ DO

choose $i_t \in \{1, \dots, N\}$ with probability $\frac{1}{N}$

compute

$$T^{(i_t)}\big(\mathbf{y}^t\big) = \underset{\mathbf{s}\in\mathbb{R}^{n_{i_t}}}{\operatorname{argmin}} \Big\{ V_{i_t}(\mathbf{y}^t, \mathbf{s}) \Big\} \tag{2.11a}$$

where

$$V_{i_t}(\mathbf{y}, \mathbf{s}) := \nabla_{i_t}\Phi(\mathbf{y})^\top \mathbf{s} + \frac{L_{i_t}}{2}\|\mathbf{s}\|^2 + \psi_{i_t}(\mathbf{y}_{i_t} + \mathbf{s}) \tag{2.11b}$$

update

$$\mathbf{y}^{t+1} = \mathbf{y}^t + U_{i_t} T^{(i_t)}\big(\mathbf{y}^t\big). \tag{2.12}$$

The convergence result for UCDC is given in [85, Theorem 5], here reported for completeness.

Theorem 2.4.6 ([85, Theorem 5])**.** *Let Assumptions 2.4.1, 2.4.2 and 2.4.3 hold. Let* $\Gamma^\star$ *denote the optimal cost of problem* (2.9)*. Then, for any* $\varepsilon \in \big(0, \Gamma(\mathbf{y}_0) - \Gamma^\star\big)$*, there exists* $\bar{t}(\varepsilon, \rho) > 0$ *such that if* $y(t)$ *is the random*

sequence generated by UCDC (Algorithm 4) applied to problem (2.9)*, then for all* $t \geq \bar{t}$ *it holds that*

$$\Pr\left(\Gamma(\mathbf{y}^t) - \Gamma^\star \leq \varepsilon\right) \geq 1 - \rho,$$

where $\mathbf{y}_0 \in \mathbb{R}^N$ *is the initial condition and* $\rho \in (0,1)$ *is the target confidence.* □

2.4.2 Analysis of DDPG

We start this section by recalling some important properties of conjugate functions that will be useful for the convergence analysis of the proposed distributed algorithms.

Lemma 2.4.7 ([7, 18])**.** *Let* φ *be a closed, strictly convex function and* φ^* *its conjugate function. Then*

$$\nabla\varphi^*(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \left\{\mathbf{y}^\top\mathbf{x} - \varphi(\mathbf{x})\right\} = \operatorname*{argmin}_{\mathbf{x}} \left\{\varphi(\mathbf{x}) - \mathbf{y}^\top\mathbf{x}\right\}.$$

Moreover, if φ *is strongly convex with convexity parameter* σ*, then* $\nabla\varphi^*$ *is Lipschitz continuous with Lipschitz constant given by* $\frac{1}{\sigma}$*.* □

In the next lemma we establish some important properties of problem (2.6) that will be useful to analyze the proposed distributed dual proximal algorithms.

Lemma 2.4.8. *Let* $\Phi(\mathbf{y}) := F^*(\mathbf{y})$ *and* $\Psi(\mathbf{y}) := G^*(\mathbf{y})$ *consistently with the notation of problem* (2.9) *in Section 2.4.1. Problem* (2.6) *satisfies Assumption 2.4.1 (block Lipschitz continuity of* $\nabla\Phi$*), with (block) Lipschitz constants given by*

$$L_i = \sqrt{\frac{1}{\sigma_i^2} + \sum_{j\in\mathcal{N}_i}\left(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\right)^2}, \quad i \in \{1,\dots,N\},$$

Assumption 2.4.2 (separability of Ψ*) and Assumption 2.4.3 (feasibility).* □

Proof. The proof is split into blocks, one for each assumption.
Block Lipschitz continuity of ∇F^***:** We show that the gradient of F^* is block coordinate-wise Lipschitz continuous. The i-th block component of ∇F^* is

$$\nabla_i F^*(\mathbf{y}) = \begin{bmatrix}\nabla_{\Lambda_i} F^*(\mathbf{y}) \\ \nabla_{\mu_i} F^*(\mathbf{y})\end{bmatrix},$$

where the block-component of $\nabla_{\Lambda_i} F^*$ associated to neighbor j is given by $\nabla_{\boldsymbol{\lambda}_{ij}} F^*$ and is equal to

$$\nabla_{\boldsymbol{\lambda}_{ij}} F^*(\mathbf{y}) = \nabla f_i^*\Big(-\sum_{k\in\mathcal{N}_i}\Big(\boldsymbol{\lambda}_{ik} - \boldsymbol{\lambda}_{ki}\Big) - \boldsymbol{\mu}_i\Big) - \nabla f_j^*\Big(-\sum_{k\in\mathcal{N}_j}\Big(\boldsymbol{\lambda}_{jk} - \boldsymbol{\lambda}_{kj}\Big) - \boldsymbol{\mu}_j\Big).$$

By Lemma 2.4.7 both ∇f_i^* and ∇f_j^* are Lipschitz continuous with Lipschitz constants $\frac{1}{\sigma_i}$ and $\frac{1}{\sigma_j}$ respectively, thus also $\nabla_{\boldsymbol{\lambda}_{ij}} F^*$ is Lipschitz continuous with constant $L_{ij} = \frac{1}{\sigma_i} + \frac{1}{\sigma_j}$.
By using the (Euclidean) 2-norm, we have that $\nabla_{\boldsymbol{\Lambda}_i} F^*(\mathbf{y})$ is Lipschitz continuous with constant $\sqrt{\sum_{j\in\mathcal{N}_i} L_{ij}^2}$.
Similarly, the gradient of F^* with respect to $\boldsymbol{\mu}_i$ is

$$\nabla_{\boldsymbol{\mu}_i} F^*(\mathbf{y}) = \nabla f_i^*\Big(-\sum_{k\in\mathcal{N}_i}\Big(\boldsymbol{\lambda}_{ik} - \boldsymbol{\lambda}_{ki}\Big) - \boldsymbol{\mu}_i\Big)$$

and is Lipschitz continuous with constant $\frac{1}{\sigma_i}$. Finally, we conclude that $\nabla_i F^*(\mathbf{y})$ is Lipschitz continuous with constant

$$L_i = \sqrt{\frac{1}{\sigma_i^2} + \sum_{j\in\mathcal{N}_i} L_{ij}^2} = \sqrt{\frac{1}{\sigma_i^2} + \sum_{j\in\mathcal{N}_i}\Big(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\Big)^2}.$$

Separability of G^*: By definition $G^*(\mathbf{y}) = \sum_{i=1}^N g_i^*(\boldsymbol{\mu}_i)$, where $\boldsymbol{\mu}_i$ is a component of the block $\mathbf{y}_i$. Thus, denoting $G_i^*(\mathbf{y}_i) := g_i^*(\boldsymbol{\mu}_i)$, it follows immediately $G^*(\mathbf{y}) = \sum_{i=1}^N g_i^*(\boldsymbol{\mu}_i) = \sum_{i=1}^N G_i^*(\mathbf{y}_i)$.

Feasibility: From Assumption 2.2.2 and the convexity condition on f_i and g_i, strong duality holds, i.e., dual problem (2.6) is feasible and admits at least a minimizer $\mathbf{y}^\star$. □

Next, we recall how the proximal operators of a function and its conjugate are related.

Lemma 2.4.9 (Moreau decomposition, [5])**.** *Let $\varphi : \mathbb{R}^d \to \mathbb{R}\cup\{+\infty\}$ be a closed, convex function and φ^* its conjugate. Then, for all $\mathbf{x}\in\mathbb{R}^d$, $\mathbf{x} = \mathbf{prox}_{\varphi}(\mathbf{x}) + \mathbf{prox}_{\varphi^*}(\mathbf{x})$.* □

Lemma 2.4.10 (Extended Moreau decomposition)**.** *Let $\varphi : \mathbb{R}^d \to \mathbb{R}\cup\{+\infty\}$ be a closed, convex function and φ^* its conjugate. Then, for any $\mathbf{x}\in\mathbb{R}^d$ and $\alpha > 0$, it holds*

$$\mathbf{x} = \mathbf{prox}_{\alpha\varphi}\big(\mathbf{x}\big) + \alpha\mathbf{prox}_{\frac{1}{\alpha}\varphi^*}\Big(\frac{\mathbf{x}}{\alpha}\Big).$$

Proof. Let $h(\mathbf{x}) = \alpha\varphi(\mathbf{x})$, then from the Moreau decomposition in Lemma 2.4.9, it holds $\mathbf{x} = \mathbf{prox}_h(\mathbf{x}) + \mathbf{prox}_{h^*}(\mathbf{x})$. To prove the result we simply need to compute $\mathbf{prox}_{h^*}(\mathbf{x})$ in terms of φ^*. First, from the definition of conjugate function we obtain $h^*(\mathbf{x}) = \alpha\varphi^*\left(\frac{\mathbf{x}}{\alpha}\right)$. Then, by using the definition of proximal operator and standard algebraic properties from minimization, it holds true that $\mathbf{prox}_{h^*}(\mathbf{x}) = \alpha\mathbf{prox}_{\frac{1}{\alpha}\varphi^*}\left(\frac{\mathbf{x}}{\alpha}\right)$, so that the proof follows. □

The next lemma shows how the (weighted) proximal operator of G^* can be split into local proximal operators that can be independently carried out by each single node.

Lemma 2.4.11. *Let* $\mathbf{y} = [\mathbf{y}_1^\top \ \dots \ \mathbf{y}_N^\top]^\top \in \mathbb{R}^{N(D+d)}$ *where* $\mathbf{y}_i = [\mathbf{\Lambda}_i^\top \ \boldsymbol{\mu}_i^\top]^\top$ *with* $\mathbf{\Lambda}_i \in \mathbb{R}^D$ *and* $\boldsymbol{\mu}_i \in \mathbb{R}^d$, $i \in \{1, \dots, N\}$. *Let* $G^*(\mathbf{y}) = \sum_{i=1}^N g_i^*(\boldsymbol{\mu}_i)$, *then for a diagonal weight matrix* $\mathbf{D}_\alpha = \mathrm{diag}(\alpha_1, \dots, \alpha_N) > 0$, *the proximal operator* $\mathbf{prox}_{\mathbf{D}_\alpha, G^*}$ *evaluated at* $\mathbf{y}$ *is given by*

$$\mathbf{prox}_{\mathbf{D}_\alpha, G^*}(\mathbf{y}) = \begin{bmatrix} \mathbf{\Lambda}_1 \\ \mathbf{prox}_{\alpha_1 g_1^*}(\boldsymbol{\mu}_1) \\ \vdots \\ \mathbf{\Lambda}_N \\ \mathbf{prox}_{\alpha_N g_N^*}(\boldsymbol{\mu}_N) \end{bmatrix}.$$

□

Proof. Let $\boldsymbol{\eta} = [\boldsymbol{\eta}_1^\top \ \dots \ \boldsymbol{\eta}_N^\top]^\top \in \mathbb{R}^{N(D+d)}$, with $\boldsymbol{\eta}_i = [\mathbf{u}_i^\top \ \mathbf{v}_i^\top]^\top$, $\mathbf{u}_i \in \mathbb{R}^D$ and $\mathbf{v}_i \in \mathbb{R}^d$, be a variable with the same block structure of $\mathbf{y} = [\mathbf{y}_1^\top \ \dots \ \mathbf{y}_N^\top]^\top \in \mathbb{R}^{N(D+d)}$, with $\mathbf{y}_i = [\mathbf{\Lambda}_i^\top \ \boldsymbol{\mu}_i^\top]^\top$ (as defined in (2.5)).

By using the definition of weighted proximal operator and the separability of both G^* and the norm function, we have

$$\begin{aligned}\mathbf{prox}_{\mathbf{D}_\alpha, G^*}(\mathbf{y}) &:= \operatorname*{argmin}_{\boldsymbol{\eta} \in \mathbb{R}^{N(D+d)}} \left\{ G^*(\boldsymbol{\eta}) + \frac{1}{2}\left\|\boldsymbol{\eta} - \mathbf{y}\right\|^2_{\mathbf{D}_\alpha^{-1}} \right\} \\ &= \operatorname*{argmin}_{\boldsymbol{\eta}} \left\{ \sum_{i=1}^N \left(g_i^*(\mathbf{v}_i) + \frac{1}{2\alpha_i}\|\mathbf{u}_i - \mathbf{\Lambda}_i\|^2 + \frac{1}{2\alpha_i}\|\mathbf{v}_i - \boldsymbol{\mu}_i\|^2 \right) \right\}.\end{aligned}$$

The minimization splits on each component $\boldsymbol{\eta}_i$ of $\boldsymbol{\eta}$, giving

$$\mathbf{prox}_{\mathbf{D}_\alpha, G^*}(\mathbf{y}) = \begin{bmatrix} \operatorname*{argmin}_{\mathbf{u}_1} \|u_1 - \mathbf{\Lambda}_1\|^2 \\ \operatorname*{argmin}_{\mathbf{v}_1} \left\{ g_1^*(\mathbf{v}_1) + \frac{1}{2\alpha_1}\|\mathbf{v}_1 - \boldsymbol{\mu}_1\|^2 \right\} \\ \vdots \\ \operatorname*{argmin}_{\mathbf{u}_N} \|\mathbf{u}_N - \mathbf{\Lambda}_N\|^2 \\ \operatorname*{argmin}_{\mathbf{v}_N} \left\{ g_N^*(\mathbf{v}_N) + \frac{1}{2\alpha_N}\|\mathbf{v}_N - \boldsymbol{\mu}_N\|^2 \right\} \end{bmatrix}$$

so that the proof follows from the definition of proximal operator. □

We are ready to show the convergence of the DDPG (Distributed Algorithm 1).

Theorem 2.4.12. *For each $i \in \{1, \dots, N\}$, let f_i be a proper, closed and strongly convex extended real-valued function with strong convexity parameter $\sigma_i > 0$, and let g_i be a proper convex extended real-valued function. Suppose that in Algorithm 1 the local step-size α_i is chosen such that $0 < \alpha_i \leq \frac{1}{NL_i}$, with L_i given by*

$$L_i = \sqrt{\frac{1}{\sigma_i^2} + \sum_{j \in \mathcal{N}_i} \left(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\right)^2}, \quad \forall\, i \in \{1, \dots, N\}. \tag{2.13}$$

Then the sequence $\mathbf{y}^t = [(\mathbf{y}_1^t)^\top \dots (\mathbf{y}_N^t)^\top]^\top$ generated by the DDPG (Algorithm 1) for all $t \geq 1$ satisfies

$$\Gamma(\mathbf{y}^t) - \Gamma(\mathbf{y}^\star) \leq \frac{\left\|\mathbf{y}_0 - \mathbf{y}^\star\right\|^2_{\mathbf{D}_\alpha^{-1}}}{2t},$$

where $\mathbf{y}^\star$ is any minimizer of (2.6), $\mathbf{y}_0 = [(\mathbf{y}_1^0)^\top \ \dots \ (\mathbf{y}_N^0)^\top]^\top$ is the initial condition and $\mathbf{D}_\alpha := \operatorname{diag}(\alpha_1, \dots, \alpha_N)$.

Proof. The theorem is proven in two stages. First, in Lemma 2.4.8 we have shown that problem (2.6) satisfies the assumptions of Theorem 2.4.5 and, thus, a (deterministic) weighted proximal gradient solves the problem. Thus, we need to show that DDPG (Distributed Algorithm 1) is a weighted proximal gradient scheme.

The weighted proximal gradient algorithm, described by (2.10), applied to problem (2.6), with $\mathbf{W} := \mathbf{D}_\alpha$, is given by

$$\mathbf{y}^{t+1} = \mathbf{prox}_{\mathbf{D}_\alpha, G^*}\Big(\mathbf{y}^t - \mathbf{D}_\alpha \nabla F^*\big(\mathbf{y}^t\big)\Big). \tag{2.14}$$

Now, by Lemma 2.4.8, L_i given in (2.13) is the Lipschitz constant of the i-th block of ∇F^*. Thus, using the hypothesis $\alpha_i \leq \frac{1}{NL_i}$, we can apply Theorem 2.4.5, which ensures convergence with a rate of $O(1/t)$ in objective value.

In order to disclose the distributed update rule, we first split (2.14) into two steps, i.e.,

$$\tilde{\mathbf{y}} = \mathbf{y}^t - \mathbf{D}_\alpha \nabla F^*\big(\mathbf{y}^t\big) \tag{2.15a}$$

$$\mathbf{y}^{t+1} = \mathbf{prox}_{\mathbf{D}_\alpha, G^*}\big(\,\tilde{\mathbf{y}}\,\big) \tag{2.15b}$$

and, then, compute explicitly each component of both the equations. Focusing on (2.15a) and considering that $\mathbf{D}_\alpha$ is diagonal, we can write the i-th block component of $\tilde{\mathbf{y}}$ as

$$\tilde{\mathbf{y}}_i = \begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_i \\ \tilde{\boldsymbol{\mu}}_i \end{bmatrix} = \mathbf{y}_i^t - \alpha_i \nabla_i F^*\big(\mathbf{y}^t\big),$$

where the explicit update of the block-component of $\tilde{\boldsymbol{\Lambda}}_i$ associated to neighbor j is

$$\tilde{\boldsymbol{\lambda}}_{ij} = \boldsymbol{\lambda}_{ij}^t - \alpha_i \frac{\partial F^*(\mathbf{y})}{\partial \boldsymbol{\lambda}_{ij}}\bigg|_{\mathbf{y}=\mathbf{y}^t} = \boldsymbol{\lambda}_{ij}^t + \alpha_i \Big[\nabla f_i^*\Big(-\sum_{k\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ik}^t - \boldsymbol{\lambda}_{ki}^t) - \boldsymbol{\mu}_i^t \Big) - \nabla f_j^*\Big(-\sum_{k\in\mathcal{N}_j} (\boldsymbol{\lambda}_{jk}^t - \boldsymbol{\lambda}_{kj}^t) - \boldsymbol{\mu}_j^t \Big) \Big] \tag{2.16}$$

and the explicit update of $\tilde{\mu}_i$ is

$$\tilde{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i^t - \alpha_i \frac{\partial F^*(\mathbf{y})}{\partial \boldsymbol{\mu}_i}\bigg|_{\mathbf{y}=\mathbf{y}^t} = \boldsymbol{\mu}_i^t + \alpha_i \nabla f_i^*\Big(-\sum_{k\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ik}^t - \boldsymbol{\lambda}_{ki}^t) - \boldsymbol{\mu}_i^t \Big). \tag{2.17}$$

Now, denoting

$$\mathbf{x}_i^t := \nabla f_i^*\Big(-\sum_{k\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ik}^t - \boldsymbol{\lambda}_{ki}^t) - \boldsymbol{\mu}_i^t \Big),$$

from Lemma 2.4.7 it holds

$$\mathbf{x}_i^t = \operatorname*{argmin}_{\mathbf{x}_i} \left\{ \mathbf{x}_i^\top \Big(\sum_{k\in\mathcal{N}_i} \Big(\boldsymbol{\lambda}_{ik}^t - \boldsymbol{\lambda}_{ki}^t \Big) + \boldsymbol{\mu}_i^t \Big) + f_i(\mathbf{x}_i) \right\}.$$

Thus, we can rewrite (2.16) and (2.17) in terms of $\mathbf{x}_i^t$ obtaining

$$\tilde{\boldsymbol{\lambda}}_{ij} = \boldsymbol{\lambda}_{ij}^t + \alpha_i \Big(\mathbf{x}_i^t - \mathbf{x}_j^t \Big)$$
$$\tilde{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i^t + \alpha_i \, \mathbf{x}_i^t.$$

Finally, the last step consists of applying the rule (2.15b) to $\tilde{\mathbf{y}}$. In order to highlight the distributed update, we rewrite (2.15b) in a component-wise fashion, i.e.,

$$\mathbf{y}^{t+1} = \begin{bmatrix} \boldsymbol{\Lambda}_1^{t+1} \\ \boldsymbol{\mu}_1^{t+1} \\ \vdots \\ \boldsymbol{\Lambda}_N^{t+1} \\ \boldsymbol{\mu}_N^{t+1} \end{bmatrix} = \mathbf{prox}_{\mathbf{D}_\alpha, G^*} \left(\begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_1 \\ \tilde{\boldsymbol{\mu}}_1 \\ \vdots \\ \tilde{\boldsymbol{\Lambda}}_N \\ \tilde{\boldsymbol{\mu}}_N \end{bmatrix} \right),$$

and applying Lemma 2.4.11 and Lemma 2.4.10 with $\varphi_i = g_i^*$, we obtain

$$\mathbf{y}^{t+1} = \begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_1 \\ \mathbf{prox}_{\alpha_1 g_1^*}(\tilde{\boldsymbol{\mu}}_1) \\ \vdots \\ \tilde{\boldsymbol{\Lambda}}_N \\ \mathbf{prox}_{\alpha_N g_N^*}(\tilde{\boldsymbol{\mu}}_N) \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_1 \\ \tilde{\boldsymbol{\mu}}_1 - \alpha_1 \mathbf{prox}_{\frac{1}{\alpha_1} g_1}\Big(\frac{\tilde{\boldsymbol{\mu}}_1}{\alpha_1} \Big) \\ \vdots \\ \tilde{\boldsymbol{\Lambda}}_N \\ \tilde{\boldsymbol{\mu}}_N - \alpha_N \mathbf{prox}_{\frac{1}{\alpha_N} g_N}\Big(\frac{\tilde{\boldsymbol{\mu}}_N}{\alpha_N} \Big) \end{bmatrix},$$

so that the proof follows. □

Remark 2.4.13 (Nesterov's acceleration)**.** *We can include a Nesterov's* extrapolation step *in the algorithm, which accelerates the algorithm (further details in* [67]*), attaining a faster* $O(1/t^2)$ *convergence rate in objective value. In order to implement the acceleration, each node needs to store a copy of the dual variables at the previous iteration. Thus, the update law in* (2.15) *would be changed in the following*

$$\begin{aligned}\tilde{\mathbf{y}} &= \mathbf{y}^t - \mathbf{D}_\alpha \nabla F^*(\mathbf{y}^t)\\ \hat{\mathbf{y}}^t &= \mathbf{prox}_{\mathbf{D}_\alpha, G^*}(\tilde{\mathbf{y}})\\ \mathbf{y}^{t+1} &= \hat{\mathbf{y}}^t + \theta_t\left(\hat{\mathbf{y}}^t - \hat{\mathbf{y}}^{t-1}\right).\end{aligned}$$

where θ_t *represents the Nesterov overshoot parameter.* □

2.4.3 Analysis of Node-Based Asynchronous Algorithm

In order to analyze the algorithm we start recalling some properties of i.i.d. exponential random variables. Let $\{i_t\}_{t\geq 0}$ with each $i_t \in \{1,\dots,N\}$, be the sequence identifying the generic t-th triggered node. Assumption 2.3.2 implies that i_t is an i.i.d. uniform process on the alphabet $\{1,\dots,N\}$. Each triggering will induce an iteration of the distributed optimization algorithm, so that t will be a universal, discrete time indicating the t-th iteration of the algorithm itself. As mentioned in Section 1.1, from an external perspective, the described local asynchronous updates result into an algorithmic evolution in which, at each iteration, only one node wakes up randomly, uniformly and independently from previous iterations. This variable t will be used in the statement and in the proof of Theorem 2.4.14. However, we want to stress that this iteration counter does not need to be known by the agents. Recall that we assumed that agents are able to complete their local computations and communications before a new timers triggers in the network.

Theorem 2.4.14. *For each* $i \in \{1,\dots,N\}$*, let* f_i *be a proper, closed and strongly convex extended real-valued function with strong convexity parameter* $\sigma_i > 0$*, and let* g_i *be a proper convex extended real-valued function. Suppose that in Algorithm 2 each local step-size* α_i *is chosen such that* $0 < \alpha_i \leq \frac{1}{L_i}$*, with*

$$L_i = \sqrt{\frac{1}{\sigma_i^2} + \sum_{j\in\mathcal{N}_i}\left(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\right)^2}, \quad \forall\, i \in \{1,\dots,N\}. \tag{2.18}$$

Then the sequence $\mathbf{y}^t = [(\mathbf{y}_1^t)^\top,\dots,(\mathbf{y}_N^t)^\top]^\top$ *generated by the A-DDPG (Algorithm 2) converges in objective value with high probability, i.e., for any* $\varepsilon \in \big(0, \Gamma(y_0)\big)$*, where* $\mathbf{y}_0 = [(\mathbf{y}_1^0)^\top,\dots,(\mathbf{y}_N^0)^\top]^\top$ *is the initial condition, and*

target confidence $0 < \rho < 1$, *there exists* $\bar{t}(\varepsilon, \rho) > 0$ *such that for all* $t \geq \bar{t}$ *it holds*

$$\Pr\Big(\Gamma(\mathbf{y}^t) - \Gamma^\star \leq \varepsilon\Big) \geq 1 - \rho,$$

where $\Gamma^\star$ *is the optimal cost of problem* (2.6).

Proof. To prove the theorem, we proceed in two steps. First, we show that we can apply the Uniform Coordinate Descent for Composite functions (Algorithm 4) to solve problem (2.6). Second, we show that, when applied to this problem, Algorithm 4 gives the iterates of our A-DDPG.

The first part follows immediately by Lemma 2.4.8, which asserts that problem (2.6) satisfies the assumptions of Theorem 2.4.6, so that Algorithm 4 solves it.

Next, we show that the two algorithms have the same update. First, by Lemma 2.4.8, L_i given in (2.18) is the Lipschitz constant of the i-th block of ∇F^*. Thus, in the rest of the proof, following [85], we set $\alpha_i = \frac{1}{L_i}$ (the maximum allowable value). Clearly the convergence is preserved if a smaller stepsize is used.

Consistently with the notation in Algorithm 4, let i_t denote the uniform-randomly selected index at iteration t. Thus,

$$T^{(i_t)}(\mathbf{y}^t) = \operatorname*{argmin}_{\mathbf{s}_{i_t} \in \mathbb{R}^{N_{i_t}}} \Big\{ V_{i_t}(\mathbf{y}^t, \mathbf{s}_{i_t}) \Big\}$$

defined in (2.11) can be written in terms of a proximal gradient update applied to the i_t-th block component of $\mathbf{y}$. In fact, by definition, for our function $\Gamma = F^* + G^*$, we have

$$T^{(i_t)}\big(\mathbf{y}^t\big) = \operatorname*{argmin}_{\mathbf{s} \in \mathbb{R}^{N_{i_t}}} \Big\{ \nabla_{i_t} F^*\big(\mathbf{y}^t\big)^\top \mathbf{s} + \frac{L_{i_t}}{2} \|\mathbf{s}\|^2 + g_{i_t}^*\big(\mathbf{y}_{i_t}^t + \mathbf{s}\big) \Big\}.$$

In order to apply the formal definition of a proximal gradient step, we add a constant term and introduce a change of variable given by $\bar{\mathbf{s}} := \mathbf{y}_{i_t}^t + \mathbf{s}$, obtaining

$$\begin{aligned} T^{(i_t)}\big(\mathbf{y}^t\big) = -\mathbf{y}_{i_t}^t + \operatorname*{argmin}_{\bar{\mathbf{s}} \in \mathbb{R}^{N_{i_t}}} \Big\{ & \mathbf{U}_{i_t}^\top F^*\big(\mathbf{y}^t\big) \\ & + \nabla_{i_t} F^*\big(\mathbf{y}^t\big)^\top (\bar{\mathbf{s}} - \mathbf{y}_{i_t}^t) + \frac{L_{i_t}}{2} \|\bar{\mathbf{s}} - \mathbf{y}_{i_t}^t\|^2 + g_{i_t}^*(\bar{\mathbf{s}}) \Big\}, \end{aligned}$$

which yields

$$T^{(i_t)}\big(\mathbf{y}^t\big) = -\mathbf{y}_{i_t}(t) + \mathbf{prox}_{\frac{1}{L_{i_t}} g_{i_t}^*}\Big(\mathbf{y}_{i_t}^t - \frac{1}{L_{i_t}} \nabla_{i_t} F^*\big(\mathbf{y}^t\big)\Big).$$

Thus, update (2.12) in fact changes only the component $\mathbf{y}_{i_t}$ of $\mathbf{y}$, which is updated as

$$\mathbf{y}_{i_t}^{t+1} = \mathbf{y}_{i_t}^t + T^{(i_t)}(\mathbf{y}^t) = \mathbf{prox}_{\frac{1}{L_{i_t}} g_{i_t}^*}\Big(\mathbf{y}_{i_t}^t - \frac{1}{L_{i_t}} \nabla_{i_t} F^*(\mathbf{y}^t)\Big), \tag{2.19}$$

while all the other ones remain unchanged, i.e., $\mathbf{y}_i^{t+1} = \mathbf{y}_i^t$ for all $i \in \{1, \dots, N\}$ with $i \neq i_t$.

Following the same steps as in the proof of Theorem 2.4.12, we split the update in (2.19) into a gradient and a proximal steps. The gradient step is given by

$$\tilde{\mathbf{y}}_{i_t} = \begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_{i_t} \\ \tilde{\boldsymbol{\mu}}_{i_t} \end{bmatrix} = \mathbf{y}_{i_t}^t - \frac{1}{L_{i_t}} \nabla_{i_t} F^*(\mathbf{y}^t)$$

where $\tilde{\boldsymbol{\Lambda}}_{i_t}$ and $\tilde{\boldsymbol{\mu}}_{i_t}$ are the same as in (2.16) and (2.17) respectively. The proximal operator step turns out to be

$$\mathbf{y}_{i_t}^{t+1} = \begin{bmatrix} \boldsymbol{\Lambda}_{i_t}^{t+1} \\ \boldsymbol{\mu}_{i_t}^{t+1} \end{bmatrix} = \mathbf{prox}_{\frac{1}{L_{i_t}} g_{i_t}^*} \left(\begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_{i_t} \\ \tilde{\boldsymbol{\mu}}_{i_t} \end{bmatrix} \right).$$

Applying Lemma 2.4.11 on the i_t-th block with $\alpha_{i_t} = 1/L_{i_t}$, we can rewrite (2.19) as

$$\begin{bmatrix} \boldsymbol{\Lambda}_{i_t}^{t+1} \\ \boldsymbol{\mu}_{i_t}^{t+1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\Lambda}}_{i_t} \\ \tilde{\boldsymbol{\mu}}_{i_t} - \frac{1}{L_{i_t}} \mathbf{prox}_{L_{i_t} g_{i_t}}\Big(L_{i_t} \tilde{\boldsymbol{\mu}}_{i_t}\Big) \end{bmatrix}, \tag{2.20}$$

where each component of $\tilde{\boldsymbol{\Lambda}}_{i_t}$ is given by

$$\tilde{\boldsymbol{\lambda}}_{i_t\, j} = \boldsymbol{\lambda}_{i_t\, j}^t + \frac{1}{L_{i_t}} \Big[\mathbf{x}_{i_t}^t - \mathbf{x}_j^t\Big],$$

and

$$\tilde{\boldsymbol{\mu}}_{i_t} = \boldsymbol{\mu}_{i_t}^t + \frac{1}{L_{i_t}}\, \mathbf{x}_{i_t}^t,$$

with

$$\mathbf{x}_i^t = \underset{\mathbf{x}_i}{\operatorname{argmin}} \left\{ \mathbf{x}_i^\top \left(\sum_{k \in \mathcal{N}_i} \Big(\boldsymbol{\lambda}_{ik}^t - \boldsymbol{\lambda}_{ki}^t\Big) + \boldsymbol{\mu}_i^t \right) + f_i(\mathbf{x}_i) \right\}.$$

Here we have used again the property from Lemma 2.4.7

$$\nabla f_i^* \Big(- \sum_{k \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ik}^t - \boldsymbol{\lambda}_{ki}^t) - \boldsymbol{\mu}_i^t \Big) = \mathbf{x}_i^t.$$

Now, from Assumption 2.3.2 a sequence of nodes i_t, $t = 1, 2...$, becomes active according to a uniform distribution, so that each node triggering can

be associated to an iteration of Algorithm 4 given by the update in (2.20). That is, only a node i_t is active in the network, which performs an update of its dual variables $\mathbf{y}_{i_t}$. In order to perform the local update, the selected node i_t needs to know the most updated information after the last triggering. As regards the neighbors' dual variables $\boldsymbol{\lambda}_{j\,i_t}$, $j \in \mathcal{N}_{i_t}$, they have been broadcast by each $j \in \mathcal{N}_{i_t}$ the last time it has become active. Regarding the primal variables $\mathbf{x}_j$, $j \in \mathcal{N}_{i_t}$, the situation is a little more tricky. Indeed, $\mathbf{x}_j$, $j \in \mathcal{N}_{i_t}$, may have changed in the past due to either j or one of its neighbors has become active. In both cases j has to broadcast to i_t its updated dual variable (either because it has become active or because it has received, in idle, an updated dual variable from one of its neighbors). Notice that we are implicitly using the assumption that these operations need a negligible amount of time. □

Remark 2.4.15. *Differently from the synchronous algorithm, in the asynchronous version nodes do not need to know the number of nodes, N, in order to set their local step-size. In fact, each node i can set its parameter α_i by only knowing the convexity parameters σ_i and σ_j, $j \in \mathcal{N}_i$.* □

Remark 2.4.16. *If a strongly convex, separable penalty term is added to the dual function $\Gamma = F^* + G^*$, then it becomes strongly convex, so that a stronger result from* [85, Theorem 7] *applies, i.e.,* linear *convergence with high probability is guaranteed. Note that strong convexity of the dual function Γ is obtained if the primal function has Lipschitz continuous gradient,* [37, Chapter X, Theorem 4.2.2]. □

2.4.4 Analysis of Edge-Based Asynchronous Algorithm

The convergence of Distributed Algorithm 3 relies essentially on the same arguments as in Theorem 2.4.14, but with a different block partition. In fact, we have to split the $\mathbf{y}$ variable into $|\mathcal{E}|$ blocks (with $|\mathcal{E}|$ the number of edges of $\mathcal{G}$). Notice that since the dual variables $\boldsymbol{\mu}_i$ are only N, they need to be associated to a subset of edges. Thus, the variable $\mathbf{y}$ is split in blocks $\mathbf{y}_{ij}$ given by $\mathbf{y}_{ij} = [\boldsymbol{\lambda}_{ij}\ \boldsymbol{\lambda}_{ji}\ \boldsymbol{\mu}_i]$ if $j = j_{\boldsymbol{\mu}_i}$ and $\mathbf{y}_{ij} = [\boldsymbol{\lambda}_{ij}\ \boldsymbol{\lambda}_{ji}]$ otherwise. This is why in the algorithm $\boldsymbol{\mu}_i$ is updated only when neighbor $j_{\boldsymbol{\mu}_i}$ becomes active.

2.5 Framework Flexibility and Numerical Computations

In this section we illustrate the flexibility of our algorithmic framework which is able to handle both local constraints and regularizers. Then we show a numerical experiment to test our distributed algorithms.

2.5.1 Flexibility of the Algorithmic Framework

In Section 1.2.1 we showed how an optimization problem with local constraints, i.e.,

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_{i=1}^{N} h_i(\mathbf{x}) \\ \text{subj. to } & \mathbf{x} \in \bigcap_{i=1}^{N} X_i, \end{aligned}$$

can be cast in the form (2.1), i.e., by setting $f_i(\mathbf{x}_i) = h_i(\mathbf{x})$ and $g_i(\mathbf{x}_i) = I_{X_i}(\mathbf{x}_i)$ for all $i \in \{1, \dots, N\}$. Treating the local constraints in this way, the local optimization step (2.7) (or (2.8)) turns out to be a local *unconstrained* minimization, while the local feasibility is entrusted to the proximal operator of g_i. In fact, the proximal operator of a convex indicator function reduces to the standard Euclidean projection, i.e.,

$$\mathbf{prox}_{I_X}(\mathbf{v}) = \operatorname*{argmin}_{\mathbf{x}} \left\{ I_X(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}\|^2 \right\} = \Pi_X(\mathbf{v}).$$

Remark 2.5.1. *When considering quadratic costs h_i, we can benefit greatly from a numerical point of view. In fact, an unconstrained quadratic program can be solved via efficient methods, which often result in division-free algorithms (possibly after some off-line precomputations), and can be implemented in fixed-point arithmetic, see* [79] *for further details.* □

An attractive feature of our set-up is that one can conveniently decide how to rewrite the local constraints. In the formulation above, we suggested to include the local constraint X_i into g_i. But it is also reasonable to include the constraint into f_i, by considering the indicator function in its definition, i.e., define

$$f_i(\mathbf{x}) := \begin{cases} h_i(\mathbf{x}) & \text{if } \mathbf{x} \in X_i \\ +\infty & \text{otherwise,} \end{cases} \tag{2.21}$$

and, thus, have g_i identically zero (still convex). This strategy results into an algorithm which is basically an asynchronous distributed dual decomposition algorithm. Notice that with this choice recursive feasibility, i.e., $\mathbf{x}_i^t \in X_i$ for all t, is obtained provided that the local algorithm solves the minimization in an interior point fashion.

Between these two extreme scenarios one could also consider other possibilities. Indeed, it could be the case that one can benefit from splitting each local constraint X_i into two distinct contributions, i.e., $X_i = Y_i \cap Z_i$. In this way the indicator function of Y_i (e.g., the positive orthant) could be included into f_i, allowing for a simpler constrained local minimization

step, while the other constraint could be mapped into the second term as $g_i(\mathbf{x}) = I_{Z_i}(\mathbf{x})$.

The choice in (2.21) leads to a simple observation: leaving the g_i equal to zero seems to be a waste of a degree of freedom that could be differently exploited, e.g., by introducing a regularization term. We next described an example where we can exploit this degree of freedom. Let us consider a special instance of problem (2.1), namely a constrained and regularized problem as the constrained LASSO (see also Section 1.3)

$$\min_{\mathtt{lb} \preceq \mathbf{x} \preceq \mathtt{ub}} \sum_{i=1}^{N} \|\mathbf{D}_i \mathbf{x} - \mathbf{b}_i\|_2^2 + \gamma \|\mathbf{x}\|_1, \tag{2.22}$$

where $\mathbf{x}$ is the decision variable, and $\mathbf{D}_i$ and $\mathbf{b}_i$ represent respectively the data matrix and the labels associated with examples assigned to node i. The inequalities $\mathtt{lb} \preceq \mathbf{x} \preceq \mathtt{ub}$ are meant component-wise.

As discussed before, the flexibility of our algorithmic framework allows us to handle, together with local constraints, also a regularization cost through the g_i. The first way to map the problem in our set-up is by defining

$$f_i(\mathbf{x}) := \begin{cases} \|\mathbf{D}_i \mathbf{x} - \mathbf{b}_i\|^2 & \text{if } \mathbf{x} \in X_i \\ +\infty & \text{otherwise} \end{cases} \tag{2.23}$$

and setting $g_i(\mathbf{x}) := \frac{\gamma}{N} \|\mathbf{x}\|_1$.

The proximal operator of the ℓ_1-norm admits an analytic solution which is well known as soft thresholding operator. When applied to a vector $\mathbf{v} \in \mathbb{R}^d$ (with ℓ-th component v_ℓ), it gives a vector in $\mathbb{R}^d$ whose ℓ-th component is

$$\left(\mathbf{prox}_{\gamma \|\cdot\|_1}(\mathbf{v})\right)_\ell = \begin{cases} v_\ell - \gamma, & v_\ell > \gamma \\ 0, & |v_\ell| \leq \gamma \\ v_\ell + \gamma, & v_\ell < -\gamma \end{cases} \tag{2.24}$$

i.e., it thresholds the components of v which are in modulus greater then γ, see, e.g., [6, 81]. In Figure 2.1 a graphical representation of the soft shareholding operator is given.

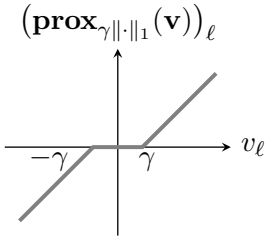


Figure 2.1: Soft-thresholding operator.

Alternatively, we may include both the constraint X_i and the regularization term into the g_i as

$$g_i(\mathbf{x}) := \begin{cases} \lambda\|\mathbf{x}\|_1 & \text{if } \mathbf{x} \in X_i \\ +\infty & \text{otherwise.} \end{cases}$$

Thus, we obtain an unconstrained local minimization at each node. This choice is particularly appealing when the constraint X_i is a box, i.e., $X_i = \{\mathbf{v} \in \mathbb{R}^d \mid \mathtt{lb}_\ell \leq v_\ell \leq \mathtt{ub}_\ell \text{ for all } \ell \in \{1, \dots, d\}\}$. In this case the proximal operator of g_i becomes a *saturated* version of the soft-thresholding operator, as depicted in Figure 2.2.

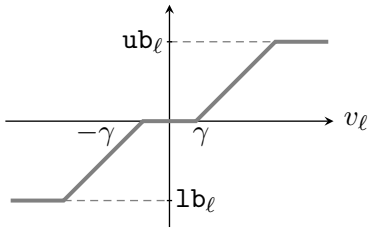


Figure 2.2: Saturated soft-thresholding operator.

2.5.2 Application to the Constrained LASSO

In this section we test the proposed distributed algorithms on a numerical instance of a constrained LASSO optimization problem (2.22).

The decision variable $\mathbf{x} \in \mathbb{R}^3$, $\mathbf{D}_i \in \mathbb{R}^{150\times 3}$ and $\mathbf{b}_i \in \mathbb{R}^{150}$ for all $i \in \{1, \dots, N\}$. We randomly generate the LASSO data following the idea suggested in [81]. Each entry of $\mathbf{D}_i$ is $\sim \mathcal{N}(0, 1)$, and each $\mathbf{b}_i$ is generated by perturbing a "true" solution $\mathbf{x}_{\text{true}}$ (which has around a half nonzero entries) with an additive noise $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, 10^{-2}\mathbf{I})$. Then the matrix $\mathbf{D}_i$ and the vector $\mathbf{b}_i$ are normalized with respect to the number of local samples at each node. The box bounds are set to $\mathtt{lb} = \begin{bmatrix}-0.8 & -0.8 & -0.8\end{bmatrix}^\top$ and $\mathtt{ub} = \begin{bmatrix}0.8 & 0.8 & 0.8\end{bmatrix}^\top$, while the regularization parameter is $\gamma = 0.1$.

To match the problem with our distributed framework, we introduce copies $\mathbf{x}_i$ of the decision variable $\mathbf{x}$. Consistently, we define the local functions f_i as the least-square costs in (2.23), where each X_i is the box defined by $\mathtt{lb}$ and $\mathtt{ub}$. We let each g_i be the ℓ_1-norm regularization term with local parameter γ/N. We initialize to zero the dual variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$, and $\boldsymbol{\mu}_i$ for all $i \in \{1, \dots, N\}$, and use as step-sizes $\alpha_i = L_i$, where L_i has the expression in (2.13), with σ_i being the smallest eigenvalue of $\mathbf{D}_i^\top \mathbf{D}_i$.

We consider an undirected connected Erdős-Rényi graph $\mathcal{G}$, with parameter 0.2, connecting $N = 50$ nodes.

We run both the synchronous and the asynchronous algorithms over this underlying graph and we stop them if the difference between the current dual cost and the optimal value drops below the threshold of 10^{-6}.

Figure 2.3 shows the difference between the dual cost at each iteration t and the optimal value, $\Gamma(\mathbf{y}^t) - \Gamma^\star$, in a logarithmic scale. In particular, the rates of convergence of the synchronous (left) and asynchronous (right) algorithms are shown. For the asynchronous algorithm, we normalize the iteration counter t with respect the number of agents N.

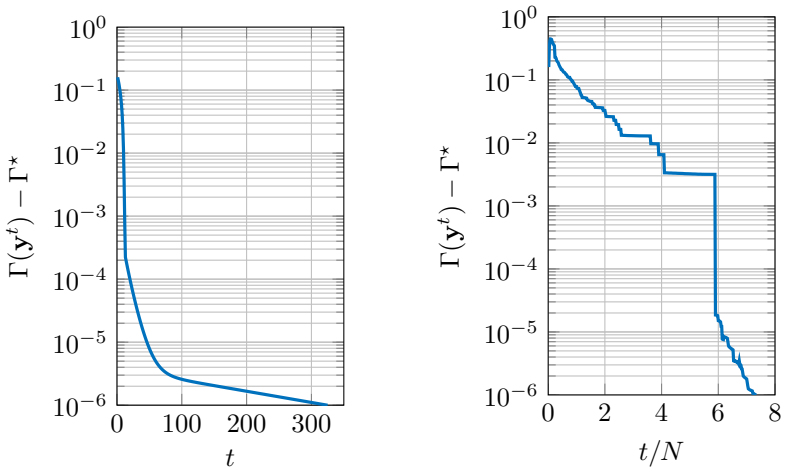


Figure 2.3: Evolution of the cost error, in logarithmic scale, for the synchronous (left) and node-based asynchronous (right) distributed algorithms.

Then we concentrate on the asynchronous algorithm. In Figure 2.4 we plot the evolution of the (three) components of the primal variables, $\mathbf{x}_i^t$, $i \in \{1, \dots, N\}$. The horizontal dotted lines represent the optimal solution. It is worth noting that the optimal solution has a first component slightly below the constraint boundary, a second component equal to zero, and a third component on the constraint boundary. This optimal solution can be intuitively explained by the "simultaneous influence" of the box constraints (which restrict the set of admissible values) and the regularization term (which enforces sparsity of the vector $\mathbf{x}$). In the picture inset, the first iterations for a subset of selected nodes are highlighted, in order to better show the transient, piece-wise constant behavior due to the gossip update and the effect of the box constraint on each component.

Specifically, for the first component, it can be seen how the temporary solution of some agents hits the boundary in some iterations (e.g., one of them hits the boundary from iteration 50 to iteration 100) and then converges to the (feasible) optimal value. The second components are always inside the box constraint and converge to zero, while the third components start inside the box and then in a finite number of iterations hit the boundary.

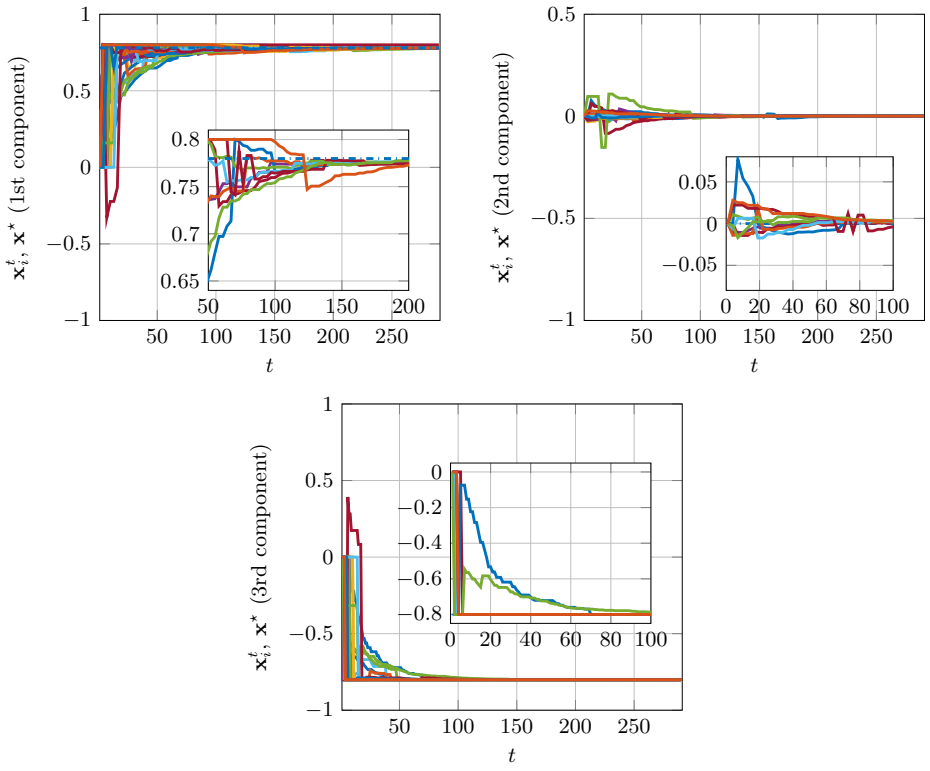


Figure 2.4: Evolution of the three components of primal variables $\mathbf{x}_i^t$, $i \in \{1, \ldots, N\}$, for the node-based A-DDPG.

Chapter 3

Asynchronous Methods for Partitioned Optimization

In this chapter we consider partitioned big-data optimization problems that we introduced in Section 1.2.2. Specifically, we propose and analyze two alternative asynchronous approaches to solve partitioned problems. The first method is based on duality and extends the methodology proposed in Chapter 2 to the partitioned set-up. In this way, we design an asynchronous distributed algorithm, based on a block-coordinate proximal gradient applied to a dual formulation of the problem, to solve strongly convex problems with costs and constraints depending only on a small portion of the huge decision variable In the second part, we propose a primal approach that allows us to design a distributed optimization algorithm for composite partitioned nonconvex costs where the second term depends on the local variable. The results of this chapter are based on [68, 69].

3.1 Literature Review

The partitioned set-up has been introduced in [32] where a distributed ADMM-based algorithm is proposed. In [22] some concrete motivating scenarios are described for the same set-up and a dual decomposition algorithm is proposed. In both the above references the algorithms are designed for a synchronous network with a fixed communication graph. In [58], an analogous problem formulation is considered within a parallel context. The authors propose a coordinate descent method and derive its convergence rate. In [94] the authors propose a distributed algorithm for a partitioned quadratic program under lossy communication. A distributed ADMM-based algorithm with applications in MPC is proposed in [56] to deal with an unconstrained optimization problem with local domains which is related to the partitioned set-up.

Usually, distributed approaches need a common clock (e.g., because a diminishing (time-varying) step-size is used). We want to avoid this limita-

tion and, to this end, we adopt an asynchronous, event-triggered communication protocol based on local and independent timers. (See Section 1.1). A Newton-Raphson consensus strategy is proposed in [105] to solve unconstrained, convex optimization problems under asynchronous, symmetric gossip communications. In [30] a self-triggered communication protocol is considered. Based on an error condition a distributed, continuous-time algorithm is developed. In [99] an asynchronous ADMM-based distributed method is proposed for a separable, constrained optimization problem with a convergence rate $O(1/t)$. A distributed, asynchronous algorithm for constrained optimization based on random projections is proposed in [48].

The analysis of the asynchronous distributed algorithms we design in this chapter are based on a (randomized) block-coordinate descent method. In [66] the coordinate method for huge scale optimization has been introduced. This powerful approach has been extended to deal with (convex) composite objective functions and parallel scenarios, see [58, 85, 86]. In [57] a coordinate approach to solve linearly constrained problems has been proposed. Using a coordinate ADMM-based approach, in [13] a distributed, asynchronous algorithm is developed.

In the second part of the chapter we investigate nonconvex problems. In [83], the authors extend the coordinate approach to large-scale nonconvex optimization proving the rate of convergence of their algorithms. A parallel algorithm based on local strongly convex approximations is exploited in [33] to cope with nonconvex optimization problems. The latter approach has been extended to a distributed context in [29]. In [15], the authors proposed an auction-based distributed algorithm for nonconvex optimization.

3.2 Partitioned Dual Decomposition for Distributed Optimization

In this section we describe a dual approach to solve partitioned optimization problems introduced in Section 1.2.2. Specifically, we consider a network of agents aiming at solving a structured optimization problem in a distributed way. The nodes, $\{1, \dots, N\}$, interact according to a fixed connected, undirected graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$. Notice that in this set-up, the graph $\mathcal{G}$ is strictly related to the structure of the optimization problem. In fact, the problem we formally aim at solving is in the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^d} \ & \sum_{i=1}^{N} f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \\ \text{subj. to } & (\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \in X_i, \qquad i \in \{1, \dots, N\}, \end{aligned} \tag{3.1}$$

where we recall that the notation $f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})$ means that f_i is, in fact, a function of $\mathbf{x}_i$ and $\mathbf{x}_j$, $j \in \mathcal{N}_i$ only, and the notation $(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \in X_i$

means that the constraint set X_i involves only the variables $\mathbf{x}_i$ and $\mathbf{x}_j$, $j \in \mathcal{N}_i$. We stress that the constraint sets X_i can involve all (neighboring) variables $(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})$ of agent i and not just $\mathbf{x}_i$. This apparently minor feature in fact adds much more generality to the problem and introduces important significant challenges. For example, the primal approach investigated in Section 3.3 does not apply to this case since it can handle only constraints in the form $\mathbf{x}_i \in X_i$.

The following assumptions on problem 3.1 will be used in this section.

Assumption 3.2.1. *For all $i \in \{1,\dots,N\}$, the function $f_i : \mathbb{R}^{\sum_{j\in\mathcal{N}_i\cup\{i\}} n_j} \to \mathbb{R}$ is strongly convex with parameter $\sigma_i > 0$.* □

Assumption 3.2.2. *The constraint sets $X_i \subseteq \mathbb{R}^{\sum_{j\in\mathcal{N}_i\cup\{i\}} n_j}$, $i \in \{1,\dots,N\}$ are nonempty convex and compact.* □

Assumption 3.2.3 (Constraint qualification)**.** *The intersection of the relative interior of the sets X_i, $i \in \{1,\dots,N\}$, is non-empty.* □

Under Assumptions 3.2.1 and 3.2.2 problem (3.1) is feasible and admits a unique optimal solution $f^\star$ attained at some $\mathbf{x}^\star \in \mathbb{R}^d$. Assumption 3.2.3 is a standard requirement to guarantee that a dual approach will enjoy the strong duality property.

3.2.1 Dual Decomposition for the Partitioned Set-up

In order to introduce our distributed dual algorithms, we derive a partitioned dual decomposition scheme by introducing suitable copies of the decision variables. Because of the structure of f_i and X_i, $i \in \{1,\dots,N\}$, the classical dual decomposition is considerably redundant. The idea is to exploit the partitioned structure to limit the range of equivalences among the auxiliary variables, and, in turn, their diffusion over the network. Once we create copies of the vector $\mathbf{x} \in \mathbb{R}^d$, we enforce each state $\mathbf{x}_i \in \mathbb{R}^{n_i}$ to be identical only for the neighboring nodes $j \in \mathcal{N}_i \cup \{i\}$ which use this information. Formally, we reformulate problem (3.1) as

$$\begin{aligned} \min \ & \sum_{i=1}^{N} f_i\big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ \text{subj. to } & \big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \in X_i, \qquad i \in \{1,\dots,N\}, \\ & \mathbf{x}_i^{(i)} = \mathbf{x}_i^{(j)}, \qquad (i,j) \in \mathcal{E}, \\ & \mathbf{x}_j^{(i)} = \mathbf{x}_j^{(j)}, \qquad (i,j) \in \mathcal{E}, \end{aligned} \tag{3.2}$$

where the symbol $\mathbf{x}_i^{(j)}$ denotes the copy of state $\mathbf{x}_i$ stored in memory of node j. Notice that connected nature of the graph $\mathcal{G}$ ensures equivalence between (3.1) and (3.2).

As an example, in Figure 3.1 we visualize the partitioned set-up for a path graph of $N = 4$ nodes. Along i-th column, we show the coupling due to the local cost f_i and the local constraint X_i, which involves only the states handled by node i, i.e., $\mathbf{x}_i^{(i)}$ and $\mathbf{x}_j^{(i)}$ with $j \in \mathcal{N}_i$. Along the i-th row, we show the coupling due to copies $\mathbf{x}_i^{(j)}$, $j \in \mathcal{N}_i$, of the variable $\mathbf{x}_i$.

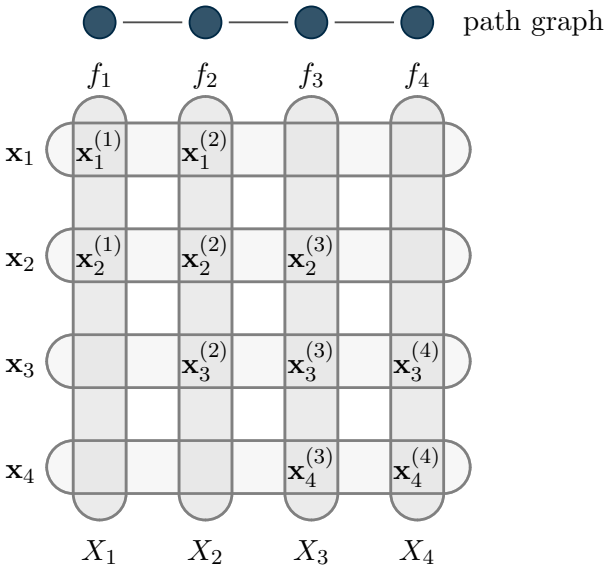


Figure 3.1: Partitioned optimization problem over a path graph of $N = 4$ nodes.

We point out that, for each pair of agents i and j, a constraint $x_i^{(i)} = x_i^{(j)}$ appears two times. This redundant formulation is not accidental, but plays an important role in exploiting the partitioned structure of the proposed algorithms.

Next, we introduce an aggregate notation for the copies, which allows us to be more compact in the derivation of the algorithms and their analysis. We denote by

$$\mathbf{y}^{(i)} := \big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \tag{3.3}$$

the set of local variables of node i, arranged as a column vector in $\mathbb{R}^{\sum_{j\in\mathcal{N}_i\cup\{i\}} n_j}$. In this way we can write equivalently

$$f_i\big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) = f_i\big(\mathbf{y}^{(i)}\big) \quad \text{and} \quad \mathbf{y}^{(i)} \in X_i.$$

To tackle problem (3.2) in a distributed way, we start by deriving its

dual problem. The partial Lagrangian for problem (3.2) is given by

$$\begin{aligned}\mathcal{L}(\mathbf{X},\mathbf{\Lambda}) = \sum_{i=1}^{N} \Bigg(& f_i\big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ & + \sum_{j\in\mathcal{N}_i} \Big(\boldsymbol{\lambda}_i^{(i,j)\top}(\mathbf{x}_i^{(i)} - \mathbf{x}_i^{(j)}) + \boldsymbol{\lambda}_j^{(i,j)\top}(\mathbf{x}_j^{(i)} - \mathbf{x}_j^{(j)})\Big)\Bigg),\end{aligned} \tag{3.4}$$

where $\mathbf{X}$ stacks all the (primal) optimization variables in the network, while $\mathbf{\Lambda}$ denotes the stack of dual variables, i.e.,

$$\mathbf{\Lambda} = \big[\mathbf{\Lambda}_1^\top, \ldots, \mathbf{\Lambda}_N^\top\big]^\top,$$

with block $\mathbf{\Lambda}_i := [\{\boldsymbol{\lambda}_i^{(i,j)}\}_{j\in\mathcal{N}_i}, \{\boldsymbol{\lambda}_j^{(i,j)}\}_{j\in\mathcal{N}_i}]$, $i \in \{1,\ldots,N\}$.

By exploiting the undirected nature and the connectivity of graph $\mathcal{G}$, the Lagrangian (3.4) can be rewritten as

$$\begin{aligned}\mathcal{L}(\mathbf{X},\mathbf{\Lambda}) = \sum_{i=1}^{N} \Bigg(& f_i\big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ & + \mathbf{x}_i^{(i)\top} \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_i^{(i,j)} - \boldsymbol{\lambda}_i^{(j,i)}) + \sum_{j\in\mathcal{N}_i} \mathbf{x}_j^{(i)\top}\big(\boldsymbol{\lambda}_j^{(i,j)} - \boldsymbol{\lambda}_j^{(j,i)}\big)\Bigg)\end{aligned} \tag{3.5}$$

which is separable with respect to $\mathbf{y}^{(i)}$, $i \in \{1,\ldots,N\}$.

Remark 3.2.4. *It is worth noting that we have not dualized the local constraints $(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}) \in X_i$ (thus the notion of partial Lagrangian) since each of them will be handled by the agents in their local optimization problem.* □

The dual function of (3.2) is obtained by minimizing the Lagrangian with respect to the primal variables, which gives

$$q(\mathbf{\Lambda}) = \min_{\mathbf{X}\in X_1\times\cdots\times X_N} \mathcal{L}(\mathbf{X},\mathbf{\Lambda}) = \sum_{i=1}^{N} q_i\Big(\big\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_i^{(j,i)}, \boldsymbol{\lambda}_j^{(i,j)}, \boldsymbol{\lambda}_j^{(j,i)}\big\}_{j\in\mathcal{N}_i}\Big)$$

with

$$\begin{aligned} q_i\Big(\big\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_i^{(j,i)}, \boldsymbol{\lambda}_j^{(i,j)}, \boldsymbol{\lambda}_j^{(j,i)}\big\}_{j\in\mathcal{N}_i}\Big) = & \\ \min_{\big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big)\in X_i} \Big(& f_i\big(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ & + \mathbf{x}_i^{(i)\top} \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_i^{(i,j)} - \boldsymbol{\lambda}_i^{(j,i)}) + \sum_{j\in\mathcal{N}_i} \mathbf{x}_j^{(i)\top}(\boldsymbol{\lambda}_j^{(i,j)} - \boldsymbol{\lambda}_j^{(j,i)})\Big).\end{aligned} \tag{3.6}$$

Notice that, since each X_i is compact and nonempty, the minimum in (3.6) is (uniquely) attained, so that q_i is always finite. Thus, the dual problem of (3.2) is the following unconstrained optimization problem

$$\max_{\mathbf{\Lambda}} \sum_{i=1}^{N} q_i\Big(\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_i^{(j,i)}, \boldsymbol{\lambda}_j^{(i,j)}, \boldsymbol{\lambda}_j^{(j,i)}\}_{j\in\mathcal{N}_i}\Big). \tag{3.7}$$

Remark 3.2.5. *Let $\varphi : \mathbb{R}^d \to \mathbb{R}\cup\{+\infty\}$, its conjugate function $\varphi^* : \mathbb{R}^d \to \mathbb{R}$ is defined as*

$$\varphi^*(\mathbf{z}) := \sup_{\mathbf{x}} \big(\mathbf{z}^\top\mathbf{x} - \varphi(\mathbf{x})\big).$$

Then,

$$q_i\Big(\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_i^{(j,i)}, \boldsymbol{\lambda}_j^{(i,j)}, \boldsymbol{\lambda}_j^{(j,i)}\}_{j\in\mathcal{N}_i}\Big) = \\ - f_i^*\Big(\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_i^{(i,j)} - \boldsymbol{\lambda}_i^{(j,i)}), \big\{(\boldsymbol{\lambda}_j^{(i,j)} - \boldsymbol{\lambda}_j^{(j,i)})\big\}_{j\in\mathcal{N}_i}\Big),$$

with f_i^* being the conjugate function of f_i. □

Remark 3.2.6. *It is worth noting that each q_i does not depend on the entire set of dual variables $\mathbf{\Lambda}$, but it itself exhibits a sparse structure, i.e., it is a function of the dual variables of the neighbors $\mathcal{N}_i$ only.* □

Differently from the other chapters in this thesis, here we use a slightly different notation to describe the algorithmic evolution and the agents' states. In particular, the iteration counter t appears in the bracket and it is not as a superscript. Moreover, the subscript will denote the component of the variables rather than the owner. This heavy notation is due to the partitioned structure of the variables.

3.2.2 Synchronous Partitioned Dual Decomposition (PDD)

With the dual problem in hand, a gradient algorithm on the dual problem, [9, Chapter 6], can be applied. This results into a minimization on the primal variables and a linear update on the dual variables. As we will show in the analysis, this gives rise to the PDD distributed algorithm, which is formally stated, from the perspective of node i, in the following table.

We point out that each node $i \in \{1,\dots,N\}$ stores and updates the primal variables $\mathbf{x}_i^{(i)}$ and $\mathbf{x}_j^{(i)}$, $j \in \mathcal{N}_i$, and the dual variables $\boldsymbol{\lambda}_i^{(i,j)}$ and $\boldsymbol{\lambda}_j^{(i,j)}$, $j \in \mathcal{N}_i$.

Distributed Algorithm 5 Partitioned Dual Decomposition (PDD)

Processor states: $(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i})$ and $\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_j^{(i,j)}\}_{j\in\mathcal{N}_i}$

Evolution:

FOR: $t = 1, 2, \ldots$ DO

Compute and broadcast primal variables

$$\begin{aligned}\big(\mathbf{x}_i^{(i)}(t+1), \{\mathbf{x}_j^{(i)}(t+1)\}_{j\in\mathcal{N}_i}\big) = \underset{(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})\in X_i}{\operatorname{argmin}} \Bigg(f_i\big(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}\big) \\ + \mathbf{x}_i^\top \sum_{j\in\mathcal{N}_i} \Big(\boldsymbol{\lambda}_i^{(i,j)}(t) - \boldsymbol{\lambda}_i^{(j,i)}(t)\Big) + \sum_{j\in\mathcal{N}_i} \mathbf{x}_j^\top \Big(\boldsymbol{\lambda}_j^{(i,j)}(t) - \boldsymbol{\lambda}_j^{(j,i)}(t)\Big)\Bigg).\end{aligned} \tag{3.8}$$

Update and broadcast dual variables via

$$\begin{aligned}\boldsymbol{\lambda}_i^{(i,j)}(t+1) &= \boldsymbol{\lambda}_i^{(i,j)}(t) + \alpha_i\Big(\mathbf{x}_i^{(i)}(t+1) - \mathbf{x}_i^{(j)}(t+1)\Big) \\ \boldsymbol{\lambda}_j^{(i,j)}(t+1) &= \boldsymbol{\lambda}_j^{(i,j)}(t) + \alpha_i\Big(\mathbf{x}_j^{(i)}(t+1) - \mathbf{x}_j^{(j)}(t+1)\Big)\end{aligned} \tag{3.9}$$

for all $j \in \mathcal{N}_i$.

Before studying the convergence properties of the proposed algorithm, let us comment on its scalability. We observe that each node has to keep in memory the set of variables $\mathbf{x}_i^{(i)}$, $\{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}$, $\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_j^{(i,j)}\}_{j\in\mathcal{N}_i}$, namely a number of variables equal to $1 + 3|\mathcal{N}_i|$. Second, the step-sizes α_i, $i \in \{1, \ldots, N\}$ are constant, local and can be initialized via local computations.

Remark 3.2.7. *Notice that, differently from existing dual decomposition schemes, the PDD algorithm does not enforce any symmetry in the dual variables, i.e., in general* $\boldsymbol{\lambda}_i^{(i,j)}(t) \neq -\boldsymbol{\lambda}_i^{(j,i)}(t)$. *The symmetry, although not necessary, can be imposed if the agents select a common step-size* $\alpha_i = \alpha$, *for all* $i \in \{1, \ldots, N\}$, *and properly initialize their dual variables. As a consequence, the algorithm can be simplified to have only one communication round to perform both the local minimization and the ascent.* □

The convergence properties of PDD (Distributed Algorithm 5) are established in the following theorem.

Theorem 3.2.8. *Let Assumptions 3.2.1, 3.2.2 and 3.2.3 hold true and assume the step-sizes* α_i, $i \in \{1, \ldots, N\}$, *to be constant and such that* $0 < \alpha_i \leq \frac{1}{NL_i}$, *with*

$$L_i = \sqrt{2\sum_{j\in\mathcal{N}_i}\Big(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\Big)^2}, \quad \forall\, i \in \{1, \ldots, N\}. \tag{3.10}$$

Then, the sequence $\{\boldsymbol{\Lambda}_1(t), \dots, \boldsymbol{\Lambda}_N(t)\}$ *generated by PDD (Distributed Algorithm 5) converges in objective value to the optimal cost* $f^\star$ *of problem* (3.1). *Moreover, let* $\mathbf{x}^\star = (\mathbf{x}_1^{\star\top}, \dots, \mathbf{x}_N^{\star\top})^\top$ *be the unique optimal solution of* (3.1), *then each primal sequence* $\mathbf{x}_i^{(i)}(t)$ *generated by PDD is such that*

$$\lim_{t\to\infty} \|\mathbf{x}_i^{(k)}(t) - \mathbf{x}_i^\star\| = 0,$$

for all $i \in \{1, \dots, N\}$ *and* $k \in \mathcal{N}_i \cup \{i\}$.

Proof. We structure the proof of the first statement in three parts in which we show that: (i) the dual gradient has a block structure and smoothness, (ii) the distributed algorithm implements a diagonally-scaled gradient method, and (iii) strong duality holds. First, consider the dual problem (3.7) and a block partitioning of dual variables $\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_1, \dots, \boldsymbol{\Lambda}_N]$, with

$$\boldsymbol{\Lambda}_i := \left(\{\boldsymbol{\lambda}_i^{(i,j)}\}_{j\in\mathcal{N}_i}, \{\boldsymbol{\lambda}_j^{(i,j)} j \in \mathcal{N}_i\}\right) \tag{3.11}$$

representing the local variables of node i, for all $i \in \{1, \dots, N\}$. Under Assumption 3.2.1, the dual function $q(\boldsymbol{\Lambda})$ is guaranteed to have block-coordinate Lipschitz continuous gradient $\nabla q(\boldsymbol{\Lambda})$ with block constants L_i, $i \in \{1, \dots, N\}$, given in (3.10). In fact, we can explicitly compute the components of $\nabla q(\boldsymbol{\Lambda})$ associated to each block $\boldsymbol{\Lambda}_i$, denoted hereafter as $\nabla_{\boldsymbol{\Lambda}_i} q(\boldsymbol{\Lambda})$, by using the chain rule of derivation and the conjugate function notation. We have that

$$\begin{aligned} \frac{\partial q(\boldsymbol{\Lambda})}{\partial \boldsymbol{\lambda}_i^{(i,j)}} &= (\nabla f_i^*)_i - (\nabla f_j^*)_i,\ j \in \mathcal{N}_i \\ \frac{\partial q(\boldsymbol{\Lambda})}{\partial \boldsymbol{\lambda}_j^{(i,j)}} &= (\nabla f_i^*)_j - (\nabla f_j^*)_j,\ j \in \mathcal{N}_i, \end{aligned} \tag{3.12}$$

where $(\nabla f_i^*)_i$ denotes the i-th component of ∇f_i^* and we omit the argument of $\nabla f_i^*(\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_i^{(i,j)} - \boldsymbol{\lambda}_i^{(j,i)}), \{(\boldsymbol{\lambda}_j^{(i,j)} - \boldsymbol{\lambda}_j^{(j,i)})\}_{j\in\mathcal{N}_i})$ to take light the notation.

Since for all $i \in \{1, \dots, N\}$, each f_i is a strongly convex function, then the gradient of its conjugate function ∇f_i^* is Lipschitz continuous with constant $1/\sigma_i$, [37, Chapter X, Theorem 4.2.2]. By considering the Euclidean 2-norm, in light of (3.12) and by simple algebraic manipulation, we can conclude that also $\nabla_{\boldsymbol{\Lambda}_i} q(\boldsymbol{\Lambda})$ is Lipschitz continuous with constant

$$L_i = \sqrt{\sum_{j\in\mathcal{N}_i} \left(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\right)^2 + \sum_{j\in\mathcal{N}_i} \left(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\right)^2},$$

which matches (3.10).

Second, we show that our PDD distributed algorithm implements a scaled gradient ascent method to solve problem (3.7). Consider a diagonal positive definite matrix defined as

$$\mathbf{W} = \operatorname{diag}(\alpha_1, \dots, \alpha_N) \preceq \operatorname{diag}\left(\frac{1}{NL_1}, \dots, \frac{1}{NL_N}\right).$$

Formally, the scaled gradient ascent method can be written as

$$\mathbf{\Lambda}(t+1) = \mathbf{\Lambda}(t) + \mathbf{W}\nabla q(\mathbf{\Lambda}(t)), \tag{3.13}$$

where t denotes the iteration counter. Since each entry of the scaling matrix satisfies $0 < \alpha_i \leq \frac{1}{NL_i}$ for all $i \in \{1, \dots, N\}$, then the following condition holds [86, Theorem 8]

$$q(\mathbf{\Lambda}(t) + \boldsymbol{\delta}) \geq q(\mathbf{\Lambda}(t)) + \nabla q(\mathbf{\Lambda}(t))^\top \delta - \frac{N}{2}\boldsymbol{\delta}^\top \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_N \end{bmatrix} \boldsymbol{\delta},$$

for every perturbation $\boldsymbol{\delta}$. Thus, using the same line of proof of the gradient algorithm [9, Chapter 2], we can conclude that the sequence $\{\Lambda(t)\}$ generated by iteration (3.13) converges in objective value to the optimal cost $q^\star$ of (3.7). Since $\mathbf{W}$ is diagonal, then (3.13) splits in a component-wise fashion giving

$$\mathbf{\Lambda}_i(t+1) = \mathbf{\Lambda}_i(t) + \mathbf{W}_{ii}\nabla_{\Lambda_i} q(\Lambda(t)),\ i \in \{1, \dots, N\}, \tag{3.14}$$

where $\mathbf{W}_{ii}$ denotes the (i,i)-th entry of $\mathbf{W}$.

By using the following property of conjugate functions

$$\nabla\varphi^*(\mathbf{z}) = \operatorname*{argmin}_{\mathbf{x}} \big(\varphi(\mathbf{x}) - \mathbf{z}^\top \mathbf{x}\big),$$

we have that the primal minimization (3.8) computes ∇f_i^* evaluated at the point $\big(\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_i^{(i,j)}(t) - \boldsymbol{\lambda}_i^{(j,i)}(t)), \{(\boldsymbol{\lambda}_j^{(i,j)}(t) - \boldsymbol{\lambda}_j^{(j,i)}(t))\}_{j\in\mathcal{N}_i}\big)$. Then

$$\begin{aligned} \frac{\partial q(\mathbf{\Lambda}(t))}{\partial \boldsymbol{\lambda}_i^{(i,j)}} &= \mathbf{x}_i^{(i)}(t+1) - \mathbf{x}_i^{(j)}(t+1),\ j \in \mathcal{N}_i \\ \frac{\partial q(\mathbf{\Lambda}(t))}{\partial \boldsymbol{\lambda}_j^{(i,j)}} &= \mathbf{x}_j^{(i)}(t+1) - \mathbf{x}_j^{(j)}(t+1),\ j \in \mathcal{N}_i, \end{aligned} \tag{3.15}$$

so that update (3.9) is the scaled gradient ascent (3.14).

Third and final, by Assumption 3.2.3 (Slater's condition), strong duality between problems (3.2) and (3.7) holds. Moreover, since problems (3.2) and (3.1) are equivalent, then they both have optimal cost $q^\star = f^\star$. Thus, the sequence $\{\mathbf{\Lambda}(t)\}_{t\geq 0}$ generated by PDD converges in objective value to the optimal cost $f^\star$ of (3.1).

For the second part of the statement, we first notice that in light of Assumptions 3.2.1 and 3.2.2, problem (3.1) has a unique optimal solution $\mathbf{x}^\star = (\mathbf{x}_1^{\star\top}, \dots, \mathbf{x}_N^{\star\top})^\top$. Further, since problem (3.2) is equivalent to problem (3.1), then $\mathbf{x}^\star$ is the unique optimal solution also for problem (3.2). Finally, the first order optimality condition for the (unconstrained) dual problem (3.7) is $\nabla q(\mathbf{\Lambda}^\star) = 0$, where $\mathbf{\Lambda}^\star$ is a limit point of the sequence $\{\mathbf{\Lambda}(t)\}_{t\geq 0}$ (which exists by the Lipschitz continuity of $\nabla q(\mathbf{\Lambda})$). This allows us to conclude, by equation (3.15), that the limit point of the primal sequences $\{\mathbf{x}_i^{(i)}(t), \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}(t)\}$ satisfy the primal coherence constraints. Thus, in the limit the copies $\mathbf{x}_i^{(i)}, \{\mathbf{x}_i^{(j)}\}_{j\in\mathcal{N}_i}$ of the variable $\mathbf{x}_i$ are equal to the (unique) optimal $\mathbf{x}_i^\star$. Iterating on $i \in \{1, \dots, N\}$ the proof follows. □

Remark 3.2.9. *Alternative expressions for L_i in* (3.10) *can be used. Larger upper bounds on the step-sizes α_i can be established by exploiting tailored descent conditions. See, e.g., works* [58, 66, 86]. □

3.2.3 Asynchronous Partitioned Dual Decomposition (Asyn-PDD)

In this subsection we present an asynchronous partitioned distributed algorithm, and prove its convergence with high probability. This algorithm can be interpreted as an extension of the PDD distributed algorithm.

As described in Section 1.1, we consider an asynchronous protocol where each node has its own concept of time defined by a local timer, which randomly and independently of the other nodes triggers when to awake itself. Each node is in an *idle* mode, wherein it continuously receives messages from neighboring nodes, until it is triggered either by the local timer or by a message from neighboring nodes. When a trigger occurs, it switches into an *awake* mode in which it updates its local variables and possibly transmits the updated information to its neighbors. The timer is modeled by means of a local clock $\tau_i \in \mathbb{R}_{\geq 0}$ and a randomly generated waiting time T_i. The timer triggers the node when $\tau_i = T_i$, so that the node switches to the awake mode and, after running the local computation, resets $\tau_i = 0$ and extracts a new realization of T_i. We make the following assumption on the local waiting times T_i.

Assumption 3.2.10 (Exponential i.i.d. local timers)**.** *The waiting times between consecutive triggering events are i.i.d. random variables with same exponential distribution.* □

Notice that here we are tacitly assuming that when an agent performs its local computations it has already received the most updated information from its neighbors, and this can be easily verified locally by each node.

Informally, the asynchronous distributed optimization algorithm is as follows. When a node i is in idle, it continuously receives messages from

awake neighbors. If the local timer τ_i triggers or new dual variables $\boldsymbol{\lambda}_i^{(j,i)}$, $\boldsymbol{\lambda}_j^{(j,i)}$ are received, it wakes up. When node i wakes up, it updates and broadcasts its primal variable $\mathbf{y}^{(i)} = \left(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\right)$, computed through a local constrained minimization. Moreover, if the transition was due to the local timer triggering, then it also updates and broadcasts its local dual variables $\boldsymbol{\lambda}_i^{(i,j)}$ and $\boldsymbol{\lambda}_j^{(i,j)}$, $j \in \mathcal{N}_i$. Since there is no global iteration counter, we highlight the difference between updated and not updated values during the "awake" phase, by means of a "+" superscript symbol, e.g., we denote the updated primal variable as $\mathbf{x}_i^{(i)+}$.

We want to stress some important aspects of the idle/awake cycle. First, these two phases are regulated by local timers and local information exchange, without the need of any central clock. Second, we assume that the computation in idle takes a negligible time compared to the one performed in the awake phase. Moreover, a constant, local step-size α_i is used in the ascent step, which can be initialized by means of local exchange of information between neighboring nodes. Finally, we point out that each agent uses the most updated values that are locally available to perform every computation.

The AsynPDD distributed algorithm is formally described in the table Distributed Algorithm 6.

As already described in Section 1.1 and Section 2.4.3, we point out that being the algorithm asynchronous, for the analysis we need to carefully formalize the concept of algorithm iterations. We will use a nonnegative integer variable t indexing a change in the whole state $\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_1 \dots \boldsymbol{\Lambda}_N]$ of the distributed algorithm. Each triggering will induce an iteration t of the distributed optimization algorithm and the value of t does not need to be known by the agents. That is, t is not a common clock and is only introduced for the sake of analysis.

Theorem 3.2.11. *Let Assumptions 3.2.1, 3.2.2 and 3.2.3 hold true. Let the timers τ_i satisfy Assumption 3.2.10 and step-sizes α_i be constant and such that $0 < \alpha_i \leq 1/L_i$, with*

$$L_i = \sqrt{2 \sum_{j\in\mathcal{N}_i} \left(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\right)^2}, \quad \forall\, i \in \{1, \dots, N\}. \tag{3.17}$$

Then, the random sequence $\{\boldsymbol{\Lambda}_1(t), \dots, \boldsymbol{\Lambda}_N(t)\}_{t\geq 0}$ generated by the AsynPDD (Distributed Algorithm 6), converges with high probability in objective value to the optimal cost $f^\star$ of problem (3.1), i.e., for any $\varepsilon \in (0, q_0)$, with $q_0 := q(\boldsymbol{\Lambda}(0))$, and target confidence $0 < \rho < 1$, there exists $\bar{t}(\varepsilon, \rho)$ such that for all $t \geq \bar{t}(\varepsilon, \rho)$ it holds

$$\Pr\left(\left|q(\boldsymbol{\Lambda}(t)) - f^\star\right| \leq \varepsilon\right) \geq 1 - \rho.$$

Distributed Algorithm 6 AsynPDD

Processor states: $(\mathbf{x}_i^{(i)}, \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i})$ and $\{\boldsymbol{\lambda}_i^{(i,j)}, \boldsymbol{\lambda}_j^{(i,j)}\}_{j\in\mathcal{N}_i}$
Set $\tau_i = 0$ and get a realization T_i

Evolution:
IDLE:
WHILE: $\tau_i < T_i$ DO:
Receive $\boldsymbol{\lambda}_i^{(j,i)}$, $\boldsymbol{\lambda}_j^{(j,i)}$ and/or $\mathbf{x}_i^{(j)}$, $\mathbf{x}_j^{(j)}$ from $j \in \mathcal{N}_i$.
IF: dual variables are received go to *AWAKE*.
go to *AWAKE*.

AWAKE:
Compute and broadcast

$$\begin{aligned}\big(\mathbf{x}_i^{(i)+}, \{\mathbf{x}_j^{(i)+}\}_{j\in\mathcal{N}_i}\big) = &\operatorname*{argmin}_{(\mathbf{x}_i,\{\mathbf{x}_j\}_{j\in\mathcal{N}_i})\in X_i} f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \\ &+ \mathbf{x}_i^\top \sum_{j\in\mathcal{N}_i} \big(\boldsymbol{\lambda}_i^{(i,j)} - \boldsymbol{\lambda}_i^{(j,i)}\big) + \sum_{j\in\mathcal{N}_i} \mathbf{x}_j^\top \big(\boldsymbol{\lambda}_j^{(i,j)} - \boldsymbol{\lambda}_j^{(j,i)}\big)\end{aligned}$$

IF: $\tau_i = T_i$ THEN: update and broadcast

$$\begin{aligned}\boldsymbol{\lambda}_i^{(i,j)+} &= \boldsymbol{\lambda}_i^{(i,j)} + \alpha_i\big(\mathbf{x}_i^{(i)+} - \mathbf{x}_i^{(j)}\big), \ \forall\, j \in \mathcal{N}_i, \\ \boldsymbol{\lambda}_j^{(i,j)+} &= \boldsymbol{\lambda}_j^{(i,j)} + \alpha_i\big(\mathbf{x}_j^{(i)+} - \mathbf{x}_j^{(j)}\big), \ \forall\, j \in \mathcal{N}_i,\end{aligned} \tag{3.16}$$

set $\tau_i = 0$ and get a new realization T_i.

Go to *IDLE*.

Proof. Our proof strategy is based on showing that the iterations of the asynchronous distributed algorithm can be written as the iterations of an ad-hoc version of the coordinate method [85], applied to the dual problem (3.7).

Let the optimization variable $\boldsymbol{\Lambda}$ be partitioned in N blocks $[\boldsymbol{\Lambda}_1, \dots, \boldsymbol{\Lambda}_N]$ as in (3.11), then a coordinate approach consists in an iterative scheme in which only a block-per-iteration, say Λ_{i_t} at time t, of the entire optimization variable $\boldsymbol{\Lambda}$ is updated at time t, while all the other components $\boldsymbol{\Lambda}_j$ with $j \in \{1, \dots, N\} \setminus \{i_t\}$ stay unchanged. Formally, a coordinate iteration can be summarized as

$$\begin{aligned}\boldsymbol{\Lambda}_{i_t}(t+1) &= \boldsymbol{\Lambda}_{i_t}(t) + \nabla_{\boldsymbol{\Lambda}_{i_t}} q(\boldsymbol{\Lambda}(t)) \\ \boldsymbol{\Lambda}_j(t+1) &= \boldsymbol{\Lambda}_j(t), \qquad\qquad j \neq i_t.\end{aligned} \tag{3.18}$$

In the following, we show that the AsynPDD distributed algorithm implements (3.18) with a uniform random selection of the blocks. Since the timers τ_i trigger independently according to the same exponential distribution, then from an external perspective the induced awakening process of

the nodes corresponds to the following: only one node per iteration, say i_t, wakes up randomly, uniformly and independently from previous iterations. (See Section 1.1 and Section 2.4.3 for details). Thus, each triggering, which induces an iteration of the distributed optimization algorithm and is indexed with t, corresponds to the (uniform) selection of a node in $\{1, \dots, N\}$ that becomes awake.

Next we show by induction that if each node i has an updated version of the neighboring variables before it gets awake, then the same holds after the update. When node i wakes up, it uses for its update its own primal variables $\mathbf{x}_i^{(i)}$ and $\mathbf{x}_j^{(i)}$, $j \in \mathcal{N}_i$, which are clearly updated since i is the one modifying them. Moreover, node i uses also $\mathbf{x}_i^{(j)}$ and $\mathbf{x}_j^{(j)}$, $j \in \mathcal{N}_i$, which are received by neighboring nodes $j \in \mathcal{N}_i$. These variables are updated by j if itself or one of its neighbors becomes awake. In both cases node j sends the updated variable to its neighbors (which include node i). An analogous argument holds for the dual variables.

Thanks to the argument just shown and by noting that $\boldsymbol{\lambda}_{i_t}^{(i_t,j)}$ and $\boldsymbol{\lambda}_j^{(i_t,j)}$, $j \in \mathcal{N}_{i_t}$ are the components of $\boldsymbol{\Lambda}_{i_t}$, we have that step (3.16) corresponds to step (3.18) with i_t randomly uniformly distributed over $\{1, \dots, N\}$. Finally, recalling that (i) the cost function $q(\boldsymbol{\Lambda})$ of problem (3.7) has block-coordinate Lipschitz continuous gradient with respect to the blocks $\boldsymbol{\Lambda}_i$ (see proof of Theorem 3.2.8) and (ii) the step-sizes α_i are constant and such that $0 < \alpha_i \leq 1/L_i$ with L_i in (3.17), we can invoke [85, Theorem 5] to conclude that the coordinate method (3.18) (and equivalently the AsynPDD distributed algorithm) converges with high probability to the optimal cost $q^\star$ of problem (3.7). Recalling that strong duality between problems (3.1) and (3.7) holds (see proof of Theorem 3.2.8), then $q^\star = f^\star$, and the proof follows. □

Remark 3.2.12. *As highlighted in Theorem 3.2.11 in order to set the local step-sizes α_i, each node i should know the convexity parameter σ_j of its neighbors but, differently from the synchronous case does not need to know the total number of agents N in the network. (Cf. condition* (3.10)*).* □

3.3 Randomized Primal Approach

In this section we propose an alternative approach to solve a partitioned big-data optimization problem. We consider a distributed optimization scenario in which the aggregate objective function to minimize is partitioned, big-data and (possibly) nonconvex. The proposed method will work on the primal formulation of the problem, thus allowing us to handle nonconvexity.

We consider partitioned optimization problems described in Section 1.2.2 having a more specific structure. Each local objective f_i has a sparsity consistent with the interaction graph, namely, for $i \in \{1, \dots, N\}$, the function

(possibly nonconvex) f_i depends only on the component of node i and of its neighbors. Also, each function g_i depends only on the component $\mathbf{x}_i$.

In light of the described structure, the problem we aim at solving in a distributed way can be written as

$$\min_{\mathbf{x}\in\mathbb{R}^d} \sum_{i=1}^{N} f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) + g_i(\mathbf{x}_i), \tag{3.19}$$

where each node i knows only the functions $f_i : \mathbb{R}^{\sum_{j\in\mathcal{N}_i\cup\{i\}} n_j} \to \mathbb{R}$ and $g_i : \mathbb{R}^{n_i} \to \mathbb{R}\cup\{+\infty\}$.

Note that, in this partitioned scenario, network structure and objective function are inherently related. That is, nodes that share a variable are neighbors in the communication graph. In the following assumptions we state the main properties of problem (3.19).

Assumption 3.3.1. *For all $i \in \{1,\dots,N\}$, f_i is a smooth function of $(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})$. In particular, f_i has block-coordinate Lipschitz continuous gradient, i.e., for all $j \in \mathcal{N}_i$ there exists constants $L_{ij} > 0$ such that for all $(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) \in \mathbb{R}^{\sum_{\ell\in\mathcal{N}_i} n_\ell}$ and $\mathbf{s}_j \in \mathbb{R}^{n_j}$ it holds*

$$\Big\|\nabla_{\mathbf{x}_j} f_i\Big(\big(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}\big) + \mathbf{U}_{ij}\mathbf{s}_j\Big) - \nabla_{\mathbf{x}_j} f_i\big(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}\big)\Big\| \le L_{ij}\|\mathbf{s}_j\|.$$

where $\mathbf{U}_{ij}$ is a suitable matrix such that $\mathbf{U}_{ij}\mathbf{s}_j$ is a vector in $\mathbb{R}^{\sum_{\ell\in\mathcal{N}_i} n_\ell}$ with j-th block-component equal to $\mathbf{s}_j$ and all the other ones equal to zero. □

In light of Assumption 3.3.1, it is easy to show that the following lemma holds.

Lemma 3.3.2. *Let Assumption 3.3.1 hold, then the aggregate function $f(\mathbf{x}) := \sum_{i=1}^{N} f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})$ has block-coordinate Lipschitz continuous gradient. In particular, for all $i \in \{1,\dots,N\}$, the partial gradient $\nabla_{\mathbf{x}_i} f$ has Lipschitz constant given by $L_i := \sum_{j\in\mathcal{N}_i} L_{ij}$.*

Proof. The proof follows straight by simply writing the norm of the aggregate cost f and then bounding each term of its gradient by using its block Lipschitz constant. □

Remark 3.3.3. *Note that one can assume directly that $\nabla_{\mathbf{x}_i} f$ is Lipschitz continuous, but while the condition we impose can be checked in a distributed way, the weaker one needs a global knowledge of the cost f.* □

Assumption 3.3.4. *For all $i \in \{1,\dots,N\}$, the function g_i is a proper, closed, proper, convex function.* □

We stress that we have not assumed any convexity condition on f_i, thus optimization problem (3.19) is nonconvex in general. Finally, we state the following assumption which is quite standard for nonconvex scenarios.

Assumption 3.3.5. *The cost* $V(\mathbf{x}) := \sum_{i=1}^{N} f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) + g_i(\mathbf{x}_i)$ *of problem* (3.19) *is a coercive function.* □

Assumption 3.3.5 guarantees that at least a local minimum for problem (3.19) exists.

3.3.1 Partitioned Coordinate Descent (PCD)

In this subsection we present our asynchronous distributed algorithm for solving problem (3.19). In order to develop the algorithm,we will adopt the asynchronous communication protocol that has been introduced in Section 1.1. Moreover, each node build a local model of the cost to perform its local (descent) update. It is based on a *local* quadratic, strongly-convex approximation of the cost function that each node computes.

Formally, each node $i \in \{1, \dots, N\}$ constructs the following local approximation of the entire cost function at a fixed $\bar{\mathbf{x}} \in \mathbb{R}^d$ (neglecting the constant term $f(\bar{\mathbf{x}})$ which does not affect the optimization),

$$\begin{aligned} q_i(\mathbf{s}_i; \bar{\mathbf{x}}) &:= \nabla_{\mathbf{x}_i} f(\bar{\mathbf{x}})^\top \mathbf{s}_i + \frac{1}{2}\|\mathbf{s}_i\|^2_{\mathbf{Q}_i(\bar{\mathbf{x}})} + g_i(\bar{\mathbf{x}}_i + \mathbf{s}_i) \\ &= \sum_{j\in\mathcal{N}_i} \nabla_{\mathbf{x}_i} f_j(\bar{\mathbf{x}}_j, \{\bar{\mathbf{x}}_h\}_{h\in\mathcal{N}_j})^\top \mathbf{s}_i + \frac{1}{2}\|\mathbf{s}_i\|^2_{\mathbf{Q}_i(\bar{\mathbf{x}})} + g_i(\bar{\mathbf{x}}_i + \mathbf{s}_i) \end{aligned} \tag{3.20}$$

with $\mathbf{Q}_i(\bar{\mathbf{x}}) \in \mathbb{R}^{n_i \times n_i}$ a symmetric, positive definite matrix satisfying the following assumption.

Assumption 3.3.6. *For any* $\mathbf{x} \in \mathbb{R}^d$ *and* $i \in \{1, \dots, N\}$ *it holds that* $\mathbf{Q}_i(\mathbf{x}) \succeq L_i \mathbf{I}$. □

Intuitively Assumption 3.3.6 guarantees the strong convexity of q_i. The role of the Lipschitz constant L_i in the bound will be clear in the analysis of the algorithm given in Section 3.3.2.

Informally, the asynchronous distributed optimization algorithms is as follows. Each node i takes care of modifying the variable $\mathbf{x}_i$. We denote $\bar{\mathbf{x}}_i$ the current state of node i, which is the estimated optimal value of $\mathbf{x}_i$.

When a node i wakes up, it updates its state $\bar{\mathbf{x}}_i$ by moving in the direction obtained from the minimization of its local approximation $q_i(\mathbf{s}_i; \bar{\mathbf{x}})$, being $\bar{\mathbf{x}}$ the current value of the decision variable. Then, it sends to each neighbor $j \in \mathcal{N}_i$ the updated $\bar{\mathbf{x}}_i$ and $\nabla_{\mathbf{x}_j} f_i(\bar{\mathbf{x}}_i, \{\bar{\mathbf{x}}_j\}_{j\in\mathcal{N}_i})$. When in idle, node i is in a listening mode. If an updated $\nabla_{\mathbf{x}_i} f_j(\bar{\mathbf{x}}_j, \{\bar{\mathbf{x}}_h\}_{h\in\mathcal{N}_j})$ is received from a neighbor j no computation is needed. If $\bar{\mathbf{x}}_j$ is also received (j was an awake node) the following happens. Node i updates the partial gradients of its local function f_i according to the new $\bar{\mathbf{x}}_j$, and sends back the updated partial gradients to its neighbors. In order to highlight the difference between updated and old variables at node i during the awake phase, we denote the updated ones with a "+" symbol, e.g., as $\bar{\mathbf{x}}_i^+$.

We want to stress two important aspects of the idle/awake cycle. First, these two phases are regulated by local timers without the need of any central clock. Second, when in idle a node only receives messages and from time to time evaluates a partial gradient, which takes a negligible time compared to the computation performed in the awake phase.

The distributed algorithm is formally reported in Distributed Algorithm 7 from the perspective of node i.

Distributed Algorithm 7 PCD

Processor state: $\bar{\mathbf{x}}_i$

Initialization: set $\tau_i = 0$ and get a realization T_i

Evolution:

IDLE:

WHILE: $\tau_i \leq T_i$ DO:

receive $\bar{\mathbf{x}}_j$ and/or $\nabla_{\mathbf{x}_i} f_j(\bar{\mathbf{x}}_j, \{\bar{\mathbf{x}}_h\}_{h\in\mathcal{N}_j})$ from $j \in \mathcal{N}_i$

evaluate $\nabla_{\mathbf{x}_j} f_i(\bar{\mathbf{x}}_i, \{\bar{\mathbf{x}}_j\}_{j\in\mathcal{N}_i})$ and send it to $j \in \mathcal{N}_i$

go to *AWAKE*.

AWAKE:

compute $$\mathbf{d}_i = \operatorname*{argmin}_{\mathbf{s}_i} q_i(\mathbf{s}_i; \bar{\mathbf{x}}) \tag{3.21}$$

update $$\bar{\mathbf{x}}_i^+ = \bar{\mathbf{x}}_i + \mathbf{d}_i \tag{3.22}$$

broadcast $\bar{\mathbf{x}}_i^+$, $\nabla_{\mathbf{x}_j} f_i(\bar{\mathbf{x}}_i^+, \{\bar{\mathbf{x}}_j\}_{j\in\mathcal{N}_i})$ to $j \in \mathcal{N}_i$

set $\tau_i = 0$, get a new realization T_i and go to *IDLE*.

We point out some aspects involving the local approximation (3.20) that each node uses in its local computations.

First, it is worth noting $q_i(\mathbf{s}_i; \bar{\mathbf{x}})$ does not depend on the entire state $\bar{\mathbf{x}}$, but only on $\bar{\mathbf{x}}_j, \{\bar{\mathbf{x}}_h\}_{h\in\mathcal{N}_j}$, $j \in \mathcal{N}_i$ and therefore is constructed by node i by using only information from its neighbors. Moreover, node i does not needed the expression of neighboring cost functions f_j to build $q_i(\mathbf{s}_i; \bar{\mathbf{x}})$, but only the gradients $\nabla_{\mathbf{x}_i} f_j$. In some special cases (discussed in the following paragraph), $\mathbf{Q}_i(\mathbf{x})$ could include second order information of f_j, $j \in \mathcal{N}_i$, i.e., $\nabla^2_{\mathbf{x}_i,\mathbf{x}_i} f_j$, that should be sent together with the gradients.

Second, different choices for the weight matrix $\mathbf{Q}_i(\mathbf{x})$ are allowed. By exploiting the block Lipschitz continuity of the gradient of f, a first simple choice is to set $\mathbf{Q}_i(\mathbf{x}) := L_i\mathbf{I}$ for all $i \in \{1, \ldots, N\}$ and $\mathbf{x} \in \mathbb{R}^d$. Motivated by existing works in the literature, e.g., [33], non diagonal choices for $\mathbf{Q}_i(\mathbf{x})$ are reasonable: for instance, assuming $f \in \mathcal{C}^2$, one can select a second order approximation, i.e., set $\mathbf{Q}_i(\mathbf{x}) := \nabla^2_{\mathbf{x}_i,\mathbf{x}_i} f(\mathbf{x}) + \epsilon_i\mathbf{I}$ for a sufficiently large

$\epsilon_i > 0$ for all $i \in \{1, \dots, N\}$. As mentioned above this information can be constructed in a distributed manner.

Third and final, recalling the definition of the proximal operator $\mathbf{prox}_{\alpha,\varphi} : \mathbb{R}^d \to \mathbb{R}^d$ of a closed, proper, convex function $\varphi : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ given by $\mathbf{prox}_{\alpha,\varphi}(\mathbf{v}) := \operatorname{argmin}_{\mathbf{x}} \left(\varphi(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{v}\|^2\right)$ with $\alpha > 0$, we have that for $\mathbf{Q}_i(x) = L_i\mathbf{I}$ the update law described in (3.21)-(3.22), can be rephrased in term of proximal operators and, thus, leading to a distributed coordinate proximal gradient method. On this regard it is worth noting that our algorithm, with a general expression for $\mathbf{Q}_i$, can be written in terms of a generalized, weighted version of the proximal operator as follows. Given a positive definite matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$, we define

$$\mathbf{prox}_{\mathbf{W},\varphi}(\mathbf{v}) := \operatorname*{argmin}_{\mathbf{x}} \left\{\varphi(\mathbf{x}) + \frac{1}{2}\big\|\mathbf{x} - \mathbf{v}\big\|^2_{\mathbf{W}^{-1}}\right\}, \tag{3.23}$$

thus, the iteration (3.21)-(3.22) can be recast as

$$\bar{\mathbf{x}}_i^+ = \mathbf{prox}_{\mathbf{Q}_i(\bar{\mathbf{x}})^{-1},g_i}\Big(\bar{\mathbf{x}}_i - \mathbf{Q}_i(\bar{\mathbf{x}})^{-1} \sum_{j\in\mathcal{N}_i} \nabla_{\mathbf{x}_i} f_j(\bar{\mathbf{x}}_j, \{\bar{\mathbf{x}}_h\}_{h\in\mathcal{N}_j})\Big).$$

3.3.2 Convergence Analysis of PCD

In this subsection we prove the convergence in probability of PCD distributed optimization algorithm.

As for the algorithms in Section 2.3 and Section 3.2, being the algorithm asynchronous, it is worth pointing out that, for the analysis we need to carefully formalize the concept of algorithm iterations. We will use a nonnegative integer variable t indexing a change in the whole state $\bar{\mathbf{x}} = [\bar{\mathbf{x}}_1^\top \dots \bar{\mathbf{x}}_N^\top]^\top$ of the distributed algorithm. In particular, each triggering will induce an *iteration* of the distributed optimization algorithm and will be indexed with t. We want to stress that this (integer) variable t does not need to be known by the agents. That is, this timer is not a common clock and is only introduced for the sake of analysis.

Theorem 3.3.7. *Let Assumptions 3.3.1, 3.3.4, 3.3.5 and 3.3.6 hold true. Then, the Partitioned Coordinate Descent distributed algorithm generates a sequence* $\mathbf{x}(t) := [\bar{\mathbf{x}}_1(t)^\top, \dots, \bar{\mathbf{x}}_N(t)^\top]^\top$ *(obtained stacking the nodes' states) such that the random variable* $V(\mathbf{x}(t))$ *converges almost surely, i.e., there exists a random variable* $V^\star$ *such that*

$$\Pr\Big(V(\mathbf{x}(t)) = V^\star\Big) = 1.$$

Moreover, any limit point $\mathbf{x}^\star$ *of* $[\bar{\mathbf{x}}_1(t)^\top, \dots, \bar{\mathbf{x}}_N(t)^\top]^\top$ *is a* stationary *point of problem* (3.19) *and, thus, satisfies its first order optimality condition, i.e., there exists a subgradient* $\widetilde{\nabla} g(\mathbf{x}^\star)$ *of* g *at* $\mathbf{x}^\star$ *such that* $\nabla f(\mathbf{x}^\star) + \widetilde{\nabla} g(\mathbf{x}^\star) = 0$. □

Coordinate descent method for composite nonconvex minimization

We consider a more general composite optimization problem and prove a result that is instrumental to the convergence proof of our distributed algorithm. We introduce a generalization of the algorithm proposed in [67,83,85] based on the quadratic approximation introduced in (3.20). We present the algorithm for problem (3.19), but we want to stress that the algorithm can be applied to a general function $f : \mathbb{R}^d \to \mathbb{R}$ with block-Lipschitz continuous gradient. This will be clear from the analysis.

We consider a coordinate descent method based on selecting a random block-component, say $\mathbf{x}_i$, of $\mathbf{x}$ at each iteration and updating only $\mathbf{x}_i$ through a suitable descent rule. The descent step is based on the quadratic approximation of the cost function given in (3.20). The coordinate descent method is formally summarized in the table Algorithm 8.

Algorithm 8 Generalized Coordinate Descent Algorithm

Choose a random block $i_t \in \{1, \dots, N\}$ with probability p_{i_t}

Compute a descent direction d_{i_t} solving

$$\mathbf{d}_{i_t} = \underset{\mathbf{s}_i}{\operatorname{argmin}}\ q_{i_t}(\mathbf{s}_i; \mathbf{x}(t)) \tag{3.24}$$

Update the decision variable according to

$$\begin{aligned} \mathbf{x}_{i_t}(t+1) &= \mathbf{x}_{i_t}(t) + \mathbf{d}_{i_t} \\ \mathbf{x}_j(t+1) &= \mathbf{x}_j(t), \qquad \text{for all } j \neq i_t \end{aligned} \tag{3.25}$$

In the following we present results for the theoretical convergence of the generalized coordinate descent algorithm.

Lemma 3.3.8. *Let Assumption 3.3.1, 3.3.4, 3.3.6 hold. Let $\mathbf{x}(t)$ be the random sequence generated by Generalized Coordinate Descent Algorithm, then for all $t \geq 0$ it holds*

$$V(\mathbf{x}(t+1)) \leq V(\mathbf{x}(t)) - \frac{L_{i_t}}{2}\|\mathbf{d}_{i_t}\|^2.$$

Proof. From Assumption 3.3.1 (Lipschitz continuity of ∇f), we can write the well-known descent lemma (see [9, Proposition A.24]), for all $i \in \{1, \dots, N\}$ and for all $\bar{\mathbf{x}} \in \mathbb{R}^d$

$$V(\bar{\mathbf{x}} + \mathbf{U}_i\mathbf{s}_i) \leq f(\bar{\mathbf{x}}) + \nabla_{\mathbf{x}_i} f(\bar{\mathbf{x}})^\top \mathbf{s}_i + \frac{L_i}{2}\|\mathbf{s}_i\|^2 + g_i(\bar{\mathbf{x}}_i + \mathbf{s}_i) + \sum_{j \neq i} g_j(\bar{\mathbf{x}}_j),$$

where $\mathbf{U}_i$ is obtained by a block decomposition of the $d \times d$ identity matrix, i.e., $\mathbf{I} = [\mathbf{U}_1 \dots \mathbf{U}_N]$, where for all $i \in \{1, \dots, N\}$ each $U_i \in \mathbb{R}^{d \times n_i}$

Since Q_i satisfies Assumption 3.3.6, then we can generalize the above descent condition by introducing a uniform bound depending on the Lipschitz constant of block i, i.e.,

$$V(\bar{\mathbf{x}} + \mathbf{U}_i \mathbf{s}_i) \leq f(\bar{\mathbf{x}}) + \nabla_{\mathbf{x}_i} f(\bar{\mathbf{x}})^\top \mathbf{s}_i + \frac{1}{2} \|\mathbf{s}_i\|^2_{\mathbf{Q}_i(\bar{\mathbf{x}})} + g_i(\bar{\mathbf{x}}_i + \mathbf{s}_i) + \sum_{j \neq i} g_j(\bar{\mathbf{x}}_j)$$

Due the partitioned structure of f, the explicit expression of $\nabla_{\mathbf{x}_i} f(\mathbf{x})$ actually depends only on f_j, $j \in \mathcal{N}_i$, thus the latter condition can be further rephrased as

$$V(\bar{\mathbf{x}} + \mathbf{U}_i \mathbf{s}_i) \leq q_i(\mathbf{s}_i; \bar{\mathbf{x}}) + f(\bar{\mathbf{x}}) + \sum_{j \neq i} g_j(\bar{\mathbf{x}}_j). \tag{3.26}$$

with $q_i(\mathbf{s}_i; \bar{\mathbf{x}})$ defined as in (3.20).

Consider a descent direction $\mathbf{d}_{i_t}$ computed as in (3.24), then $\mathbf{d}_{i_t}$ satisfies the first order necessary condition of optimality for problem (3.24)

$$\nabla_{\mathbf{x}_{i_t}} f(\mathbf{x}(t)) + \mathbf{Q}_{i_t}(\mathbf{x}(t))\, \mathbf{d}_{i_t} + \widetilde{\nabla} g_{i_t}(\mathbf{x}_{i_t}(t) + \mathbf{d}_{i_t}) = 0, \tag{3.27}$$

where $\widetilde{\nabla} g_{i_t} \in \mathbb{R}^{n_{i_t}}$ denotes a subgradient of g_{i_t}.

Starting form equation (3.26) with the following identification $\bar{\mathbf{x}} = \mathbf{x}(t)$ and $\bar{\mathbf{x}} + \mathbf{U}_{i_t} \mathbf{d}_{i_t} = \mathbf{x}(t+1)$, and adding and subtracting the term $g_{i_t}(\mathbf{x}_{i_t}(t))$ we obtain

$$\begin{aligned}
V(\mathbf{x}(t+1)) &\leq V(\mathbf{x}(t)) + \nabla_{\mathbf{x}_{i_t}} f(\mathbf{x}(t))^\top \mathbf{d}_{i_t} + \frac{1}{2} \|\mathbf{d}_{i_t}\|^2_{\mathbf{Q}_{i_t}(\mathbf{x}(t))} \\
&\qquad + g_{i_t}(\mathbf{x}_{i_t}(t) + \mathbf{d}_{i_t}) - g_{i_t}(\mathbf{x}_{i_t}(t)) \\
&\leq V(\mathbf{x}(t)) + \nabla_{\mathbf{x}_{i_t}} f(\mathbf{x}(t))^\top \mathbf{d}_i \\
&\qquad + \frac{1}{2} \|\mathbf{d}_{i_t}\|^2_{\mathbf{Q}_{i_t}(\mathbf{x}(t))} + \widetilde{\nabla} g_{i_t}(\mathbf{x}_{i_t}(t) + \mathbf{d}_{i_t})^\top \mathbf{d}_{i_t} \\
&\leq V(\mathbf{x}(t)) - \frac{1}{2} \|\mathbf{d}_{i_t}\|^2_{\mathbf{Q}_{i_t}(\mathbf{x}(t))} \\
&\leq V(\mathbf{x}(t)) - \frac{L_i}{2} \|\mathbf{d}_{i_t}\|^2
\end{aligned}$$

where we used the convexity of g_{i_t}, the optimality condition (3.27) and the uniform bound in Assumption 3.3.6. □

Theorem 3.3.9. *Let Assumptions 3.3.1, 3.3.4, 3.3.5 and 3.3.6 hold true. Then, the Generalized Coordinate Descent Algorithm generates a sequence $\{\mathbf{x}(t)\}_{t \geq 0}$ such that the random variable $V(\mathbf{x}(t))$ converges almost surely. Moreover, any limit point $\mathbf{x}^\star$ of $\mathbf{x}(t)$ is a stationary point of V and, thus, satisfies the first order necessary condition for optimality for problem* (3.19)*, i.e., there exists a subgradient $\widetilde{\nabla} g(\mathbf{x}^\star)$ of g at $\mathbf{x}^\star$ such that*

$$\nabla f(\mathbf{x}^\star) + \widetilde{\nabla} g(\mathbf{x}^\star) = 0$$

Proof. The result is proven by following the same line as in [83, Theorem 1] where the generalized Lemma 3.3.8 is used in place of [83, Lemma 3]. □

Proof of Theorem 3.3.7

Our proof strategy is based on showing that the iterations of the asynchronous distributed algorithm can be written as the iterations of an ad-hoc version of the coordinate descent method for composite nonconvex functions given in the previous section and summarized in Algorithm 8.

Timer model and uniform node extraction. The timers trigger independently according to the same exponential distribution and induce an awakening process of the nodes corresponds in which only one agent per iteration wakes up randomly, uniformly and independently from previous iterations. (See Section 1.1 and Section 2.4.3 for details). Thus, each triggering event induces an iteration t of the distributed optimization algorithm and corresponds to the (uniform) selection of a node in $\{1, \dots, N\}$ that becomes awake. We denote i_t the extracted node. Notice that node i_t changes the value of its state $\bar{\mathbf{x}}_{i_t}$ while all the other states are not changed by the algorithm.

State consistency (inductive argument). Next we show by induction that if all the nodes have a consistent and updated information before a node i gets awake, then the same holds after the update. By consistent we mean that for a variable $\mathbf{x}_\ell$, all the nodes in $\mathcal{N}_\ell$ have the same state $\bar{\mathbf{x}}_\ell$. By updated we mean that each node ℓ has an updated value of the gradients $\nabla_{\mathbf{x}_\ell} f_j$, $j \in \mathcal{N}_\ell$. First, node i changes only its state $\bar{\mathbf{x}}_i$ relative to the variable $\mathbf{x}_i$. This variable is shared only with neighbors $j \in \mathcal{N}_i$, which receive the new state $\bar{\mathbf{x}}_i$ after the update. As regards the gradients, the ones affected by the change of the variable $\mathbf{x}_i$ are $\nabla_{\mathbf{x}_i} f_j$, with $j \in \mathcal{N}_i$. Notice that these gradients are only used by nodes $h \in \mathcal{N}_j$. But after the broadcast performed by i, each idle $j \in \mathcal{N}_i$ receives the updated $\bar{\mathbf{x}}_i$, updates the gradients, and sends them to its neighbors $h \in \mathcal{N}_j$. The variables and gradients for the rest of the nodes in the network are not changed by the update of node i.

Coordinate descent equivalence and convergence. Finally, we simply notice that, thanks to the consistency argument just shown, steps (3.21)-(3.22) correspond to steps (3.24)-(3.25). Thus, we have shown that the Distributed Algorithm 7 implements the centralized coordinate method Algorithm 8 and therefore inherits its convergence properties. By invoking Theorem 3.3.9, the proof follows.

3.4 Numerical Analysis

In this section we test the algorithms proposed in this section to numerical examples in order to corroborate the theoretical discussion.

3.4.1 Dual Method Numerical Analysis

We test the proposed distributed algorithms described in Section 3.2.1 on a quadratic program enjoying the partitioned structure described in the previous sections. Specifically, we consider a network of $N = 100$ agents communicating according to an undirected connected Erdős-Rényi random graph $\mathcal{G}$ with parameter $p = 0.2$. Thus, letting $\left(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}\right)$ denote a column vector, we consider the following partitioned optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_{i=1}^{N} \left(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}\right)^\top \mathbf{H}_i \left(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}\right) + \mathbf{r}_i^\top \left(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}\right) \\ \text{subj. to } & \mathbf{A}_i\left(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}\right) \preceq \mathbf{b}_i, \quad i \in \{1, \dots, N\}, \end{aligned} \tag{3.28}$$

where each $\mathbf{x}_i \in \mathbb{R}^{n_i}$ and m_i is uniformly drawn from $\{1, 2, 3, 4\}$. This optimization problem has the same partitioned structure discussed in Section 3.2.1. In particular, we have quadratic cost functions $f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i})$ and linear constraints $X_i = \{(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}) \mid \mathbf{A}_i(\mathbf{x}_i, \left\{\mathbf{x}_j\right\}_{j\in\mathcal{N}_i}) \preceq \mathbf{b}_i\}$. The matrices $\mathbf{H}_i$ are positive definite with eigenvalues uniformly generated in $[1, 20]$, while the vectors $\mathbf{r}_i$ have entries randomly generated in $[0, 100]$. Moreover, each pair $\mathbf{A}_i$, $\mathbf{b}_i$ describes a linear constraint having a number of rows uniformly drawn from $\{1, 2\}$. Each $\mathbf{A}_i$ has entries normally distributed with zero mean and unitary variance, while $\mathbf{b}_i$ are suitably generated to always obtain feasible linear constraints. For all $i \in \{1, \dots, N\}$, we use constant step-sizes $\alpha_i = 1/L_i$ with L_i computed as in (3.10) for the synchronous algorithm and as in (3.17) for the asynchronous case. All the dual variables are initialized to zero.

In Figure 3.2 we show the convergence rate of the synchronous distributed algorithm by plotting the difference between the dual cost $q(\mathbf{\Lambda}(t))$ at each iteration t and the optimal value $q^\star = f^\star$ of problem (3.28).

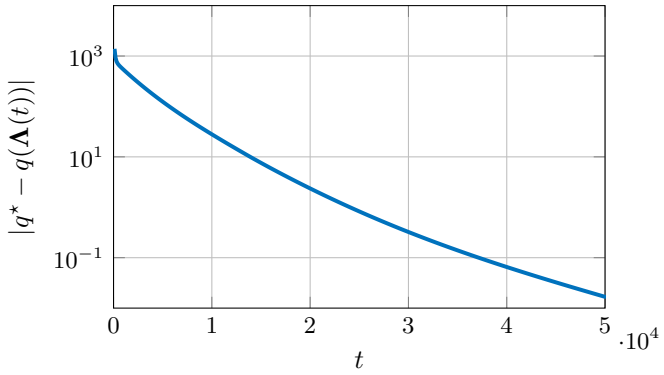


Figure 3.2: Evolution of the cost error for the synchronous distributed algorithm.

In Figure 3.3 we show the evolution of the difference between the generated primal sequence $\{\mathbf{x}_1^{(1)}(t), \ldots, \mathbf{x}_N^{(N)}(t)\}$ and the (unique) optimal primal solution $\mathbf{x}^\star$.

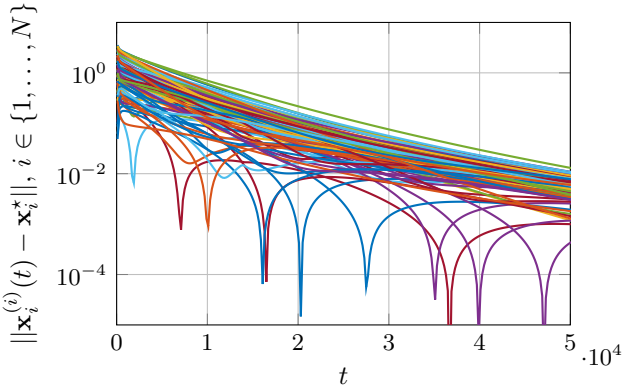


Figure 3.3: Evolution of the error on primal variables $\mathbf{x}_i^{(i)}$, $i \in \{1, \ldots, N\}$, for the synchronous distributed algorithm.

In Figure 3.4 we show the disagreement on the primal variable $\mathbf{x}_2$ between neighboring nodes $\mathcal{N}_2 \cup \{2\}$. In particular, we plot the norm of $\mathbf{x}_2^{(2)}(t) - \mathbf{x}_2^{(j)}(t)$, for all $j \in \mathcal{N}_2 \cup \{2\}$.

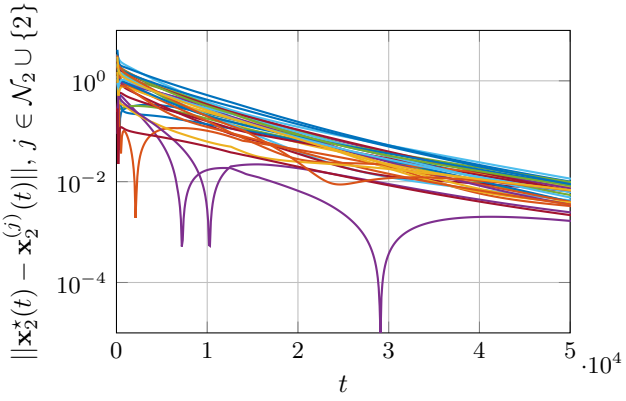


Figure 3.4: Evolution of the disagreement on $\mathbf{x}_2$ between agents 2 and its neighbors $j \in \mathcal{N}_2$ for the synchronous distributed algorithm.

Finally, in Figure 3.5 we show the convergence rate for the AsynPDD distributed algorithm. Since we are dealing with an asynchronous algorithm, we normalize the iteration counter t with respect to the total number of agents N. It is worth noting the cost evolution is not monotone as expected for the class of randomized algorithms.

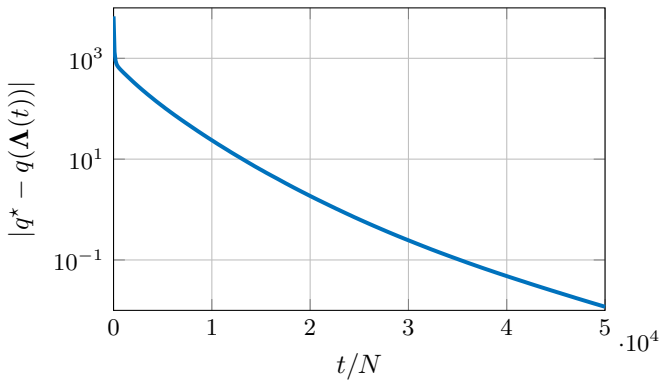


Figure 3.5: Evolution of the cost error for the asynchronous distributed algorithm.

3.4.2 Primal Method Numerical Analysis

We present a numerical example showing the effectiveness of the proposed primal algorithm presented in Section 3.3.

We consider an undirected connected Erdős-Rényi random graph $\mathcal{G}$, with parameter 0.2, connecting $N = 50$ nodes and we test the distributed algorithm on a partitioned nonconvex constrained quadratic program in the form

$$\min_{x_1,\dots,x_N} \sum_{i=1}^{N} \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right)^\top \mathbf{H}_i \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) + \mathbf{r}_i^\top \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) + I_{X_i}(x_i), \tag{3.29}$$

where each $x_i \in \mathbb{R}$ for all $i \in \{1,\dots,N\}$ and each cost matrix $\mathbf{H}_i \in \mathbb{R}^{(|\mathcal{N}_i|+1)\times(|\mathcal{N}_i|+1)}$ is only symmetric (*not* positive definite). We construct $\mathbf{H}_i$ as the difference between a positive definite matrix $\widetilde{\mathbf{H}}_i \in \mathbb{R}^{(|\mathcal{N}_i|+1)\times(|\mathcal{N}_i|+1)}$ and a suitable scaled version of the identity matrix. Finally, each function I_{X_i} denotes the indicator function of the segment $X_i = [-\ell_i, u_i]$, i.e., we constrain each x_i to lie into an interval. We set $\ell_i = -30$ and $u_i = 20$ for all $i \in \{1,\dots,N\}$.

Problem (3.29) fits our set-up described in Section 3.3 by defining

$$f_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) := \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right)^\top \mathbf{H}_i \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) + \mathbf{r}_i^\top \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right)$$

and

$$g_i(x_i) := I_{X_i}(x_i) = \begin{cases} x_i & \text{if } \ell_i \le x_i \le u_i \\ +\infty & \text{otherwise.} \end{cases}$$

Moreover, we use the local approximation $q_i(s_i, \bar{\mathbf{x}})$ as in (3.20) with $\mathbf{Q}_i = \frac{1}{\alpha_i}\mathbf{I}$ with $\alpha_i = 0.01$ for all $i \in \{1,\dots,N\}$.

In Figure 3.6 we plot the evolution of two selected components of the decision variable $\mathbf{x}$ at each iteration t (defined as discussed in Section 3.3.2), i.e., x_i^t, $i = 14, 48$. The horizontal dotted lines represent the centralized solution. Since the algorithm is asynchronous and based on a coordinate approach, we plot the rate of convergence with respect to the normalized iterations t/N in order to show the effective behavior with respect to the global time.

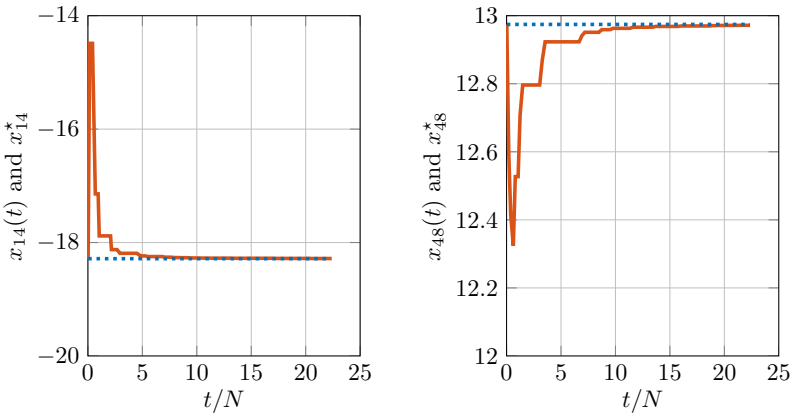


Figure 3.6: Evolution of two decision variables x_i, $i = 14, 48$, for the distributed algorithm.

In Figure 3.7 we show the difference, in logarithmic scale, between the cost $V(\mathbf{x}^t)$ at each iteration t and the value of V attained at the limit point $\mathbf{x}^\star$ of $\mathbf{x}^t$ (proven to be a stationary point).

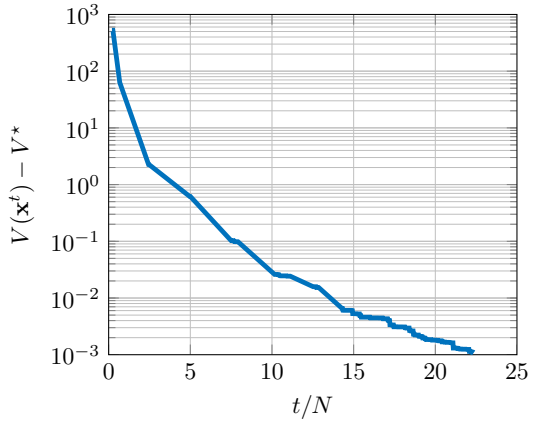


Figure 3.7: Evolution of the cost error, in logarithmic scale, for the Partitioned Coordinate Descent distributed algorithm.

Chapter 4

Distributed Big-Data Optimization via Block-Iterative Gradient Tracking and Averaging

In this chapter we study distributed big-data optimization over networks. We consider the (constrained) minimization of the sum of smooth (possibly nonconvex) functions, i.e., the sum of agent utilities, plus a convex (possibly) nonsmooth regularizer. We propose a novel distributed solution method where at each iteration agents perform local computations only on a small number of blocks of the entire decision variable. Asymptotic convergence to stationary solutions of the nonconvex problem is established and numerical results show the effectiveness of the proposed algorithm. The results of this chapter are based on [76, 77].

4.1 Literature Review

We organize the relevant literature of this chapter in two parts, namely: (i) centralized and parallel methods for big-data optimization (a.k.a. large- or huge-scale optimization), and (ii) primal distributed algorithms for optimization in peer-to-peer, multi-agent networks.

A coordinate-descent method for huge-scale optimization has been introduced in [66] and extended in order to deal with (convex) composite objective functions in a parallel framework in [58, 85, 86]. A parallel algorithm based on local Successive Convex Approximations (SCA) has been proposed in [33] to cope with nonconvex optimization problems. An asynchronous extension with its complexity analysis has been proposed in [21]. Paper [55] proposes a parallel stochastic-gradient algorithm where each processor randomly chooses a component of the separable cost function and updates only a random block of the decision variable. Finally, an asynchronous parallel mini-batch algorithm for stochastic optimization has been proposed in [35]. We point out that the above mentioned parallel approaches are not suited

for a distributed implementation.

In the last years the distributed computation paradigm has been widely investigated to solve optimization problems. We refer only to primal approaches, which are more closely related to the one we propose in this chapter. In [57] a coordinate descent approach for the solution of linearly constrained problems over undirected networks has been proposed. In [60] a broadcast-based algorithm over time-varying digraphs, termed subgradient-push, has been proposed, and extended to distributed online optimization in [2]. In [52] a distributed algorithm for convex optimization over time-varying networks, in the presence of constraints and uncertainty, using a proximal minimization approach, has been proposed. A distributed algorithm, termed NEXT, combining SCA techniques with a novel gradient tracking mechanism, instrumental to locally estimate the average of the agents' gradients, has been proposed in [29, 50] to solve nonconvex constrained optimization problems over time-varying networks. Even though NEXT can be applied to directed graphs, its implementability relies on the possibility of building a sequence of doubly-stochastic consensus matrices that are commensurate with the sequence of underlying time-varying communication digraphs. This assumption has been removed in [93] and [92], resulting in a more flexible algorithm. The gradient tracking scheme has been used also in [62, 84, 100, 102] where (strongly) convex unconstrained problems are consider under several communication topologies. In these works also the convergence rate of each distributed algorithm is proven for several constant step-size choices. A Newton-Raphson consensus strategy has been introduced in [105] to solve unconstrained, convex optimization problems under asynchronous, symmetric gossip communications. The same technique has been extended for designing an algorithm for directed, asynchronous and lossy communications networks in [23]. A distributed, asynchronous algorithm for constrained optimization based on random projections is proposed in [48]. We point out that none of the references above is able to solve big-data optimization problems in a distributed scenario.

4.2 Big-Data Problem Set-up and Assumptions

In this section we recall the big-data cost-coupled optimization problem introduced in Section 1.2.3. Let the decision vector $\mathbf{x} \in \mathbb{R}^{dB}$ be partitioned in B blocks as

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_B \end{bmatrix},$$

then, we consider a multiagent system of N nodes that want to solve the following (possibly) nonconvex optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & U(\mathbf{x}) \triangleq \sum_{i=1}^{N} f_i(\mathbf{x}) + \sum_{\ell=1}^{B} r_\ell(\mathbf{x}_\ell) \\ \text{subj. to } & \mathbf{x}_\ell \in \mathcal{K}_\ell, \quad \ell \in \{1,\dots,B\}, \end{aligned} \tag{4.1}$$

with each block $\mathbf{x}_\ell \in \mathbb{R}^d$, $\ell \in \{1,\dots,B\}$; $f_i : \mathbb{R}^{dB} \to \mathbb{R}$ is the cost function of agent i, assumed to be smooth but (possibly) nonconvex; $r_\ell : \mathbb{R}^d \to \mathbb{R}$, $\ell \in \{1,\dots,B\}$, is a convex (possibly nonsmooth) function; and $\mathcal{K}_\ell$, $\ell \in \{1,\dots,B\}$, is a closed convex set. In the following, we will denote by $\mathcal{K} \triangleq \mathcal{K}_1 \times \cdots \times \mathcal{K}_B$ the feasible set of (4.1). We study problem (4.1) under the following assumptions.

Assumption 4.2.1 (On the Optimization Problem)**.**

1. *Each $\mathcal{K}_\ell \neq \emptyset$ is closed and convex;*
2. *Each $f_i : \mathbb{R}^{dB} \to \mathbb{R}$ is C^1 on (an open set containing) $\mathcal{K}$;*
3. *Each ∇f_i is L_i-Lipschitz continuous and bounded on $\mathcal{K}$;*
4. *Each $r_\ell : \mathbb{R}^d \to \mathbb{R}$ is convex (possibly nonsmooth) on $\mathcal{K}$, with bounded subgradients over $\mathcal{K}$;*
5. *U is coercive on $\mathcal{K}$, i.e., $\lim_{\mathbf{x}\in\mathcal{K},\|\mathbf{x}\|\to\infty} U(\mathbf{x}) = \infty$.* □

The above assumptions are quite standard and satisfied by many practical problems; see, e.g. [33]. Here, we only remark that we do not assume any convexity of f_i. In the following, we also make the blanket assumption that each agent i knows only its own cost function f_i, the regularizers r_ℓ and the feasible set $\mathcal{K}$, but not the other agents' utility functions.

On the communication network: The communication network of the agents is modeled as a fixed, directed graph $\mathcal{G} = (\{1,\dots,N\}, \mathcal{E})$. We denote by $\mathcal{N}_i$ the set of *in-neighbors* of node i in the fixed graph $\mathcal{G}$, i.e., $\mathcal{N}_i \triangleq \{j \in \{1,\dots,N\} \mid (j,i) \in \mathcal{E}\}$. We assume that $\mathcal{E}$ contains self-loops and, thus, $\mathcal{N}_i$ contains $\{i\}$ itself. We use the following assumption.

Assumption 4.2.2. *The graph $\mathcal{G}$ is strongly connected.* □

We want to solve problem (4.1) in a distributed fashion, leveraging local communications among neighboring agents. As a major departure from current literature on distributed optimization, here we focus on *big-data* instances of problem (4.1) wherein the vector of variables $\mathbf{x}$ is composed of a huge number of components ($B \gg 1$ is very large and possibly depending on N). In such problems, minimizing the sum-utility with respect to the whole

$\mathbf{x}$, or even computing the gradient or evaluating the value of a single function f_i, can require substantial computational efforts. Moreover, exchanging an estimate of the *entire* local decision variable $\mathbf{x}$ over the network (like current distributed schemes do) is not efficient or even feasible, due to the excessive communication overhead.

4.3 Overview of Distributed Gradient Tracking Algorithms

In this section we recall existing distributed schemes that do *not* deal with big-data optimization but already employ the surrogate approach and a gradient tracking machinery. In order to introduce the main ideas, we will describe a distributed algorithm, called in-Network succEssive conveX approximaTion (NEXT), proposed in [29]. Other similar schemes have been proposed in the literature that work under different assumptions, such as convex costs or time-varying networks, see, e.g., [61, 84].

Consider a (low-dimensional) optimization problem in the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}) + r(\mathbf{x}) \\ \text{subj. to } & \mathbf{x} \in \mathcal{K}, \end{aligned} \tag{4.2}$$

where, differently from (4.1), the regularizer and the constraint do not have any specific dependence on the blocks of $\mathbf{x}$. The algorithm is based on an iterative two-step procedure where all the agents first update their local estimate $\mathbf{x}_{(i)}$ of the optimization variable $\mathbf{x}$ by solving a suitable convexification of (4.2), and then communicate with their neighbors to asymptotically force an agreement among the local variables while converging to a stationary solution of (4.2). Specifically, first, each agent i locally minimizes a strongly convex approximation of the original nonconvex function (subject to the same constraints as in (4.2)) having the following form: the local nonconvex cost function f_i is replaced by a suitable strongly convex *surrogate function* $\widetilde{f}_i$ (see [33]), while the sum of the unknown functions of the other agents $\sum_{j\neq i} f_j$ is approximated by a linear term whose coefficients track the gradient of $\sum_{j\neq i} f_j$. Formally, given the current iterate $\mathbf{x}_{(i)}^t$, the optimization step reads:

$$\begin{aligned} \widetilde{\mathbf{x}}_{(i)}^t &= \operatorname*{argmin}_{\mathbf{x}\in\mathcal{K}} \widetilde{f}_i(\mathbf{x}; \mathbf{x}_{(i)}^t) + (\mathbf{x} - \mathbf{x}_{(i)}^t)^\top \widetilde{\boldsymbol{\pi}}_{(i)}^t + r(\mathbf{x}) \\ \Delta\mathbf{x}_{(i)}^t &= \widetilde{\mathbf{x}}_{(i)}^t - \mathbf{x}_{(i)}^t \end{aligned}$$

where $\widetilde{\boldsymbol{\pi}}_{(i)}^t$ is a local estimate of $\sum_{j\neq i} \nabla f_j(\mathbf{x}_{(i)}^t)$ (that needs to be properly updated).

The second step of NEXT consists in communicating with the neighbors in order to update the local decision variables as well as the gradient estimates $\widetilde{\boldsymbol{\pi}}^t_{(i)}$. Formally, we have the following two consensus-based updates:

$$\begin{aligned}
\mathbf{x}^{t+1}_{(i)} &= \sum_{j=1}^{N} a^t_{ij}\Big(\mathbf{x}^t_{(j)} + \gamma^t \Delta \mathbf{x}^t_{(j)}\Big)\\
\mathbf{y}^{t+1}_{(i)} &= \sum_{j=1}^{N} a^t_{ij}\mathbf{y}^t_{(j)} + \nabla f_i(\mathbf{x}^{t+1}_{(i)}) - \nabla f_i(\mathbf{x}^t_{(i)})\\
\widetilde{\boldsymbol{\pi}}^{t+1}_{(i)} &= N\cdot \mathbf{y}^t_{(i)} - \nabla f_i(\mathbf{x}^{t+1}_{(i)}),
\end{aligned}$$

where γ^t is a (diminishing) step-size, $\mathbf{y}^t_{(i)}$ is an auxiliary local variable (exchanged among neighbors) instrumental to update $\widetilde{\boldsymbol{\pi}}^t_{(i)}$; and $\mathbf{A}^t \triangleq [a^t_{ij}]^N_{i,j=1}$ is a *doubly stochastic* matrix that matches the (possibly time-varying) communication graph. Specifically, $a^t_{ij} \in [\theta, 1]$, for some $\theta \in (0,1)$, if j sends information to i at time t, and $a^t_{ij} = 0$ otherwise.

Note that NEXT requires the weight matrices $\mathbf{A}^t$ to be doubly-stochastic, which limits the applicability of the algorithm to directed graphs that admit a doubly-stochastic matrix. This limitation has been overcome by the algorithm distributed Successive cONvex Approximation algorithm over Time-varying digrAphs (SONATA), proposed in [92, 93], where the plain consensus scheme has been replaced by a push-sum consensus [16], which recovers dynamic average consensus of the local decision variable $\mathbf{x}^t_{(i)}$ and gradient estimates $\mathbf{y}^t_{(i)}$ by using *column stochastic* weight matrices. Introducing an auxiliary local variable $\phi^t_{(i)}$ at each agent's side, the consensus protocol proposed in SONATA reads

$$\begin{aligned}
\phi^{t+1}_{(i)} &= \sum_{j=1}^{N} a^t_{ij}\phi^t_{(j)}\\
\mathbf{x}^{t+1}_{(i)} &= \sum_{j=1}^{N} \tfrac{a^t_{ij}\phi^t_{(j)}}{\phi^{t+1}_{(i)}}\Big(\mathbf{x}^t_{(j)} + \gamma \Delta \mathbf{x}^t_{(j)}\Big)\\
\mathbf{y}^{t+1}_{(i)} &= \frac{1}{\phi^{t+1}_{(i)}}\Big(\sum_{j=1}^{N} a^t_{ij}\phi^t_{(j)}\mathbf{y}^t_{(j)} + \nabla f_i(\mathbf{x}^{t+1}_{(i)}) - \nabla f_i(\mathbf{x}^t_{(i)})\Big)\\
\widetilde{\boldsymbol{\pi}}^{t+1}_{(i)} &= N\cdot \mathbf{y}^t_{(i)} - \nabla f_i(\mathbf{x}^{t+1}_{(i)}),
\end{aligned}$$

where $\phi^0_{(i)} = 1$ for all $i \in \{1, \ldots, N\}$ and $\mathbf{A}^t \triangleq [a^t_{ij}]^N_{i,j=1}$ is column stochastic.

4.4 Block-Iterative Gradient Tracking and Averaging Distributed Algorithm

In this section we introduce the new distributed big-data optimization algorithm. Differently from existing distributed methods, a distinctive feature of

this algorithm is that it performs *block-wise* updates and communications. A building block of the proposed scheme is a novel block-wise consensus protocol of independent interest, which is introduced in Section 4.4.1. Then, we will be ready to describe our new algorithm in Section 4.4.2.

4.4.1 Block-Wise Push-Sum Running Consensus

We propose a push-sum scheme that acts at the level of each block $\ell \in \{1,\dots,B\}$. Specifically, consider a system of N agents, whose communication network is modeled as a digraph $\mathcal{G}$ satisfying Assumption 4.2.2, aiming at reaching consensus on a common (possibly) time-varying signal.

While at each iteration t agents can update their entire vector $\mathbf{x}^t_{(i,:)}$, they can however send to their neighbors *only one block*; let us denote by $\mathbf{x}^t_{(i,\ell^t_i)}$ the block ℓ^t_i that, at time t, agent i selects (according to a suitably chosen rule) and sends to its neighbors, with $\ell^t_i \in \{1,\dots,B\}$ for all $t \geq 0$. Thus, at each iteration, agent i runs a perturbed push-sum protocol on the ℓ-th block by using only the information received from in-neighbors that sent block ℓ at time t (if any). A natural way to model this protocol is to introduce a *block-dependent* neighbor set, defined as

$$\mathcal{N}^t_{i,\ell} \triangleq \{j \in \mathcal{N}_i \mid \ell^t_j = \ell\} \cup \{i\} \subseteq \mathcal{N}_i,$$

which includes, besides agent i, only the in-neighbors of agent i in $\mathcal{G}$ that sent block ℓ at time t. Consistently, we denote by $\mathcal{G}^t_\ell \triangleq (\{1,\dots,N\}, \mathcal{E}^t_\ell)$ the *time-varying* subgraph of $\mathcal{G}$ associated to block ℓ at iteration t, with edge set given by

$$\mathcal{E}^t_\ell \triangleq \{(j,i) \in \mathcal{E} \mid j \in \mathcal{N}^t_{i,\ell}, i \in \{1,\dots,N\}\}.$$

Following the idea of consensus protocols over time-varying digraphs, we introduce a weight matrix $\mathbf{A}^t_\ell \triangleq [a^t_{ij\ell}]^N_{i,j=1}$ matching $\mathcal{G}^t_\ell$, such that $a^t_{ij\ell} > 0$ if $(j,i) \in \mathcal{E}^t_\ell$ and $a^t_{ij\ell} = 0$ otherwise. Using $\mathbf{A}^t_\ell$, we write the perturbed push-sum protocol for each block ℓ as

$$\begin{aligned} \phi^{t+1}_{(i,\ell)} &= \sum_{j \in \mathcal{N}^t_{i,\ell}} a^t_{ij\ell}\, \phi^t_{(j,\ell)} \\ \mathbf{x}^{t+1}_{(i,\ell)} &= \sum_{j \in \mathcal{N}^t_{i,\ell}} \frac{a^t_{ij\ell}\, \phi^t_{(j,\ell)}}{\phi^{t+1}_{(i,\ell)}} \mathbf{x}^t_{(j,\ell)}, \end{aligned} \tag{4.3}$$

for all $\ell \in \{1,\dots,B\}$, where $\phi^0_{(i,\ell)} = 1$ and $\mathbf{x}^0_{(i,\ell)}$ are given, for all $\ell \in \{1,\dots,B\}$ and $i \in \{1,\dots,N\}$.

We study now under which conditions the block-wise running consensus protocol (4.3) reaches an asymptotic agreement. For the push-sum algorithm to achieve asymptotic consensus, the following assumption is needed [16].

Assumption 4.4.1. *For all $\ell \in \{1, \dots, B\}$ and $t \geq 0$, the matrix $\mathbf{A}_\ell^t$ is column stochastic, that is, $\mathbf{1}^\top \mathbf{A}_\ell^t = \mathbf{1}^\top$.* □

We show next how nodes can *locally* build a matrix $\mathbf{A}_\ell^t$ satisfying Assumption 4.4.1 for each time-varying, directed graph $\mathcal{G}_\ell^t$. Since in our distributed optimization algorithm we work with a static, strongly connected digraph $\mathcal{G}$ (cf. Assumption 4.2.2), we assume that a column stochastic matrix $\tilde{\mathbf{A}}$ that matches $\mathcal{G}$ is available, i.e., $\tilde{a}_{ij} > 0$ if $(j, i) \in \mathcal{E}$ and $\tilde{a}_{ij} = 0$ otherwise, and $\mathbf{1}^\top \tilde{\mathbf{A}} = \mathbf{1}^\top$.

To show how $\mathbf{A}_\ell^t$ can be constructed in a distributed way, we start by observing that at iteration t, an agent j either sends a block ℓ to all its out-neighbors in $\mathcal{G}$, $\ell = \ell_j^t$, or to none, $\ell \neq \ell_j^t$. Thus, let us concentrate on the j-th column $\mathbf{A}_\ell^t(:, j)$ of $\mathbf{A}_\ell^t$. If agent j does not send block ℓ at iteration t, $\ell \neq \ell_j^t$, then all elements of $\mathbf{A}_\ell^t(:, j)$ will be zero except $a_{jj\ell}^t$. Thus, for the j-th column to be stochastic, it must be $\mathbf{1}^\top \mathbf{A}_\ell^t(:, j) = a_{jj\ell}^t = 1$ (i.e., $\mathbf{A}_\ell^t(:, j)$ is the j-th vector of the canonical basis). Viceversa, if j sends block ℓ, all its out-neighbors in $\mathcal{G}$ will receive it and, thus, column $\mathbf{A}_\ell^t(:, j)$ has the same nonzero entries as column $\tilde{\mathbf{A}}(:, j)$ of $\tilde{\mathbf{A}}$. Since $\tilde{\mathbf{A}}$ is column stochastic, the same entries can be chosen, that is, $\mathbf{A}_\ell^t(:, j) = \tilde{\mathbf{A}}(: j)$. This rule can be stated from the point of view of each agent i and its in-neighbors, thus showing that each agent can locally construct its own weights. For each $i \in \{1, \dots, N\}$ and $\ell \in \{1, \dots, B\}$, weights $a_{ij\ell}^t$ can be defined as

$$a_{ij\ell}^t \triangleq \begin{cases} \tilde{a}_{ij}, & \text{if } j \in \mathcal{N}_i \text{ and } \ell = \ell_j^t, \\ 1, & \text{if } j = i \text{ and } \ell \neq \ell_i^t, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Besides imposing $\mathbf{A}_\ell^t$ to be column stochastic for each t, another key aspect to achieve consensus is that the time-varying digraphs $\mathcal{G}_\ell^t$ be T-strongly connected, i.e., for all $t \geq 0$ the union digraph $\bigcup_{\tau=0}^{T-1} \mathcal{G}_\ell^{t+\tau}$ is strongly connected.

Since each (time-varying) digraph $\mathcal{G}_\ell^t$ is induced by the block selection rule, its connectivity properties are strictly related to the chosen rule. Thus, the T-strongly connectivity requirement imposes a condition on the way the blocks will be selected. The following general *essentially cyclic* rule is enough to guarantee that the time-varying digraph $\mathcal{G}_\ell^t$ is T-strongly connected.

Assumption 4.4.2 (Block Updating Rule)**.** *For each agent $i \in \{1, \dots, N\}$ there exists a (finite) constant $T_i > 0$ such that*

$$\bigcup_{\tau=0}^{T_i-1} \{\ell_i^{t+\tau}\} = \{1, \dots, B\}, \text{ for all } t \geq 0. \qquad \square$$

Note that the above rule does not impose any coordination among the agents, but agents selects their own block independently. Therefore, at the

same iteration, different agents may update different blocks. Moreover, some blocks can be updated more often than others. However, the rule guarantees that, within a finite time window of length $T \leq \max_{i\in\{1,\dots,N\}} T_i$, all the blocks have been updated at least once by all the agents. This is enough for $\mathcal{G}_\ell^t$ to be T-strongly connected, as formalized in the next proposition.

Proposition 4.4.3. *Under Assumptions 4.2.2 and 4.4.2, there exits a* $0 < T \leq \max_{i\in\{1,\dots,N\}} T_i$*, such that* $\bigcup_{\tau=0}^{T-1} \mathcal{G}_\ell^{t+\tau}$*,* $\ell \in \{1,\dots,B\}$*, is strongly connected, for all* $t \geq 0$*.*

Proof. Consider a particular block ℓ, and define $s_i^t(\ell)$ as the last time agent i sends block ℓ in the time window $[t, t+T-1]$, where $t \geq 0$. The essentially cyclic rule (cf. Assumption 4.4.2) implies that $s_i^t(\ell) \leq T-1$ for all $i \in \{1,\dots,N\}$. By definition of $\mathcal{G}_\ell^t$, we have that any edge $(j,i) \in \mathcal{E}$ also belongs to $\mathcal{G}_\ell^{t+s_i^t(\ell)}$. Since $(\{1,\dots,N\},\mathcal{E}) \subseteq \bigcup_{i\in\{1,\dots,N\}} \mathcal{G}_\ell^{t+s_i^t(\ell)} \subseteq \bigcup_{\tau=t}^{t+T-1} \mathcal{G}_\ell^{\tau}$, we have that $\bigcup_{\tau=0}^{T-1} \mathcal{G}_\ell^{t+\tau}$ is strongly connected, since $\mathcal{G}$ is so (cf. Assumption 4.2.2). □

Since $\{\mathcal{G}_\ell^t\}_{t\in\mathbb{N}}$ is T-strongly connected for all $\ell \in \{1,\dots,B\}$ and each $\mathbf{A}_\ell^t$ is a column stochastic matrix matching $\mathcal{G}_\ell^t$, a direct application of [60, Corollary 2] leads to the following convergence result for the matrix product $\mathbf{A}_\ell^{t+T:t} \triangleq \mathbf{A}_\ell^{t+T}\mathbf{A}_\ell^{t+T-1}\cdots\mathbf{A}_\ell^t$.

Proposition 4.4.4 ([60, Corollary 2])**.** *Suppose that Assumptions 4.2.2 and 4.4.2 hold true, and let* $\mathbf{A}_\ell^t$ *be defined as in* (4.4)*, with* $\ell \in \{1,\dots,B\}$*. Then, the matrix product* $\mathbf{A}_\ell^{t+T:t} \triangleq \mathbf{A}_\ell^{t+T}\mathbf{A}_\ell^{t+T-1}\cdots\mathbf{A}_\ell^t$ *satisfies the geometrical decay property, i.e., there exists a sequence of (real) stochastic vectors* $\{\boldsymbol{\xi}_\ell^t\}_{t\in\mathbb{N}}$ *such that*

$$|(\mathbf{A}_\ell^{t+T:t})_{ij} - (\boldsymbol{\xi}_\ell^t)_i| \leq c_\ell(\rho_\ell)^T,$$

for some $c_\ell > 0$ *and* $\rho_\ell \in (0,1)$*, where the notation* $(\rho_\ell)^T$ *means* ρ_ℓ *to the power of* T*.* □

Invoking Proposition 4.4.4, we can finally obtain the desired convergence result for the proposed block consensus protocol (4.3).

Theorem 4.4.5. *Consider the block consensus scheme (4.3), under Assumptions 4.2.2 and 4.4.2, and let each weight matrix* $\mathbf{A}_\ell^t$ *be defined according to* (4.4)*. Then, the following holds:*

$$\lim_{t\to\infty} \left\| \mathbf{x}_{(i,\ell)}^t - \sum_{i=1}^N \mathbf{x}_{(i,\ell)}^0 \right\| = 0, \qquad \text{for all } \ell \in \{1,\dots,B\}. \qquad \square$$

Remark 4.4.6. *In this chapter, we assumed to have a strongly connected, static directed network. However, more general time-varying communication*

topologies are highly desiderable. The only requirement that the network should always fulfill is that the combination of the block selection rule with the communication digraph preserves the T-strongly connectivity of the block induced graphs $\mathcal{G}_\ell^t$ for all $\ell \in \{1, \dots, B\}$. □

4.4.2 Algorithm Design

We are now in the position to introduce the novel distributed big-data optimization algorithm, which combines the proposed block-wise push-sum running consensus scheme with the successive convex approximations approach (suitably tailored to a block implementation). Specifically, each agent i maintains a set of local variables that will be iteratively updated along the algorithmic evolution, namely $\boldsymbol{\phi}_{(i,:)}$, $\mathbf{x}_{(i,:)}$ and $\mathbf{y}_{(i,:)}$ Consistently with the block structure of the optimization variable, also these states have a block structure (and can, thus, be updated block-wise). Thus, for example, we denote by $\mathbf{x}_{(i,\ell)}^t \in \mathbb{R}^d$ the ℓ-th block-component of local estimate $\mathbf{x}_{(i,:)}^t$ that agent i has at time t.

Informally, agent i performs a partial minimization only with respect to the block it selects. Then, agent i performs the block-wise consensus update introduced in the previous subsection. The Block Iterative Gradient Tracking and Averaging is formally reported in Distributed Algorithm 9 from the perspective of node i and discussed in detail afterwards. Each agent i initializes the local states as: $\mathbf{x}_{(i,:)}^0$ to an arbitrary value, $\boldsymbol{\phi}_{(i,:)}^0 = [1, \dots, 1]^\top \triangleq \mathbf{1}$, and $\mathbf{y}_{(i,:)}^0 = \nabla f_i(\mathbf{x}_{(i,:)}^0)$.

Let us comment the steps of the distributed algorithm. At each iteration t, each agent i selects a block $\ell_i^t \in \{1, \dots, B\}$ according to a rule satisfying Assumption 4.4.2. Then, a local approximation of problem (4.1) is constructed by replacing the nonconvex cost f_i with a strongly convex surrogate $\widehat{f}_{(i,\ell_i^t)}$ depending on the current estimates of $\mathbf{x}_{(i,:)}^t$, and on a gradient estimate $\mathbf{y}_{(i,\ell_i^t)}^t$ of the smooth part of the cost function, i.e., an estimate of $\sum_{i=1}^N \nabla f_i$. Notice that we use a slightly different notation for the surrogate from NEXT algorithm recalled in Section 4.3. We will describe next how the two expressions are related. Formally, in (4.5) agent i performs a local minimization of the surrogate function only with respect to $\ell = \ell_i^t$ and does not update the other blocks of $\mathbf{x}_{(i,:)}^t$. Then, it constructs a local descent direction as in (4.6).

After these updates, each node i sends to its out-neighbors $\mathbf{x}_{(i,\ell_i^t)}^t + \gamma^t \Delta \mathbf{x}_{(i,\ell_i^t)}^t$, where γ^t is a suitable diminishing step-size. We want to stress that this value together with $\mathbf{y}_{(i,\ell_i^t)}^t$ and $\phi_{(i,\ell_i^t)}^t$ are the sole quantities sent by agent i to its neighbors. Thus, Block Iterative Gradient Tracking and Averaging (Block-SONATA) not only involves the solution of low dimensional optimization problems, but also the transmission of a limited amount

Distributed Algorithm 9 Block Iterative Gradient Tracking and Averaging (Block-SONATA)

Local Optimization:

select $\ell_i^t \in \{1, \dots, B\}$ and compute

$$\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)} = \underset{\mathbf{x}_{\ell_i^t} \in \mathcal{K}_{\ell_i^t}}{\operatorname{argmin}} \ \widehat{f}_{i,\ell_i^t}\big(\mathbf{x}_{\ell_i^t}; \mathbf{x}^t_{(i,:)}, \mathbf{y}^t_{(i,\ell_i^t)}\big) + r_{\ell_i^t}(\mathbf{x}_{\ell_i^t}) \tag{4.5}$$

$$\Delta \mathbf{x}^t_{(i,\ell)} = \begin{cases} \widetilde{\mathbf{x}}^t_{(i,\ell_i^t)} - \mathbf{x}^t_{(i,\ell_i^t)}, & \text{if } \ell = \ell_i^t \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{4.6}$$

Averaging and Gradient Tracking:

FOR EACH $j \in \mathcal{N}_i$: receive $\phi^t_{(j,\ell_j^t)}$ and $\mathbf{x}^t_{(j,\ell_j^t)} + \gamma^t \Delta \mathbf{x}^t_{(j,\ell_j^t)}$

FOR EACH $\ell \in \{1, \dots, B\}$: compute

$$\phi^{t+1}_{(i,\ell)} = \sum_{j \in \mathcal{N}^t_{i,\ell}} a^t_{ij\ell}\, \phi^t_{(j,\ell)} \tag{4.7}$$

$$\mathbf{x}^{t+1}_{(i,\ell)} = \sum_{j \in \mathcal{N}^t_{i,\ell}} \frac{a^t_{ij\ell}\, \phi^t_{(j,\ell)}}{\phi^{t+1}_{(i,\ell)}} \Big(\mathbf{x}^t_{(j,\ell)} + \gamma^t \Delta \mathbf{x}^t_{(j,\ell)} \Big) \tag{4.8}$$

FOR EACH $j \in \mathcal{N}_i$: receive $\phi^t_{(j,\ell_j^t)}\, \mathbf{y}^t_{(j,\ell_j^t)} + \nabla_{\ell_j^t} f_j\big(\mathbf{x}^{t+1}_{(j,:)}\big) - \nabla_{\ell_j^t} f_j\big(\mathbf{x}^t_{(j,:)}\big)$

FOR EACH $\ell \in \{1, \dots, B\}$: compute

$$\mathbf{y}^{t+1}_{(i,\ell)} = \sum_{j \in \mathcal{N}^t_{i,\ell}} \frac{a^t_{ij\ell}}{\phi^{t+1}_{(i,\ell)}} \Big(\phi^t_{(j,\ell)}\, \mathbf{y}^t_{(j,\ell)} + \nabla_\ell f_j\big(\mathbf{x}^{t+1}_{(j,:)}\big) - \nabla_\ell f_j\big(\mathbf{x}^t_{(j,:)}\big) \Big) \tag{4.9}$$

of information.

As for the consensus step, agent i updates its local variables by means of the novel block-wise running consensus protocol described in Section 4.4.1. We point out once more that only the information received by the in-neighbors that have updated the corresponding block are used.

4.4.3 Algorithm Features and Alternative Formulations

A few comments about our distributed algorithm are in order at this point. First, we stress that a limited amount of data is transmitted at each iteration. Indeed, we point out that (although the notation in the consensus updates might hide this aspect) agents send *only one* block per iteration. Thus, each agent i receives exactly $|\mathcal{N}_i|$ updated quantities that are recombined by means of (4.8) and (4.9). In other words, the for-loop over ℓ in

fact consists of at most $|\mathcal{N}_i|$ non-trivial consensus steps. Moreover, due to the presence of the weights $a^t_{ij\ell}$, each non-trivial consensus step involves the sum of at most $|\mathcal{N}_i|$ terms over all the blocks.

As for the consensus iterations of the proposed algorithm, we notice that we can extrapolate the explicit push-sum scheme by introducing slack variables to obtain

$$\phi^{t+1}_{(i,\ell)} = \sum_{j\in\mathcal{N}_i} a^t_{ij\ell}\, \phi^t_{(j,\ell)} \tag{4.10}$$

$$\mathbf{s}^{t+1}_{(i,\ell)} = \sum_{j\in\mathcal{N}_i} a^t_{ij\ell}\Big(\mathbf{s}^t_{(j,\ell)} + \gamma^t \phi^t_{(j,\ell)}\Delta\mathbf{x}^t_{(j,\ell)}\Big) \tag{4.11}$$

$$\mathbf{x}^{t+1}_{(i,\ell)} = \frac{\mathbf{s}^{t+1}_{(i,\ell)}}{\phi^{t+1}_{(i,\ell)}} \tag{4.12}$$

$$\boldsymbol{\sigma}^{t+1}_{(i,\ell)} = \sum_{j\in\mathcal{N}_i} a^t_{ij\ell}\Big(\boldsymbol{\sigma}^t_{(j,\ell)} + \nabla_\ell f_j\big(\mathbf{x}^{t+1}_{(j,:)}\big) - \nabla_\ell f_j\big(\mathbf{x}^t_{(j,:)}\big)\Big) \tag{4.13}$$

$$\mathbf{y}^{t+1}_{(i,\ell)} = \frac{\boldsymbol{\sigma}^{t+1}_{(i,\ell)}}{\phi^{t+1}_{(i,\ell)}}, \tag{4.14}$$

for all $\ell \in \{1,\dots,B\}$. Notice that the above scheme is an equivalent formulation of (4.7), (4.8) and (4.9) that is more convenient for the analysis. In Section 4.5 we will study the convergence of Block Iterative Gradient Tracking and Averaging relying on the push-sum formulation of the consensus protocol.

We also point out that (4.12) and (4.14) are performed in the so-called Adapt-Then-Combine scheme (ATC). However, they could be also performed in the Combine-Then-Adapt (CTA) way. Formally, for all $\ell \in \{1,\dots,B\}$, agent i can perform

$$\mathbf{x}^{t+1}_{(i,\ell)} = \sum_{j\in\mathcal{N}^t_{i,\ell}} \frac{a^t_{ij\ell}\phi^t_{(j,\ell)}}{\phi^{t+1}_{(j,\ell)}}\, \mathbf{x}^t_{(j,\ell)} + \gamma^t \phi^t_{(i,\ell)}\Delta\mathbf{x}^t_{(i,\ell)}$$

$$\mathbf{y}^{t+1}_{(i,\ell)} = \sum_{j\in\mathcal{N}^t_{i,\ell}} \frac{a^t_{ij\ell}\,\phi^t_{(j,\ell)}}{\phi^{t+1}_{(i,\ell)}}\mathbf{y}^t_{(j,\ell)} + \frac{\nabla_\ell f_i\big(\mathbf{x}^{t+1}_{(i,:)}\big) - \nabla_\ell f_i\big(\mathbf{x}^t_{(i,:)}\big)}{\phi^{t+1}_{(i,\ell)}}.$$

Moreover, it is worth noting that in order to perform (4.14) (and also its CTA counterpart), agent i needs to compute the entire gradient $\nabla f_i(\mathbf{x}^{t+1}_{(i,:)})$ (recall from (4.4) that $a_{ii\ell} > 0$ for all ℓ). This possible drawback may be avoided by a slightly different version of Block Iterative Gradient Tracking and Averaging in which $\nabla_\ell f_i$ is replaced by an auxiliary variable $\widehat{\mathbf{g}}_{(i,\ell)}$, which

is iteratively updated as

$$\widehat{\mathbf{g}}_{(i,\ell)}^{t+1} = \begin{cases} \nabla_{\ell_i^t} f_i\big(\mathbf{x}_{(i,:)}^{t+1}\big), & \text{if } \ell = \ell_i^t, \\ \widehat{\mathbf{g}}_{(i,\ell)}^{t}, & \text{otherwise,} \end{cases}$$

$$\mathbf{y}_{(i,\ell)}^{t+1} = \sum_{j \in \mathcal{N}_{i,\ell}^t} \frac{a_{ij\ell}^t \phi_{(j,\ell)}^t}{\phi_{(i,\ell)}^{t+1}} \mathbf{y}_{(j,\ell)}^t + \frac{\widehat{\mathbf{g}}_{(i,\ell)}^{t+1} - \widehat{\mathbf{g}}_{(i,\ell)}^{t}}{\phi_{(i,\ell)}^{t+1}}.$$

An important property of Block Iterative Gradient Tracking and Averaging (inherited by its non-big-data counterpart SONATA) is the recursive feasibility of each sequence $\mathbf{x}_{(i,:)}^t$. Indeed, from the update (4.12) of the local solution estimate, we notice that, being $\Delta\mathbf{x}_{(j,\ell)}^t$ a feasible (descent) direction, if $\mathbf{x}_{(j,\ell)}^t$ is feasible for the constraint $\mathcal{K}_\ell$ then also the point $\mathbf{x}_{(j,\ell)}^t + \gamma^t \Delta\mathbf{x}_{(j,\ell)}^t$ is feasible for $\mathcal{K}_\ell$. Moreover, since the weights involved in the consensus mixing, $a_{ij\ell}^t \phi_{(j,\ell)}^t / \phi_{(i,\ell)}^{t+1}$, $j \in \mathcal{N}_i$, sum up to one, then also the new iterate $\mathbf{x}_{(i,\ell)}^{t+1}$ is feasible for $\mathcal{K}_\ell$.

Finally, we describe a special instances of Block Iterative Gradient Tracking and Averaging. Consider an unconstrained version of problem (4.1) with $r_\ell = 0$ for all ℓ, and suppose one chooses the simplest surrogate in (4.12), namely the linearization of f_i, then Block Iterative Gradient Tracking and Averaging boils down to

$$\phi_{(i,\ell)}^{t+1} = \sum_{j \in \mathcal{N}_{i,\ell}^t} a_{ij\ell}^t \, \phi_{(j,\ell)}^t$$

$$\mathbf{x}_{(i,\ell)}^{t+1} = \sum_{j \in \mathcal{N}_{i,\ell}^t} \frac{a_{ij\ell}^t \, \phi_{(j,\ell)}^t}{\phi_{(i,\ell)}^{t+1}} \Big(\mathbf{x}_{(j,\ell)}^t - \gamma^t \mathbf{y}_{(j,\ell)}^t \Big)$$

$$\mathbf{y}_{(i,\ell)}^{t+1} = \sum_{j \in \mathcal{N}_{i,\ell}^t} \frac{a_{ij\ell}^t}{\phi_{(i,\ell)}^{t+1}} \Big(\phi_{(j,\ell)}^t \, \mathbf{y}_{(j)}^t + \nabla_\ell f_j(\mathbf{x}_{(j,:)}^{t+1}) - \nabla_\ell f_j(\mathbf{x}_{(j,:)}^t) \Big),$$

which is the block-wise implementation of existing distributed (non-big-data) algorithms, as e.g., [62].

To conclude this subsection, we want to stress a key aspect of Block Iterative Gradient Tracking and Averaging related to the block-dependent communication graph. Even if the starting communication network is static and undirected, we cannot escape from the block-wise push-sum running consensus scheme. In fact, the block selection rule will still give rise to a *directed* time-varying graph.

4.4.4 Algorithm Convergence Analysis

In this subsection we discuss the assumptions under which Block Iterative Gradient Tracking and Averaging converges, together with the formal convergence theorem. Specifically, under mild (quite standard) requirements

on the surrogate functions $\widehat{f}_{i,\ell}$ [cf. (4.5)] and the step-size sequence $\{\gamma^t\}_{t\geq 0}$ the algorithm is guaranteed to converge.

The surrogate function $\widehat{f}_{i,\ell}(\bullet; \mathbf{x}, \mathbf{g}) : \mathbb{R}^d \to \mathbb{R}$ is a function parameterized by $\mathbf{x} \in \mathbb{R}^{dB}$ and $\mathbf{g} \in \mathbb{R}^{dB}$ for all $i \in \{1, \ldots, N\}$ and $\ell \in \{1, \ldots, B\}$. Roughly speaking, $\mathbf{x}$ is the point at which we approximate the function f_i while $\mathbf{g}$ contains the first-order information assigned to $\widehat{f}_{i,\ell}$. To further highlight the properties of the surrogate, we express it as the sum of two terms as follows

$$\widehat{f}_{i,\ell}(\mathbf{x}_\ell; \mathbf{x}^t_{(i,:)}, \mathbf{g}^t) = \widetilde{f}_{i,\ell}(\mathbf{x}_\ell, \mathbf{x}^t_{(i,:)}) + \left(\mathbf{g}^t - \nabla_\ell f_i(\mathbf{x}^t_i)\right)^\top (\mathbf{x}_\ell - \mathbf{x}^t_{(i,\ell)}) \tag{4.15}$$

where the function $\widetilde{f}_{i,\ell}$ satisfies the following properties.

Assumption 4.4.7 (On the surrogate functions)**.** *Given problem* (4.1) *under Assumption 4.2.1, each surrogate function $\widetilde{f}_{i,\ell} : \mathcal{K}_\ell \times \mathcal{K} \to \mathbb{R}$ is chosen such that*

1. *$\widetilde{f}_{i,\ell}(\bullet; \mathbf{x})$ is strongly convex with constant $\tau_i > 0$ on $\mathcal{K}_\ell$ for all $\mathbf{x} \in \mathbb{R}^{dB}$;*

2. *$\nabla \widetilde{f}_{i,\ell}(\mathbf{x}_\ell; \mathbf{x}) = \nabla_\ell f_i(\mathbf{x})$, for all $\mathbf{x} \in \mathcal{K}$;*

3. *$\nabla \widetilde{f}_{i,\ell}(\mathbf{x}_\ell; \bullet)$ is Lipschitz continuous on $\mathcal{K}$ for all $\mathbf{x}_\ell \in \mathbb{R}^d$;*

where the notation $\nabla \widetilde{f}_{i,\ell}$ denotes the partial gradient of $\widetilde{f}_{i,\ell}$ with respect to its first argument. □

The conditions above are mild assumptions. Condition (iii) is a simple Lipschitzianity requirement that is readily satisfied if, for example, the set $\mathcal{K}$ is bounded. In general, $\widehat{f}_{i,\ell}$ should be regarded as a (simple) convex, local, approximation of f_i that preserves the first-order properties of f_i at the point $\mathbf{x}$ and encodes additional first-order information specified in the second parameter.

Several valid instances of $\widehat{f}_{i,\ell}$ are possible for a given f_i; the appropriate one depends on the problem at hand and on the computational requirements. Detailed examples can be found in [29, 33]. As for the step-size γ^t, we need the following.

Assumption 4.4.8 (On the step-size)**.** *The sequence $\{\gamma^t\}_{t\geq 0}$, with each $0 < \gamma^t \leq 1$, satisfies:*

(i) $\gamma^{t+1} \leq \gamma^t$, for all $t \geq 0$;

(ii) $\sum_{t=0}^{\infty} \gamma^t = \infty$ and $\sum_{t=0}^{\infty} (\gamma^t)^2 < \infty$. □

Conditions (ii) are standard and satisfied by most practical diminishing step-size rules. The upper bound condition in (i) just states that the sequence is nonincreasing whereas the lower bound condition dictates that $\sum_{t=0}^{\kappa} \gamma^t \geq \gamma^0(\eta^0 + \eta^1 + \cdots + \eta^\kappa)$, that is, the partial sums $\sum_{t=0}^{\kappa} \gamma^t$ must be lower-bounded by a *convergent* geometric series. This impose a maximum decay rate to $\{\gamma^t\}$. Given that $\sum_{t=0}^{\infty} \gamma^t = +\infty$, the aforementioned requirement is clearly very mild and indeed it is also satisfied by most classical diminishing stepsize rules. For example, the following rule, proposed in [33], satisfies Assumption 5.2.3 and has been found to be very effective in our experiments: $\gamma^{t+1} = \gamma^t \left(1 - \mu\gamma^t\right)$, with $\gamma^0 \in (0, 1]$ and $\mu \in (0, 1)$.

We are now in the position to state the main convergence result, as given below.

Theorem 4.4.9. *Let* $\left\{\mathbf{x}^t_{(1,:)}, \ldots, \mathbf{x}^t_{(N,:)}\right\}_{t\geq 0}$ *be the sequence generated by* Block-SONATA *distributed algorithm, and consider their weighted average having components*

$$\bar{\boldsymbol{s}}^t_\ell = \frac{1}{N} \sum_{i=1}^{N} \phi^t_{(i,\ell)} \mathbf{x}^t_{(i,\ell)}$$

for all $\ell \in \{1, \ldots, B\}$*. Suppose that Assumptions 4.2.1, 4.2.2, 4.4.1 4.4.7, and 4.4.8 are satisfied; then the following holds:*

1. `consensus`: $\|\mathbf{x}^t_{(i,:)} - \bar{\boldsymbol{s}}^t\| \to 0$ *as* $t \to \infty$*, for all* $i \in \{1, \ldots, N\}$*;*
2. `convergence`: $\{\bar{\boldsymbol{s}}^t\}_{t\geq 0}$ *is bounded, and every of its limit points is a stationary solution of problem* (4.1). □

Section 4.5 is dedicated to the proof of this theorem. To conclude this section we comment on its twofold result. First, a consensus is asymptotically achieved among the local estimates $\mathbf{x}^t_{(i,:)}$ over all the blocks. Second, the weighted average estimate $\bar{\boldsymbol{s}}^t$ converges to the set $\mathcal{S}$ of stationary solutions of problem (4.1). Therefore, the sequence $\{\mathbf{x}^t_{(1,:)}, \ldots, \mathbf{x}^t_{(N,:)}, \}_{t\geq 0}$ converges to the set $\{\mathbf{1}_N \otimes \mathbf{x}^* : \mathbf{x}^* \in \mathcal{S}\}$. In particular, if U in (4.1) is convex, Block Iterative Gradient Tracking and Averaging converges (in the aforementioned sense) to the set of global optimal solutions of the convex problem.

Remark 4.4.10. *It is worth pointing out that one can consider also a time-varying digraph* $\mathcal{G}^t$ *connecting the agents. The block-selection rule should be combined to a proper long-term connectivity assumption of* $\mathcal{G}^t$ *that guarantees that the induced graphs associated to each block are* T*-strongly connected.* □

4.5 Convergence Analysis

The proof consists of several stages that are organized in subsections. We start with some preliminaries on the perturbed push-sum. Then we prove

that the local estimates of the agents are asymptotically consensual to their average and, finally we derive a series of results to show that any limit point of the average sequence of the primal variables is a stationary solution for problem (4.1).

4.5.1 Preliminaries on the Perturbed Push-Sum Consensus

Consider the generic perturbed push-sum protocol

$$\begin{aligned} \psi_i^{t+1} &= \sum_{j \in \mathcal{N}_i^t} a_{ij}^t \psi_j^t \\ \eta_i^{t+1} &= \sum_{j \in \mathcal{N}_i^t} a_{ij}^t (\eta_j^t + \epsilon_j^t) \\ z_i^{t+1} &= \frac{\eta_i^{t+1}}{\psi_i^{t+1}} \end{aligned} \tag{4.16}$$

where a_{ij}^t are entries of a column stochastic matrix $\mathbf{A}^t$ while the variables ψ_i, η_i and z_i are scalar $i \in \{1, \dots, N\}$. Moreover, $\psi_i^0 = 1$ for all $i \in \{1, \dots, N\}$.

Lemma 4.5.1 ([60, Lemma 1])**.** *Consider the protocol* (4.16)*, then it holds*

(a)

$$\left| z_i^{t+1} - \frac{1}{N} \sum_{j=1}^{N} (\eta_j^t + \epsilon_j^t) \right| \leq c_1 (\rho)^t + c_2 \sum_{\tau=1}^{t} (\rho)^{t-\tau} \sum_{i=1}^{N} |\epsilon_i^\tau| \tag{4.17}$$

where $\rho \in (0, 1)$ and c_1 and c_2 are positive scalars;

(b) if the perturbations are vanishing, i.e., $\lim_{t \to \infty} \epsilon_i^t = 0$, then

$$\lim_{t \to \infty} \left| z_i^{t+1} - \frac{1}{N} \sum_{i=1}^{N} (\eta_i^t + \epsilon_i^t) \right| = 0$$

□

The previous lemma can be generalized when a vector form of the push-sum protocol is considered. The following extension of (4.17) holds

$$\left\| \mathbf{z}_i^{t+1} - \bar{\boldsymbol{\eta}}^{t+1} \right\| \leq c_1 (\rho)^t + c_2 \sum_{\tau=1}^{t} (\rho)^{t-\tau} \sum_{i=1}^{N} \|\boldsymbol{\epsilon}_i^\tau\|_1, \tag{4.18}$$

where we set $\bar{\boldsymbol{\eta}}^{t+1} = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{\eta}_j^{t+1} = \frac{1}{N} \sum_{j=1}^{N} (\boldsymbol{\eta}_j^t + \boldsymbol{\epsilon}_j^t)$.

4.5.2 Consensus Achievement

In this subsection we prove that the estimates $\mathbf{x}_{(i,:)}^t$ reach asymptotic consensus to their weighted average $\bar{\boldsymbol{s}}^t$. Moreover, we study the consensus dynamics

of the gradient tracking $\mathbf{y}^t_{(i,:)}$. We start by writing the algorithmic evolution of the average of the decision variable estimates $\bar{\boldsymbol{s}}^t$ and the gradient tracking estimates $\bar{\boldsymbol{\sigma}}^t$. Starting from (4.5)-(4.6) and (4.10)-(4.14), it is not difficult to show that the following holds true

$$\bar{\boldsymbol{s}}^{t+1}_\ell = \bar{\boldsymbol{s}}^t_\ell + \gamma^t \frac{1}{N} \sum_{i=1}^N \phi^t_{(i,\ell)} \Delta \mathbf{x}^t_{(i,\ell)} \tag{4.19}$$

$$\bar{\boldsymbol{\sigma}}^{t+1}_\ell = \bar{\boldsymbol{\sigma}}^t_\ell + \sum_{i=1}^N \Big(\nabla_\ell f_i(\mathbf{x}^{t+1}_{(i,:)}) - \nabla_\ell f_i(\mathbf{x}^t_{(i,:)}) \Big). \tag{4.20}$$

In the following we prove two preparatory results which represent uniform upper bounds on the gradient tracking sequence and on the local descent directions.

Lemma 4.5.2. *Consider the sequences generated by* Block-SONATA. *Then, the following holds*

1. *for all* $\ell \in \{1, \dots, B\}$, $i \in \{1, \dots, N\}$

$$\sup_{t \geq 0} \left\| \mathbf{y}^t_{(i,\ell)} - \bar{\boldsymbol{\sigma}}^t_\ell \right\| < C_1, \tag{4.21}$$

   with C_1 *positive, finite scalar;*

2. *for all* $\ell \in \{1, \dots, B\}$, $i \in \{1, \dots, N\}$

$$\sup_{t \geq 0} \left\| \Delta \mathbf{x}^t_{(i,\ell)} \right\| < C_2, \tag{4.22}$$

   with C_2 *positive, finite scalar.*

Proof. As for the uniform bound on the gradient tracking, it follows by noticing that the gradient tracking scheme described in (4.10), (4.13) and (4.14) is a perturbed push-sum. Thus, we can invoke Lemma 4.5.1 (cf. eq. (4.18)) to conclude that

$$\left\| \mathbf{y}^t_{(i,\ell)} - \bar{\boldsymbol{\sigma}}^t_\ell \right\| \leq c_1 (\rho)^{t-1} + \sum_{\tau=1}^{t-1} (\rho)^{t-1-\tau} \sum_{i=1}^N \| \nabla_\ell f_i(\mathbf{x}^{\tau+1}_{(i,:)}) - \nabla_\ell f_i(\mathbf{x}^\tau_{(i,:)}) \|_1, \tag{4.23}$$

with $\rho \in (0, 1)$. Since by Assumption 4.2.1 ∇f_i is bounded, then also the forcing term $\| \nabla_\ell f_i(\mathbf{x}^{\tau+1}_{(i,:)}) - \nabla_\ell f_i(\mathbf{x}^\tau_{(i,:)}) \|$ is uniformly bounded for all i and τ, and thus condition (4.21) follows.

The condition 2. is characterizing the local descent direction $\Delta \mathbf{x}_{(i,\ell)}$ that each agent computes. For $\ell \neq \ell^t_i$ it is equal to zero implying trivially (4.22). For $\ell = \ell^t_i$, we notice that $\widetilde{\mathbf{x}}^t_{i,\ell}$ is the solution of the composite strongly convex optimization problem (4.5), thus it holds (see Lemma 4.5.10 in Section 4.5.6)

$$\| \Delta \mathbf{x}^t_{(i,\ell^t_i)} \| \leq \frac{N}{\tau_i} \| \mathbf{y}^t_{(i,\ell^t_i)} \| + \frac{B_r}{\tau_i}.$$

where we used the fact that the gradient of $\widehat{f}_{i,\ell}$ is $\mathbf{y}^t_{(i,\ell)}$ (cf. Assumption 4.4.7) and the boundedness of the subgradients $\widetilde{\nabla} r_\ell$ (cf. Assumption 4.2.1). By adding and subtracting suitable terms

$$\begin{aligned}\|\Delta \mathbf{x}^t_{(i,\ell^t_i)}\| &\leq \frac{N}{\tau_i}\|\mathbf{y}^t_{(i,\ell^t_i)} - \bar{\boldsymbol{\sigma}}^t_{(i,\ell^t_i)}\| + \|\bar{\boldsymbol{\sigma}}^t_{(i,\ell^t_i)}\| + \frac{B_r}{\tau_i}. \\ &\leq \frac{N}{\tau_i}\sum_{\ell=1}^{B}\|\mathbf{y}^t_{(i,\ell)} - \bar{\boldsymbol{\sigma}}^t_{(i,\ell)}\| + \Big\|\sum_{i=1}^{N}\nabla_{\ell^t_i} f_i(\mathbf{x}^t_{(i,:)})\Big\| + \frac{B_r}{\tau_i}.\end{aligned}$$

and using (4.21) and the boundedness of ∇f_i (cf. Assumption 4.2.1), condition (4.22) follows, concluding the proof. □

In the following proposition we characterize the consensual behavior of the decision variables. We point out that the following discussion is independent of the block-wise updates of our distributed algorithm and relies mainly on the network connectivity properties induced by the block selection rule.

Proposition 4.5.3. *Consider the sequences generated by* Block-SONATA. *Then, the decision variables are asymptotically consensual to $\bar{\boldsymbol{s}}^t$, i.e.,*

$$\lim_{t\to\infty}\|\mathbf{x}^t_{(i,:)} - \bar{\boldsymbol{s}}^t\| = 0 \tag{4.24}$$

for all $i \in \{1,\dots,N\}$. Moreover, the following summability conditions hold true

$$\sum_{t=0}^{\infty}\gamma^t\|\mathbf{x}^t_{(i,:)} - \bar{\boldsymbol{s}}^t\| < \infty \tag{4.25}$$

$$\sum_{t=0}^{\infty}\|\mathbf{x}^t_{(i,:)} - \bar{\boldsymbol{s}}^t\|^2 < \infty \tag{4.26}$$

Proof. It is sufficient to prove the previous conditions (4.24), (4.25) and (4.26) for each block ℓ. Notice that the algorithmic evolution of $\mathbf{x}^t_{(:,\ell)}$ is a perturbed push-sum. By Lemma. 4.5.2 and since the step-size γ^t is vanishing, then each perturbation $\boldsymbol{\epsilon}^t_{i,\ell} = \gamma^t\phi^t_{(i,\ell)}\Delta\mathbf{x}^t_{(i,\ell)}$ is vanishing. Thus, we can invoke Lemma 4.5.1 (b) to conclude that

$$\lim_{t\to\infty}\|\mathbf{x}^t_{(i,\ell)} - \bar{\boldsymbol{s}}^t_\ell\| = 0 \tag{4.27}$$

As for proving (4.25), we use Lemma 4.5.1 (a) (cf. eq. (4.18) to write

$$\sum_{t=0}^{\infty}\gamma^{t+1}\|\mathbf{x}^{t+1}_\ell - \bar{\boldsymbol{s}}^{t+1}_\ell\| \leq \sum_{t=0}^{\infty}\gamma^{t+1}\Big(c_1(\rho)^t + c_2\sum_{\tau=1}^{t}(\rho)^{t-\tau}\gamma^\tau\|\Delta\mathbf{x}^t_{(:,:)}\|_1\Big) \tag{4.28}$$

and use boundedness of $\|\Delta\mathbf{x}^t_{(:,:)}\|_1$ with [65, Lemma 7] to conclude the summability.

Finally, by using the same approach as before, we have

$$\begin{aligned}
&\sum_{t=0}^{\infty} \|\mathbf{x}_\ell^{t+1} - \bar{\boldsymbol{s}}_\ell^{t+1}\|^2 \\
&\leq \sum_{t=0}^{\infty} \Big(c_1(\rho)^t + c_3 \sum_{\tau=1}^{t} \rho^{t-\tau}\gamma^\tau\Big)^2 \\
&= \sum_{t=0}^{\infty} \Big(c_1^2(\rho)^{2t} + c_3^2 \sum_{\tau=1}^{t}\sum_{s=1}^{t} \gamma^\tau\gamma^s(\rho)^{t-\tau}(\rho)^{t-s} + 2c_1c_3 \sum_{\tau=0}^{t} \gamma^\tau(\rho)^{t-\tau}(\rho)^t\Big).
\end{aligned}$$

We can invoke [29, Lemma 7] to conclude that the consensus error is square summable, i.e., (4.26) holds true, and the proof follows. □

It is worth noting at this point that Proposition 4.5.3 (cf. eq. (4.24)) proves the first statement of Theorem 5.2.4. In the following result we characterize the gradient tracking scheme.

Proposition 4.5.4. *Consider the sequences generated by* Block-SONATA. *Then, the gradient tracking error is vanishing, i.e.,*

$$\lim_{t\to\infty} \Big\|\mathbf{y}_{(i,:)}^t - \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\mathbf{x}_{(i,:)}^t)\Big\| = 0, \tag{4.29}$$

for all $i \in \{1,\dots,N\}$.

Proof. It is sufficient to prove (4.5.4) for each block ℓ. Since we initialize the gradient tracker as $\mathbf{y}_{(i,\ell)}^0 = \nabla_\ell f_i(\mathbf{x}_{(i,:)}^0)$, for all $i \in \{1,\dots,N\}$ and $\ell \in \{1,\dots,B\}$, it holds

$$\frac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\mathbf{x}_{(j,:)}^t) = \bar{\boldsymbol{\sigma}}_\ell^t$$

for all $\ell \in \{1,\dots,B\}$.

Since the gradient tracking scheme is a perturbed push-sum, we can invoke Lemma 4.5.1 and write

$$\begin{aligned}
&\big\|\mathbf{y}_{(i,\ell)}^t - \frac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\mathbf{x}_{(j,:)}^t)\big\| = \big\|\mathbf{y}_{(i,\ell)}^t - \bar{\boldsymbol{\sigma}}_\ell^t\big\| \\
&\leq c_1(\rho)^{t-1} + c_2 \sum_{\tau=1}^{t-1} (\rho)^{t-1-\tau} \sum_{i=1}^{N} \big\|\nabla_\ell f_i(\mathbf{x}_{(i,:)}^{\tau+1}) - \nabla_\ell f_i(\mathbf{x}_{(i,:)}^{\tau})\big\| \qquad (4.30)\\
&\overset{(a)}{\leq} c_1(\rho)^{t-1} + c_3 \sum_{\tau=1}^{t-1} (\rho)^{t-1-\tau} \sum_{i=1}^{N} \big\|\mathbf{x}_{(i,:)}^{\tau+1} - \mathbf{x}_{(i,:)}^{\tau}\big\|_1
\end{aligned}$$

where in (a) we used the Lipschitz continuity of each ∇f_i (Assumption 4.2.1). Then, we notice that

$$\begin{aligned}
\|\mathbf{x}_{(i,:)}^{\tau+1} - \mathbf{x}_{(i,:)}^{\tau}\| &\leq \|\mathbf{x}_{(i,:)}^{\tau+1} - \bar{\boldsymbol{s}}^{\tau+1}\| + \|\mathbf{x}_{(i,:)}^{\tau} - \bar{\boldsymbol{s}}^{\tau}\| + \|\bar{\boldsymbol{s}}^{\tau+1} - \bar{\boldsymbol{s}}^{\tau}\| \\
&\leq \|\mathbf{x}_{(i,:)}^{\tau+1} - \bar{\boldsymbol{s}}^{\tau+1}\| + \|\mathbf{x}_{(i,:)}^{\tau} - \bar{\boldsymbol{s}}^{\tau}\| + \gamma^\tau \sum_{\ell=1}^{B}\sum_{i=1}^{N} \|\Delta\mathbf{x}_{i,\ell}^\tau\|
\end{aligned} \tag{4.31}$$

and invoking Proposition 4.5.3 (cf. eq. (4.27)) and Lemma 4.5.2 (ii) and since $\gamma^t \to 0$ (cf. Assumption 5.2.3), we can conclude $\lim_{t\to\infty} \|\mathbf{x}^{\tau+1}_{(i,:)} - \mathbf{x}^{\tau}_{(i,:)}\|_1 = 0$. Thus, we can invoke [65, Lemma 7] to conclude the proof. □

4.5.3 Cost Descent on the Average of the Decision Estimates

Once we have proven that all the agents are asymptotically consensual to $\bar{\boldsymbol{s}}^t$, we can restrict our attention to the convergence properties of the sequence $\{\bar{\boldsymbol{s}}^t\}_{t\geq 0}$.

In this subsection we show that the function V^t given by

$$V^t \triangleq \sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^{t+1}) + \sum_{i=1}^{N}\sum_{\ell=1}^{B} \phi^t_{(i,\ell)} r_\ell(\mathbf{x}^t_{(i,\ell)}) \tag{4.32}$$

satisfies a descent condition along the algorithmic evolution. We start by focusing on the regularizer r in the cost.

Proposition 4.5.5. *Consider the sequences generated by* Block-SONATA. *Then, the regularization term satisfies*

$$\sum_{i=1}^{N} \phi^{t+1}_{(i,\ell)} r_\ell(\mathbf{x}^{t+1}_{(i,\ell)}) - \sum_{i=1}^{N} \phi^t_{(i,\ell)} r_\ell(\mathbf{x}^t_{(i,\ell)}) \leq \gamma^t \frac{1}{N} \sum_{i=1}^{N} \phi^t_{(i,\ell)} \Big(r_\ell(\widetilde{\mathbf{x}}^t_{(i,\ell)}) - r_\ell(\mathbf{x}^t_{(i,\ell)}) \Big) \tag{4.33}$$

for all $\ell \in \{1, \dots, B\}$.

Proof. From the evolution of $\mathbf{x}^{t+1}_{(i,\ell)}$ and $\phi^{t+1}_{(i,\ell)}$ in Block-SONATA, we have

$$\begin{aligned}
\sum_{i=1}^{N} \phi^{t+1}_{(i,\ell)}\, r_\ell(\mathbf{x}^{t+1}_{(i,\ell)}) &= \sum_{i=1}^{N}\sum_{j=1}^{N} a^t_{ij\ell}\phi^t_{(j,\ell)}\, r_\ell\Big(\tfrac{1}{\phi^{t+1}_{(i,\ell)}} \sum_{j=1}^{N} a^t_{ij\ell}\phi^t_{(j,\ell)} \left(\mathbf{x}^t_{(j,\ell)} + \gamma^t \Delta \mathbf{x}^t_{(j,\ell)}\right)\Big) \\
&\leq \sum_{i=1}^{N}\sum_{j=1}^{N} a^t_{ij\ell}\phi^t_{(j,\ell)} \sum_{j=1}^{N} \tfrac{1}{\phi^{t+1}_{(i,\ell)}} a^t_{ij\ell}\phi^t_{(j,\ell)}\, r_\ell\Big(\mathbf{x}^t_{(j,\ell)} + \gamma^t \Delta \mathbf{x}^t_{(j,\ell)}\Big) \\
&\leq \sum_{i=1}^{N}\sum_{j=1}^{N} a^t_{ij\ell}\phi^t_{(j,\ell)} \Big((1-\gamma^t) r_\ell(\mathbf{x}^t_{(j,\ell)}) + \gamma^t r_\ell(\widetilde{\mathbf{x}}^t_{(j,\ell)}) \Big)
\end{aligned}$$

where we also used the convexity of r_ℓ. Using the column stochasticity of $a^t_{ij\ell}$ the condition (4.33) follows, concluding the proof. □

Lemma 4.5.6. *Consider the sequences generated by* Block-SONATA *and the following quantity*

$$P^t = c_2 \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \Big\| \frac{1}{N} \sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) - \mathbf{y}^t_{(i,\ell)} \Big\| + c_1 (\gamma^t)^2,$$

with $c_1 \geq 0$ *and* $c_2 \geq 0$. *Then,* $\sum_{t=0}^{\infty} P^t < \infty$.

Proof. We first notice that the second term is summable by Assumption 5.2.3. Focusing on the first term, we can write

$$\begin{aligned}
&\gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \Big\| \frac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) - \mathbf{y}^t_{(i,\ell)} \Big\| \\
&\leq \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \big\| \tfrac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\mathbf{x}^t_{(j,:)}) - \mathbf{y}^t_{(i,\ell)} \big\| \\
&+ \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \big\| \tfrac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\mathbf{x}^t_{(j,:)}) - \tfrac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) \big\|
\end{aligned}$$

where we added and subtracted $\frac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\mathbf{x}^t_{(j,:)})$ inside the norm and applied the triangle inequality.

For the first term we can retrace the reasoning in Proposition 4.5.4 and exploit (4.23) and (4.31) to conclude the summability of the first term. Using the Lipschitz continuity of ∇f_i (cf. Assumption 4.2.1) and Proposition 4.5.3 (cf. eq. (4.25)) the second term is summable. So that the proof follows. □

Proposition 4.5.7. *Consider the sequences generated by* Block-SONATA. *Then* $\{V^t\}_{t\geq 0}$ *satisfies*

$$V^{t+1} \leq V^t - c_3 \sum_{\ell=1}^{B}\sum_{i=1}^{N} \gamma^t \|\Delta \mathbf{x}^t_{(i,\ell)}\|^2 + P^t \tag{4.34}$$

for some positive, finite scalar c_3 *and with* P^t *defined as*

$$P^t = c_2\gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \Big\| \tfrac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) - \mathbf{y}^t_{(i,\ell)} \Big\| + c_1(\gamma^t)^2, \tag{4.35}$$

with c_1 *and* c_2 *being positive, finite scalars.*

Proof. Since $\sum_{i=1}^{N} f_i$ has Lipschitz continuous gradient with constant $L = \sum_{i=1}^{N} L_i$, the descent lemma ([9, Lemma A.24]) gives

$$\begin{aligned}
\sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^{t+1}) &\leq \sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^t) + \Big(\sum_{j=1}^{N} \nabla f_j(\bar{\boldsymbol{s}}^t)\Big)^\top \Big(\bar{\boldsymbol{s}}^{t+1} - \bar{\boldsymbol{s}}^t\Big) + \tfrac{L}{2}\|\bar{\boldsymbol{s}}^{t+1} - \bar{\boldsymbol{s}}^t\|^2 \\
&\leq \sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^t) + \sum_{\ell=1}^{B} \Big(\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t)\Big)^\top \frac{\gamma^t}{N} \sum_{i=1}^{N} \phi^t_{(i,\ell)} \Delta \mathbf{x}^t_{(i,\ell)} \\
&+ \frac{L}{2} \sum_{\ell=1}^{B} \Big\| \tfrac{1}{N}\gamma^t \sum_{i=1}^{N} \phi^t_{(i,\ell)} \Delta \mathbf{x}^t_{(i,\ell)} \Big\|^2.
\end{aligned}$$

At this point we add and subtract $\gamma^t \sum_{i=1}^{N}\sum_{\ell=1}^{B} \phi^t_{(i,\ell)} (\mathbf{y}^t_{(i,\ell)})^\top \Delta \mathbf{x}^t_{(i,\ell)}$ and

rearrange terms to obtain

$$\sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^{t+1}) \leq \sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^{t}) + \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \phi^t_{(i,\ell)} \big(\mathbf{y}^t_{(i,\ell)}\big)^\top \Delta \mathbf{x}^t_{(i,\ell)} \tag{4.36}$$

$$+ \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \phi^t_{(i,\ell)} \Big(\tfrac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) - \mathbf{y}^t_{(i,\ell)}\Big)^\top \Delta \mathbf{x}^t_{(i,\ell)} \tag{4.37}$$

$$+ (\gamma^t)^2 \frac{L}{2} \sum_{\ell=1}^{B}\sum_{i=1}^{N} \tfrac{\phi^t_{(i,\ell)}}{N} \|\Delta \mathbf{x}^t_{(i,\ell)}\|^2, \tag{4.38}$$

where we used the fact that for all $\ell \in \{1,\dots,B\}$ the coefficients $\phi^t_{(i,\ell)}/N$, $i \in \{1,\dots,N\}$ sum up to 1. We notice that (4.36) represents a gradient-related-type condition, thus Lemma 4.5.10 (ii) applies. Moreover, eq. (4.37) contains a term involving the gradient tracking error, so we can exploit Proposition 4.5.4. Finally, for (4.37) and (4.38) we can use the uniform boundedness of each $\Delta \mathbf{x}^t_{(i,\ell)} = \widetilde{\mathbf{x}}^t_{(i,\ell)} - \mathbf{x}^t_{(i,\ell)}$ given in Lemma 4.5.2 (ii). Thus, we can write

$$\begin{aligned}\sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^{t+1}) \leq \sum_{i=1}^{N} f_i(\bar{\boldsymbol{s}}^{t}) &- \gamma^t \tau \sum_{\ell=1}^{B}\sum_{i=1}^{N} \phi^t_{(i,\ell)} \|\Delta \mathbf{x}^t_{(i,\ell)}\|^2 \\ &- \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \phi^t_{(i,\ell)} \big(r_\ell(\widetilde{\mathbf{x}}^t_{(i,\ell)}) - r_\ell(\mathbf{x}^t_{(i,\ell)})\big) \\ &+ \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \phi^t_{(i,\ell)} \Big\| \tfrac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) - \mathbf{y}^t_{(i,\ell)} \Big\| \|\Delta \mathbf{x}^t_{(i,\ell)}\| \\ &+ c_1 (\gamma^t)^2.\end{aligned}$$

Using Proposition 4.5.5, we can conclude that

$$\begin{aligned}V^{t+1} \leq V^t &- c_3 \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \|\Delta \mathbf{x}^t_{(i,\ell)}\|^2 \\ &+ c_2 \gamma^t \sum_{\ell=1}^{B}\sum_{i=1}^{N} \Big\| \frac{1}{N}\sum_{j=1}^{N} \nabla_\ell f_j(\bar{\boldsymbol{s}}^t) - \mathbf{y}^t_{(i,\ell)} \Big\| + c_1 (\gamma^t)^2,\end{aligned}$$

where we used the lower bound on $\phi^t_{(i,\ell)}$, cf. [60, Corollary 2 (c)]. Using the definition of P^t in (4.35), the proof is complete. □

4.5.4 Best-Response Map

In order to prove the convergence to a stationary point of problem (4.1) we will rely on the properties of the following *best-response map* (see Section 4.5.6)

$$\widehat{\mathbf{x}}_{(i,\ell)}\big(\mathbf{x}^t_{(i,:)}\big) \triangleq \operatorname*{argmin}_{\mathbf{x}_\ell \in \mathcal{K}_\ell} \widehat{f}_{i,\ell}\Big(\mathbf{x}_\ell; \mathbf{x}^t_{(i,:)}, \tfrac{1}{N}\sum_{i=1}^{N} \nabla_\ell f_i(\mathbf{x}^t_{(i,:)})\Big) + r_\ell(\mathbf{x}_\ell), \tag{4.39}$$

that are reported in the following lemmas.

Lemma 4.5.8 ([33]). *The map $\widehat{\mathbf{x}}_{(i,\ell)}(\bullet)$ is Lipschitz continuous with constant $\widehat{L}$ for all $i \in \{1,\dots,N\}$ and $\ell \in \{1,\dots,B\}$. Moreover, for all $i \in \{1,\dots,N\}$, let $\widehat{\mathbf{x}}_{(i,:)}(\bullet)$ be the stack of $\widehat{\mathbf{x}}_{(i,\ell)}(\bullet)$ for all $\ell \in \{1,\dots,B\}$. Then the set of fixed points of $\widehat{\mathbf{x}}_{(i,:)}(\bullet)$ coincides with the stationary solutions of problem* (4.1). □

Lemma 4.5.9. *Consider the sequences generated by* Block-SONATA. *The following property holds true*

$$\lim_{t\to\infty} \big\|\widehat{\mathbf{x}}_{(i,\ell_i^t)}\big(\mathbf{x}^t_{(i,:)}\big) - \widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}\big\| = 0 \tag{4.40}$$

for all $i \in \{1,\dots,N\}$.

Proof. We use the shorthand $\widehat{\mathbf{x}}^t_{(i,\ell_i^t)}$ for $\widehat{\mathbf{x}}_{(i,\ell_i^t)}\big(\mathbf{x}^t_{(i,:)}\big)$. By writing the optimality conditions of $\widehat{\mathbf{x}}^t_{(i,\ell_i^t)}$ and $\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}$, we obtain that for all $\mathbf{w} \in \mathcal{K}_{\ell_i^t}$ the following conditions hold

$$\big(\mathbf{w} - \widehat{\mathbf{x}}^t_{(i,\ell_i^t)}\big)^\top \Big(\nabla_{\ell_i^t}\widehat{f}_{i,\ell_i^t}(\widehat{\mathbf{x}}^t_{(i,\ell_i^t)}; \mathbf{x}^t_{(i,:)}, \tfrac{1}{N}\sum_{i=1}^N \nabla_{\ell_i^t} f_i(\mathbf{x}^t_{(i,:)})) + \widetilde{\nabla} r_{\ell_i^t}\big(\widehat{\mathbf{x}}^t_{(i,\ell_i^t)}\big)\Big) \geq 0 \tag{4.41}$$

and

$$\big(\mathbf{w} - \widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}\big)^\top \Big(\nabla_{\ell_i^t}\widehat{f}_{i,\ell_i^t}(\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}; \mathbf{x}^t_{(i,:)}, \mathbf{y}^t_{(i,\ell_i^t)}) + \widetilde{\nabla} r_{\ell_i^t}\big(\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}\big)\Big) \geq 0 \tag{4.42}$$

Combining the two inequalities (4.42) and (4.41), we obtain

$$\begin{aligned}\Big(\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)} - \widehat{\mathbf{x}}^t_{(i,\ell_i^t)}\Big)^\top \Big(&\nabla_{\ell_i^t}\widehat{f}_{i,\ell_i^t}(\widehat{\mathbf{x}}^t_{(i,\ell_i^t)}; \mathbf{x}^t_{(i,:)}, \tfrac{1}{N}\sum_{i=1}^N \nabla_{\ell_i^t} f_i(\mathbf{x}^t_{(i,:)})) + \widetilde{\nabla} r_{\ell_i^t}\big(\widehat{\mathbf{x}}^t_{(i,\ell_i^t)}\big)\\ &- \nabla_{\ell_i^t}\widehat{f}_{i,\ell_i^t}(\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}; \mathbf{x}^t_{(i,:)}, \mathbf{y}^t_{(i,\ell_i^t)}) - \widetilde{\nabla} r_{\ell_i^t}\big(\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)}\big)\Big) \geq 0.\end{aligned}$$

By using the expression of $\widehat{f}_{i,\ell_i^t}$ as the sum of $\widetilde{f}_{i,\ell}$ and a first-order term, using the strong convexity of $\widetilde{f}_{i,\ell_i^t}$ (cf. Assumption 4.4.7), the convexity $r_{\ell_i^t}$ (cf. Assumption 4.2.1) and the Cauchy-Schwarz inequality, the following holds

$$\big\|\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)} - \widehat{\mathbf{x}}^t_{(i,\ell_i^t)}\big\| \leq \frac{1}{\tau_i}\Big\|\tfrac{1}{N}\sum_{i=1}^N \nabla_{\ell_i^t} f_i(\mathbf{x}^t_{(i,:)}) - \mathbf{y}^t_{(i,\ell_i^t)}\Big\|.$$

Finally, by noticing that

$$\begin{aligned}\limsup_{t\to\infty} \|\tfrac{1}{N}\sum_{i=1}^N \nabla_{\ell_i^t} f_i(\mathbf{x}^t_{(i,:)}) - \mathbf{y}^t_{(i,\ell_i^t)}\| &\leq \limsup_{t\to\infty} \sum_{\ell=1}^B \|\tfrac{1}{N}\sum_{i=1}^N \nabla_\ell f_i(\mathbf{x}^t_{(i,:)}) - \mathbf{y}^t_{(i,\ell)}\|\\ &\leq \sum_{\ell=1}^B \limsup_{t\to\infty} \|\tfrac{1}{N}\sum_{i=1}^N \nabla_\ell f_i(\mathbf{x}^t_{(i,:)}) - \mathbf{y}^t_{(i,\ell)}\|\end{aligned}$$

and invoking Proposition 4.5.4, we have $\limsup_{t\to\infty} \big\|\widetilde{\mathbf{x}}^t_{(i,\ell_i^t)} - \widehat{\mathbf{x}}_{(i,\ell_i^t)}\big\| = 0$, which implies (4.40) and the proof follows. □

4.5.5 Limit Points of Decision Estimates Average are Stationary

In this subsection we prove that every limit point of $\{\bar{\boldsymbol{s}}^t\}$ is a stationary point of problem (4.1).

As first step we notice that, since U is coercive and P^t, defined in (4.35), is summable by Lemma 4.5.6, then we have that $\{V^t\}_{t\geq 0}$ converges to a finite value and

$$\sum_{t=0}^{\infty}\sum_{i=1}^{N}\sum_{\tau=0}^{T-1}\gamma^{t+\tau}\|\Delta\mathbf{x}_{(i,\ell_i^{t+\tau})}^{t+\tau}\|^2<\infty. \tag{4.43}$$

For all $t\geq 0$, let us consider a (finite) time window of T consecutive steps, $[t,t+T-1]$. By Assumption 4.4.2 we have that all the blocks have been updated at least once by each agent $i\in\{1,\dots,N\}$ within this window. Since, in general $T\geq B$, then some blocks might have been updated more than once, i.e., there exist $0\leq s_1<s_2\leq T-1$ such that $\ell_i^{t+s_1}=\ell_i^{t+s_2}$. Thus, we split the summation in two terms: the first one contains only a single term for each block $\ell\in\{1,\dots,B\}$, while the second one contains all the remaining terms. Formally, for all $i\in\{1,\dots,N\}$, it holds

$$\sum_{i=1}^{N}\sum_{s=0}^{T-1}\|\Delta\mathbf{x}_{(i,\ell_i^{t+s})}^{t+s}\|=\sum_{i=1}^{N}\sum_{\ell=1}^{B}\|\Delta\mathbf{x}_{(i,\ell_i^{t+s_i^t(\ell)})}^{t+s_i^t(\ell)}\|+\sum_{i=1}^{N}\sum_{\substack{s=0\\ s\neq s_i^t(\ell),\ell\in\{1,\dots,B\}}}^{T-1}\|\Delta\mathbf{x}_{(i,\ell_i^{t+s})}^{t+s}\|$$

where $t+s_i^t(\ell)$ is defined as the last time node i has updated block ℓ in $[t,t+T-1]$. Notice that $\ell_i^{t+s_i^t(\ell)}=\ell$.

Let us introduce the following quantity

$$\boldsymbol{\Delta}^t\triangleq\sum_{i=1}^{N}\sum_{\ell=1}^{B}\|\Delta\mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)}\|. \tag{4.44}$$

It will play a key role to prove (subsequence) convergence of $\{\bar{\boldsymbol{s}}^t\}_{t\geq 0}$.

Since γ^t is not summable (Cf. Assumption 5.2.3), by (4.43) it follows that $\liminf_{t\to\infty}\boldsymbol{\Delta}^t=0$. Next, we prove that also the lim sup is zero. Assume by contradiction that $\limsup_{t\to\infty}\boldsymbol{\Delta}^t>0$. Then, there exists a $\delta>0$ such that $\boldsymbol{\Delta}^t<\delta$ for infinitely many t and also $\boldsymbol{\Delta}^t>2\delta$ for infinitely many t. Therefore, we can always find an infinite set of indexes, say $\mathcal{T}$, having the following properties: for any $t\in\mathcal{T}$, there exists an integer $\theta_t>t$ such that

$$\begin{aligned}&\boldsymbol{\Delta}^t<\delta,\ \boldsymbol{\Delta}^{\theta_t}>2\delta\\ &\delta\leq\boldsymbol{\Delta}^{\tau}\leq 2\delta,\qquad t<\tau<\theta_t\end{aligned}$$

Therefore, for all $t \in \mathcal{T}$, we have

$$\begin{aligned}\delta &< \boldsymbol{\Delta}^{\theta_t} - \boldsymbol{\Delta}^t \\ &= \sum_{i=1}^{N}\sum_{\ell=1}^{B}\Big(\|\widetilde{\mathbf{x}}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)} - \mathbf{x}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)}\| - \|\widetilde{\mathbf{x}}_{(i,\ell)}^{t+s_i^t(\ell)} - \mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)}\|\Big) \\ &\leq \sum_{i=1}^{N}\sum_{\ell=1}^{B}\Big(\|\widetilde{\mathbf{x}}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)} - \widetilde{\mathbf{x}}_{(i,\ell)}^{t+s_i^t(\ell)}\| + \|\mathbf{x}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)} - \mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)}\|\Big)\end{aligned} \tag{4.45}$$

where we used the (reverse and standard) triangle inequality to separate optimization-related from consensus-related quantities. Then, by adding and subtracting suitable terms involving the best response map $\widehat{\mathbf{x}}_{(i,\ell)}(\bullet)$ at different iterations, we have

$$\begin{aligned}\delta &< \sum_{i=1}^{N}\sum_{\ell=1}^{B}\Big(\|\widetilde{\mathbf{x}}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)} - \widehat{\mathbf{x}}_{(i,\ell)}(\mathbf{x}_{(i,:)}^{\theta_t+s_i^{\theta_t}(\ell)})\| \\ &\qquad + \|\widehat{\mathbf{x}}_{(i,\ell)}(\mathbf{x}_{(i,:)}^{\theta_t+s_i^{\theta_t}(\ell)}) - \widehat{\mathbf{x}}_{(i,\ell)}(\mathbf{x}_{(i,:)}^{t+s_i^t(\ell)})\| \\ &\qquad + \|\widehat{\mathbf{x}}_{(i,\ell)}(\mathbf{x}_{(i,:)}^{t+s_i^t(\ell)}) - \widetilde{\mathbf{x}}_{(i,\ell)}^{t+s_i^t(\ell)}\| + \|\mathbf{x}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)} - \mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)}\|\Big) \\ &\leq (1+\widehat{L})\sum_{i=1}^{N}\sum_{\ell=1}^{B}\|\mathbf{x}_{(i,:)}^{\theta_t+s_i^{\theta_t}(\ell)} - \mathbf{x}_{(i,:)}^{t+s_i^t(\ell)}\| + e_1^t\end{aligned} \tag{4.46}$$

where we used the triangle inequality, the Lipschitz continuity of $\widehat{\mathbf{x}}_{(i,\ell)}(\bullet)$ (Cf. Proposition 4.5.11) and we set

$$\begin{aligned}e_1^t \triangleq \sum_{i=1}^{N}\sum_{\ell=1}^{B}\Big(&\|\widehat{\mathbf{x}}_{(i,\ell)}(\mathbf{x}_{(i,:)}^{\theta_t+s_i^{\theta_t}(\ell)}) - \widetilde{\mathbf{x}}_{(i,\ell)}^{\theta_t+s_i^{\theta_t}(\ell)}\| \\ &+\|\widehat{\mathbf{x}}_{(i,\ell)}(\mathbf{x}_{(i,:)}^{t+s_i^t(\ell)}) - \widetilde{\mathbf{x}}_{(i,\ell)}^{t+s_i^t(\ell)}\|\Big).\end{aligned} \tag{4.47}$$

Let us add other suitable terms involving the $\bar{\boldsymbol{s}}$: applying the triangle inequality, we obtain

$$\delta < (1+\widehat{L})\sum_{i=1}^{N}\sum_{\ell=1}^{B}\|\bar{\boldsymbol{s}}^{\theta_t+s_i^{\theta_t}(\ell)} - \bar{\boldsymbol{s}}^{t+s_i^t(\ell)}\| + e_1^t + e_2^t,$$

where we set

$$e_2^t \triangleq \sum_{i=1}^{N}\sum_{\ell=1}^{B}\Big(\|\mathbf{x}_{(i,:)}^{\theta_t+s_i^{\theta_t}(\ell)} - \bar{\boldsymbol{s}}^{\theta_t+s_i^{\theta_t}(\ell)}\| + \|\bar{\boldsymbol{s}}^{t+s_i^t(\ell)} - \mathbf{x}_{(i,:)}^{t+s_i^t(\ell)}\|\Big). \tag{4.48}$$

Since $\theta_t + s_i^{\theta_t}(\ell)$ is the *last* time at which block ℓ has been updated by agent i in $[\theta_t, \theta_t + T]$, it must hold that $\theta_t + s_i^{\theta_t}(\ell) \geq t + s_i^t(\ell)$ for all $t \geq 0$. Without loss of generality, we consider the worst case in which the strict inequality $\theta_t + s_i^{\theta_t}(\ell) > t + s_i^t(\ell)$ holds for all $i \in \{1, \ldots, N\}$ and $\ell \in \{1, \ldots, B\}$. In this way the interval $[t + s_i^t(\ell), \theta_t + s_i^{\theta_t}(\ell)]$ is nonempty.

Recalling the linear dynamics of $\bar{\boldsymbol{s}}_\ell$ given by (4.19), we can upper bound $\|\bar{\boldsymbol{s}}_\ell^{\theta_t+s_i^{\theta_t}(\ell)} - \bar{\boldsymbol{s}}_\ell^{t+s_i^t(\ell),}\|$ by the norm of the forced response of the system, obtaining

$$\delta < c_2 \sum_{i=1}^{N} \sum_{\ell=1}^{B} \sum_{\tau=t+s_i^t(\ell)}^{\theta_t+s_i^{\theta_t}(\ell)-1} \sum_{h=1}^{N} \gamma^\tau \|\Delta \mathbf{x}^\tau_{(h,\ell_h^\tau)}\| + e_1^t + e_2^t.$$

By adding a finite number of terms in the summation indexed by τ, by rearranging terms and changing the index letter h into i, we can write

$$\delta < c_3 \sum_{i=1}^{N} \sum_{\tau=t+1}^{\theta_t+T-1} \gamma^\tau \|\Delta \mathbf{x}^\tau_{(i,\ell_i^\tau)}\| + e_1^t + e_2^t.$$

For all $i \in \{1, \dots, N\}$, by definition of $\boldsymbol{\Delta}^t$ in (4.44) it holds

$$\Delta \mathbf{x}^{t+T-1}_{(i,\ell_i^{t+T-1})} = \Delta \mathbf{x}^{t+s_i^t(\ell_i^{t+T-1})}_{(i,\ell_i^{t+T-1})} \leq \boldsymbol{\Delta}^t,$$

for all $t \geq 0$. Thus, it holds also for $t - T$, so that we can write

$$\delta < c_4 \sum_{\tau=t+1}^{\theta_t+T-1} \gamma^\tau \boldsymbol{\Delta}^{\tau-T+1} + e_1^t + e_2^t \leq c_4 \sum_{\tau=t+1}^{\theta_t} \gamma^{\tau+T-1} \boldsymbol{\Delta}^\tau + e_1^t + e_2^t + e_3^t,$$

where we shifted the index and set

$$e_3^t \triangleq \sum_{\tau=t-T+2}^{t} \gamma^{\tau+T-1} \boldsymbol{\Delta}^\tau. \tag{4.49}$$

Since by hypothesis $\boldsymbol{\Delta}^\tau \geq \delta$ for $\tau \in [t, \theta_t - 1]$, then we can invoke Lemma 4.5.12 and introduce the squared norm, obtaining

$$\delta < c_5 \sum_{\tau=t+1}^{\theta_t} \gamma^{\tau+T-1} \sum_{i=1}^{N} \sum_{\ell=1}^{B} \|\Delta \mathbf{x}^{\tau+s_i^\tau(\ell)}_{(i,\ell)}\|^2 + e_1^t + e_2^t + e_3^t.$$

Finally, by adding terms we can write

$$\begin{aligned} \delta &< c_5 \sum_{\tau=t+1}^{\theta_t} \gamma^{\tau+T-1} \sum_{i=1}^{N} \sum_{s=0}^{T-1} \|\Delta \mathbf{x}^{\tau+s}_{(i,\ell_i^{\tau+s})}\|^2 + e_1^t + e_2^t + e_3^t \\ &\overset{(a)}{\leq} c_5 \sum_{\tau=t+1}^{\theta_t} \sum_{i=1}^{N} \sum_{s=0}^{T-1} \gamma^{\tau+s} \|\Delta \mathbf{x}^{\tau+s}_{(i,\ell_i^{\tau+s})}\|^2 + e_1^t + e_2^t + e_3^t. \end{aligned} \tag{4.50}$$

where in (a) we used the non-increasing property of the step-size γ^t (cf. Assumption 5.2.3).

We notice that e_1^t, e_2^t and e_3^t (defined in (4.47), (4.48), (4.49) respectively) are vanishing because of Proposition 4.5.3, Lemma 4.5.9 and since (4.43) holds, then there exists a sufficiently large $\bar{t}$ such that the "tail" (indexed

by $\tau \in [t, \theta_t - 1]$) is arbitrary small for all $t \geq \bar{t}$, due to the convergence of the series in (4.43); then it must hold

$$c_5 \sum_{\tau=t+1}^{\theta_t-1} \sum_{i=1}^{N} \sum_{s=0}^{T-1} \gamma^{\tau+s} \|\Delta \mathbf{x}_{(i,\ell_i^{\tau+s})}^{\tau+s}\|^2 + e_1^t + e_2^t + e_3^t < \delta$$

for all $t \geq \bar{t}$, giving a contradiction with (4.50). Thus, it must hold that $\limsup_{t\to\infty} \boldsymbol{\Delta}^t = 0$, and hence

$$\lim_{t\to\infty} \sum_{i=1}^{N} \sum_{\ell=1}^{B} \|\Delta \mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)}\| = 0. \tag{4.51}$$

As a final step of the proof, we show that any limit point of $\{\bar{\boldsymbol{s}}^t\} t \geq 0$. Since U is coercive and V^t converges, then the sequence $\{\bar{\boldsymbol{s}}^t\}_{t\geq 0}$ is bounded and admits a limit point $\bar{\boldsymbol{s}}^\infty$ in $\mathcal{K}$.

By Lemma 4.5.8, $\bar{\mathbf{s}}^\infty$ is a stationary solution of problem (4.1), if

$$\lim_{t\to\infty} \left\| \widehat{\mathbf{x}}_{(i,:)}\big(\bar{\boldsymbol{s}}^t\big) - \bar{\boldsymbol{s}}^t \right\| = 0, \tag{4.52}$$

for all $i \in \{1, \ldots, N\}$. To prove (4.52), we first bound $\|\widehat{\mathbf{x}}_{(i,:)}\big(\bar{\boldsymbol{s}}^t\big) - \bar{\boldsymbol{s}}^t\|$ as follows

$$\begin{aligned}
\|\widehat{\mathbf{x}}_{(i,:)}\big(\bar{\boldsymbol{s}}^t\big) - \bar{\boldsymbol{s}}^t\| &= \sum_{\ell=1}^{B} \|\widehat{\mathbf{x}}_{(i,\ell)}\big(\bar{\boldsymbol{s}}^t\big) - \bar{\boldsymbol{s}}_\ell^t\| \\
&\overset{(a)}{\leq} \sum_{\ell=1}^{B} \Big(\|\widehat{\mathbf{x}}_{(i,\ell)}\big(\bar{\boldsymbol{s}}^t\big) - \widehat{\mathbf{x}}_{(i,\ell)}\big(\bar{\boldsymbol{s}}^{t+s_i^t(\ell)}\big)\| \\
&\quad + \|\widehat{\mathbf{x}}_{(i,\ell)}\big(\bar{\boldsymbol{s}}^{t+s_i^t(\ell)}\big) - \bar{\boldsymbol{s}}_\ell^{t+s_i^t(\ell)}\| + \|\bar{\boldsymbol{s}}_\ell^{t+s_i^t(\ell)} - \bar{\boldsymbol{s}}_\ell^t\| \Big) \\
&\overset{(b)}{\leq} \sum_{\ell=1}^{B} \Big(\|\widehat{\mathbf{x}}_{(i,\ell)}\big(\bar{\boldsymbol{s}}^{t+s_i^t(\ell)}\big) - \bar{\boldsymbol{s}}_\ell^{t+s_i^t(\ell)}\| + (1+\hat{L})\|\bar{\boldsymbol{s}}^{t+s_i^t(\ell)} - \bar{\boldsymbol{s}}^t\| \Big) \\
&\overset{(c)}{\leq} \sum_{\ell=1}^{B} \Big(\|\widehat{\mathbf{x}}_{(i,\ell)}\big(\mathbf{x}_{(i,:)}^{t+s_i^t(\ell)}\big) - \widetilde{\mathbf{x}}_{(i,\ell)}^{t+s_i^t(\ell)}\| \\
&\quad + \|\widehat{\mathbf{x}}_{(i,\ell)}\big(\bar{\boldsymbol{s}}^{t+s_i^t(\ell)}\big) - \widehat{\mathbf{x}}_{(i,\ell)}\big(\mathbf{x}_{(i,:)}^{t+s_i^t(\ell)}\big)\| + \|\Delta \mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)}\| \\
&\quad + \|\mathbf{x}_{(i,\ell)}^{t+s_i^t(\ell)} - \bar{\boldsymbol{s}}_\ell^{t+s_i^t(\ell)}\| + (1+\hat{L})\|\bar{\boldsymbol{s}}^{t+s_i^t(\ell)} - \bar{\boldsymbol{s}}^t\| \Big)
\end{aligned}$$

where in (a) and in (c) we added terms and used the triangle inequality while in (b) we used the Lipschitz continuity of the map $\widehat{\mathbf{x}}_{(i,\ell)}(\bullet)$.

Using the consensus agreement Proposition 4.5.3 (Cf. (4.24)), the best response consistency Lemma 4.5.9, the condition (4.51), we get (4.52) for all $i \in \{1, \ldots, N\}$. So that the proof follows.

4.5.6 Technicalities

Consider the following optimization problem

$$\widetilde{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{K}} \; h(\mathbf{x}) + r(\mathbf{x}) \tag{4.53}$$

where h has L-Lipschitz continuous gradient, r is convex (possibly nonsmooth) and $\mathcal{K}$ is a closed, convex set.

Lemma 4.5.10. *If, in addition, h is τ-strongly convex, then, for all $\mathbf{v} \in \mathcal{K}$, the following holds*

1. $\|\widetilde{\mathbf{x}} - \mathbf{v}\| \leq \frac{1}{\tau}\|\nabla h(\mathbf{v})\| + \frac{1}{\tau}\|\widetilde{\nabla} r(\widetilde{\mathbf{x}})\|$;
2. $\nabla h(\mathbf{v})^\top(\widetilde{\mathbf{x}} - \mathbf{v}) \leq -\tau\|\widetilde{\mathbf{x}} - \mathbf{v}\|^2 - (r(\widetilde{\mathbf{x}}) - r(\mathbf{v}))$.

Proof. The optimality conditions gives

$$\big(\nabla h(\widetilde{\mathbf{x}}) + \widetilde{\nabla} r(\widetilde{\mathbf{x}})\big)^\top(\widetilde{\mathbf{x}} - \mathbf{v}) \geq 0.$$

for all $\mathbf{v} \in \mathcal{K}$. By adding and subtracting $\nabla h(\mathbf{v})$, it follows

$$\big(\nabla h(\widetilde{\mathbf{x}}) \pm \nabla h(\mathbf{v}) + \widetilde{\nabla} r(\widetilde{\mathbf{x}})\big)^\top(\widetilde{\mathbf{x}} - \mathbf{v}) \geq 0,$$

which, by using strong convexity of h, implies

$$\|\widetilde{\mathbf{x}} - \mathbf{v}\|^2 \leq -\frac{1}{\tau}\big(\nabla h(\mathbf{v}) + \widetilde{\nabla} r(\widetilde{\mathbf{x}})\big)^\top(\widetilde{\mathbf{x}} - \mathbf{v}).$$

Using the Cauchy-Schwarz inequality, the proof of (i) follows.

Since r is convex, we have $\widetilde{\nabla} r(\widetilde{\mathbf{x}})^\top(\mathbf{v} - \widetilde{\mathbf{x}}) \leq r(\mathbf{v}) - r(\widetilde{\mathbf{x}})$. Thus, we can write

$$\tau\|\widetilde{\mathbf{x}} - \mathbf{v}\|^2 \leq \big(\nabla h(\mathbf{v}) + \widetilde{\nabla} r(\widetilde{\mathbf{x}})\big)^\top(\mathbf{v} - \widetilde{\mathbf{x}}) \leq -\nabla h(\mathbf{v})^\top(\widetilde{\mathbf{x}} - \mathbf{v}) + r(\mathbf{v}) - r(\widetilde{\mathbf{x}})$$

which can be rearranged as $\nabla h(\mathbf{v})^\top(\widetilde{\mathbf{x}} - \mathbf{v}) \leq -\tau\|\widetilde{\mathbf{x}} - \mathbf{v}\|^2 + r(\mathbf{v}) - r(\widetilde{\mathbf{x}})$, concluding the proof. □

The latter equation represents a generalization of the gradient-related condition for composite minimization.

Proposition 4.5.11. *Given a strongly convex approximation $\widehat{h}(\mathbf{x}; \mathbf{w}, \nabla h(\mathbf{w}))$ of h (with parameter τ) about $\mathbf{w}$, where recall that $\nabla\widehat{h}(\mathbf{w}; \mathbf{w}, \nabla h(\mathbf{w})) = \nabla h(\mathbf{w})$, then the best response map $\widehat{\mathbf{x}}(\mathbf{w})$ defined as*

$$\widehat{\mathbf{x}}(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{K}} \; \widehat{h}(\mathbf{x}; \mathbf{w}, \nabla h(\mathbf{w})) + r(\mathbf{x}) \tag{4.54}$$

satisfies

(i) $\widehat{\mathbf{x}}(\bullet)$ *is Lipschitz continuous on* $\mathcal{K}$*, i.e., there exists a positive constant* $\widehat{L}$ *such that*

$$\|\widehat{\mathbf{x}}(\mathbf{w}_1) - \widehat{\mathbf{x}}(\mathbf{w}_2)\| \leq \widehat{L}\|\mathbf{w}_1 - \mathbf{w}_2\| \tag{4.55}$$

for all $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{K}$*.*

(ii) The set of fixed points of $\widehat{\mathbf{x}}(\bullet)$ *coincides with the set of stationary solutions of problem* (4.53)*; therefore* $\widehat{\mathbf{x}}(\bullet)$ *has a fixed point.* □

Lemma 4.5.12. *Consider a vector* $(w_1, \ldots, w_H) \in \mathbb{R}^H$ *with positive entries and such that it satisfies* $\sum_{h=1}^H w_h \geq \delta > 0$ *for some* $\delta > 0$*. Then it holds*

$$\sum_{h=1}^H w_h \leq \frac{H}{\delta} \sum_{h=1}^H w_h^2.$$

Proof. We can write $\delta \mathbf{1}^\top \mathbf{w} \leq (\mathbf{1}^\top \mathbf{w})^2 \leq H^2 \Big(\frac{1}{H}\mathbf{1}^\top \mathbf{w}\Big)^2 \leq H \sum_{h=1}^H w_h^2$, where we used the convexity of the square norm. □

4.6 Application to Nonconvex Sparse Regression

In this section we apply Block-SONATA to the distributed sparse regression problem described in Section 1.5. Consider a network of N agents taking linear measurements of a sparse signal $\mathbf{x}_0 \in \mathbb{R}^m$, with data matrix $\mathbf{D}_i \in \mathbb{R}^{n_i \times m}$. The observation taken by agent i can be expressed as $\mathbf{b}_i = \mathbf{D}_i \mathbf{x}_0 + \mathbf{n}_i$,where $\mathbf{n}_i \in \mathbb{R}^{n_i}$ accounts for the measurement noise. To estimate the underlying signal $\mathbf{x}_0$, we formulate the problem as:

$$\min_{\mathbf{x} \in \mathcal{K}} \sum_{i=1}^N \underbrace{\|\mathbf{D}_i \mathbf{x} - \mathbf{b}_i\|_2^2}_{f_i(\mathbf{x})} + r(\mathbf{x}) \tag{4.56}$$

where $\mathbf{x} \in \mathbb{R}^m$; $\mathcal{K}$ is the box constraint set $\mathcal{K} \triangleq [k_L, k_U]^m$, with scalars $k_L \leq k_U$; and $r : \mathbb{R}^m \to \mathbb{R}$ is a difference-of-convex (DC) sparsity-promoting regularizer, given by

$$r(\mathbf{x}) \triangleq \lambda \cdot \sum_{j=1}^m r_0(x_j), \quad r_0(x_j) \triangleq \frac{\log(1 + \theta|x_j|)}{\log(1+\theta)}.$$

The first step for the application of Block-SONATA is to build a valid surrogate $\widetilde{f}_{i,\ell}$ of f_i (cf. Assumption 4.4.7). To this end, we first rewrite r_0 as a difference-of-convex function. It is not difficult to check that

$$r_0(x) = \underbrace{\eta(\theta)|x|}_{r_0^+(x)} - \underbrace{(\eta(\theta)|x| - r_0(x))}_{r_0^-(x)},$$

where $r_0^+ : \mathbb{R} \to \mathbb{R}$ is convex non-smooth with $\eta(\theta) \triangleq \frac{\theta}{\log(1+\theta)}$, and $r_0^- : \mathbb{R} \to \mathbb{R}$ is convex and has Lipschitz continuous first order derivative given by

$$\frac{dr_0^-}{dx}(x) = \text{sign}(x) \cdot \frac{\theta^2|x|}{\log(1+\theta)(1+\theta|x|)}.$$

Denoting the coordinates associated with block ℓ as $\mathcal{I}_\ell$, define matrix $\mathbf{D}_{i,\ell}$ [resp. $\mathbf{D}_{i,-\ell}$] constructed by picking the columns of $\mathbf{D}_i$ that belong [resp. do not belong] to $\mathcal{I}_\ell$. Then, the following functions are two alternative valid surrogate functions.

– *Partial linearization:* Since f_i is convex, a first natural choice to satisfy Assumption 4.4.7 is to keep f_i unaltered while linearizing the nonconvex part in r_0, which leads to the following surrogate

$$\begin{aligned}\tilde{f}_{i,\ell}^{PL}(\mathbf{x}_\ell; \mathbf{x}_{(i,:)}^t) = \|\mathbf{D}_{i,\ell}\mathbf{x}_\ell + \mathbf{D}_{i,-\ell}\mathbf{x}_{(i,-\ell)}^t - \mathbf{b}_i\|_2^2 \\ + \frac{\tau_i^{PL}}{2}\|\mathbf{x}_\ell - \mathbf{x}_{(i,\ell)}^t\|^2 - \sum_{k\in\mathcal{I}_\ell} \frac{dr_0^-((\mathbf{x}_{(i,\ell)}^t)_k)}{dx}(\mathbf{x}_\ell - \mathbf{x}_{(i,\ell)}^t)_k,\end{aligned} \tag{4.57}$$

where we set $v_{ik}^t \triangleq \frac{dr_0^-((\mathbf{x}_{(i,\ell)}^t)_k)}{dx}$, while x denotes a scalar variable and, e.g., $(\mathbf{x}_{(i,\ell)}^t)_k$ denotes the k-th scalar component of $\mathbf{x}_{(i,\ell)}^t$.

– *Linearization:* A second surrogate can be obtained linearizing also f_i, which leads to

$$\begin{aligned}\tilde{f}_{i,\ell}^{L}(\mathbf{x}_\ell; \mathbf{x}_{(i,:)}^t) = \Big(2\mathbf{D}_{i,\ell}^\top(\mathbf{D}_i - \mathbf{b}_i)\Big)^\top(\mathbf{x}_\ell - \mathbf{x}_{(i,\ell)}^t) \\ + \frac{\tau_i^{L}}{2}\|\mathbf{x}_\ell - \mathbf{x}_{(i,\ell)}^t\|^2 - \sum_{k\in\mathcal{I}_\ell} v_{ik}^t(\mathbf{x}_\ell - \mathbf{x}_{(i,\ell)}^t)_k.\end{aligned} \tag{4.58}$$

Notice that the minimizer of $\tilde{f}_{i,\ell}^L$ can be computed in closed form as

$$\mathbf{x}_{(i,\ell)}^{t+1} = \mathcal{P}_{\mathcal{K}_\ell}\Big(\mathcal{S}_{\frac{\lambda\eta}{\tau_i^L}}\Big(\mathbf{x}_{(i,\ell)}^t - \frac{1}{\tau_i^L}(2\mathbf{D}_{i,\ell}^\top(\mathbf{D}_i - \mathbf{b}_i) - \mathbf{v}_{i,\ell}^t)\Big)\Big)$$

where $\mathbf{v}_{i,\ell}^t \triangleq (v_{ik}^t)_{k\in\mathcal{I}_\ell}$, $\mathcal{S}_\lambda(\mathbf{x}) \triangleq \text{sign}(\mathbf{x}) \cdot \max\{|\mathbf{x}| - \lambda, 0\}$ (operations are performed element-wise), and $\mathcal{P}_{\mathcal{K}_\ell}$ is the Euclidean projection onto $\mathcal{K}_\ell$.

We term the two algorithm versions based on (4.57) and (4.58) Block-SONATA-PL and Block-SONATA-L, respectively, and we test them considering the following simulation set-up. The variable dimension m is set to be 2000, $\mathcal{K}$ is set to be $[-10, 10]^{2000}$, and the regularization parameters are set to $\lambda = 0.1$ and $\theta = 20$. The network is composed of $N = 50$ agents, communicating over a fixed undirected graph $\mathcal{G}$ generated using an Erdős-Rényi random having algebraic connectivity 6. The components of the ground-truth signal $\mathbf{x}_0$ are i.i.d., generated according to the Normal distribution $\mathcal{N}(0, 1)$. To impose sparsity on $\mathbf{x}_0$, we set the 80% smallest entries of $\mathbf{x}_0$

to zero. Each agent i has a measurement matrix $\mathbf{D}_i \in \mathbb{R}^{400\times 2000}$ with i.i.d. $\mathcal{N}(0,1)$ distributed entries (with ℓ_2-normalized rows), and the observation noise $\mathbf{n}_i$ has entries i.i.d. distributed according to $\mathcal{N}(0, 0.1)$.

We compare the two algorithms, Block-SONATA-PL and Block-SONATA-L, with the (sub)gradient-projection algorithm proposed in [14]. Note that there is no formal proof of convergence for such an algorithm in the nonconvex setting. We used the following tuning for the algorithms. The diminishing step-size is chosen as $\gamma^t = \gamma^{t-1}(1-\mu\gamma^{t-1})$, with $\gamma^0 = 0.5$ and $\mu = 10^{-5}$; the proximal parameter is $\tau_i^{PL} = 3.5$ and $\tau_i^{L} = 4.5$. To evaluate the algorithmic performance we use two merit functions. The first one measures the distance from stationarity of the weighted average of the agents' iterates $\bar{\boldsymbol{s}}^t$ and is given by

$$J^t \triangleq \left\| \bar{\boldsymbol{s}}^t - \mathcal{P}_{\mathcal{K}}\Big(\mathcal{S}_{\eta\lambda}\Big(\bar{\boldsymbol{s}}^t - \big(\sum_{i=1}^{N} \nabla f_i(\bar{\boldsymbol{s}}^t) - r(\bar{\boldsymbol{s}}^t)\big)\Big)\Big) \right\|_{\infty}.$$

Note that J^t is a valid merit function: it is continuous and it is zero if and only if its argument $\bar{\boldsymbol{s}}^t$ is a stationary point of problem (4.56). The second merit function quantifies the consensus disagreement at each iteration, and is defined as

$$D^t \triangleq \max_{i\in\{1,\dots,N\}} \left\| \mathbf{x}^t_{(i,:)} - \bar{\boldsymbol{s}}^t \right\|.$$

The performance of Block-SONATA-PL and Block-SONATA-L for different choices of the block dimension are reported in Figure 4.1 and Figure 4.2, respectively.

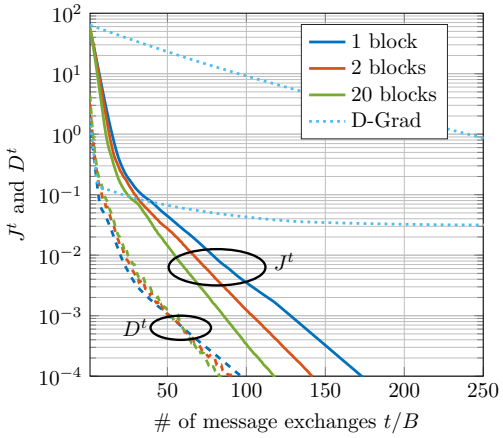


Figure 4.1: Optimality measurement J^t (solid) and consensus error D^t (dashed) versus the number of message exchange for several choices of the blocks' number B of Block-SONATA-PL.

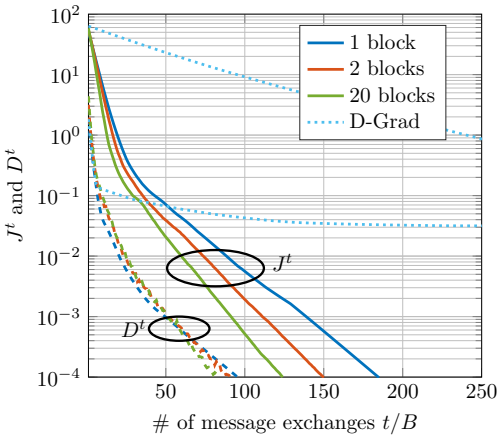


Figure 4.2: Optimality measurement J^t (solid) and consensus error D^t (dashed) versus the number of message exchange for several choices of the blocks' number B of Block-SONATA-L.

To fairly compare the algorithms run for different block sizes, we plot J^t and D^t versus the average agents' "message exchanges", defined as t/B, where t is the iteration counter used in the algorithm description. The figures show that both consensus and stationarity have been achieved by Block-SONATA-PL and Block-SONATA-L within 200 message exchanges while the plain gradient scheme [60] using all the blocks is much slower.

Finally, Figure 4.3 shows the number of message exchanges required to obtain $J^t < 10^{-4}$ versus the number of blocks B of the two algorithm instances. The figure clearly shows that the overall communication cost is reduced by increasing the number of blocks, validating the proposed block-wise optimization/communication strategy. Note also that Block-SONATA-PL outperforms Block-SONATA-L. This is due to the fact that the partial linearization scheme better preserves the (partial) convexity of the objective function.

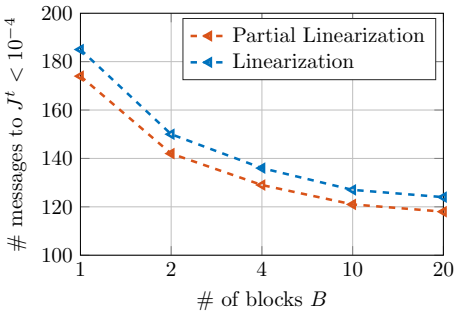


Figure 4.3: Number of message exchanges required to obtain $J^t < 10^{-4}$ versus the number of blocks B.

Chapter 5

Constraint-Coupled Distributed Optimization: a Relaxation and Duality Approach

In this chapter we consider a constraint-coupled distributed optimization scenario in which agents of a network want to minimize the sum of local convex cost functions, each one depending on a local variable, subject to convex local and coupling constraints, with the latter involving all the decision variables. We propose a novel distributed algorithm based on a relaxation of the primal problem and an elegant exploration of duality theory. Then, we corroborate the theoretical results by showing how the methodology applies to an instance of a Distributed Model Predictive Control scheme in a microgrid control scenario and to an instance of peak-minimization in Demand Side Management of smart grids. The results of this chapter are based on [70–73].

5.1 Literature Review

We organize the relevant literature to this chapter in two parts: (i) parallel and distributed methods for general constraint-coupled problems, and (ii) cooperative distributed MPC schemes. Parallel methods, based on a master-subproblem architecture, solving constraint-coupled optimization problems trace back to late 90s [12]. Duality is a widely-used tool for parallel and (classical) distributed optimization algorithms as shown, e.g., in the tutorial [80]. Parallel augmented Lagrangian methods are proposed in [26, 39], where a nonconvex version of the problem with linear coupling constraints is considered.

Purely distributed approaches with no central authority, as the one we propose in this chapter, have been also proposed for this set-up. In [20] a cutting-plane consensus scheme is proposed in which agents iteratively approximate their local problem with linear constraints (cutting-planes) and

exchange the active ones with their neighbors. This algorithm works under asynchronous, directed networks, but it badly scales with the network size since it requires the exchange of a number of constraints that depends on the number of agents. In [25] a distributed consensus-based primal-dual perturbation algorithm is proposed to solve a constraint-coupled optimization problem in which a coupling term known to all the agents can also be present. Differently from our set-up, they assume to have differentiable costs and constraints. Paper [24] proposes a distributed ADMM algorithm to solve problems in which the coupling constraint is linear and no local constraints are present. Our set-up allows for general convex coupling and local constraints. In [91] and [34], distributed algorithms are proposed relying on a consensus-based dual decomposition and a dual proximal minimization respectively. Primal recovery mechanisms are devised in [25, 34, 91] in order to recover a primal solution by suitably applying running average schemes borrowed from the centralized literature, see, e.g., [63] and references therein. Notice that the converge rate of running averages is, in general, slow. Our methodology does not need such a mechanism to obtain a feasible solution and appears to be much faster in the simulations we carried out. A class of min-max optimization problems, strictly related to the same problem set-up investigated in this chapter, are addressed in [53]. They solve the min-max problem through a Laplacian-based saddle-point subgradient scheme and use a totally different methodology from ours.

As for cooperative Model Predictive Control (MPC) schemes, in [95] the dual decomposition method is combined with a penalty approach to solve separable nonconvex optimization problems arising in MPC. A linear convergence rate for a dual gradient algorithm for linearly constrained separable convex problems is proven in [59]. In [96] an inexact dual decomposition scheme combined with smoothing techniques is proposed. In [38] a real-time strategy is proposed to solve parametric nonconvex programs usually arising in nonlinear MPC using a primal-dual approach. Differently from our framework, the above papers propose parallel algorithms where a centralized authority is needed to perform some algorithmic steps. In [36] an algorithm based on an accelerated dual decomposition is proposed to solve a distributed MPC problem. A sparse structure in the coupling dynamics is assumed and exploited to obtain a distributed implementation of the algorithm. Notice that in our case, we do not need such a special structure in the constraints, while we can handle general coupling constraints that might possibly involve all the agents in the network. In [19] a fully-distributed optimization algorithm is applied to solve MPC problems with coupled outputs. The same technique has been extended to the robust economic dispatch problem in [49]. This approach works under asynchronous and unreliable communication, but at each algorithmic iteration agents need to exchange multiple predicted trajectories to agree on the entire solution vector.

5.2 Relaxation and Successive Distributed Decomposition Algorithm

In this section we formally state the problem we aim at solving and introduce the proposed distributed algorithm along with its convergence theorem.

Consider the following optimization problem

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to } & \mathbf{x}_i \in X_i, \qquad i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}, \end{aligned} \tag{5.1}$$

where for all $i \in \{1,\dots,N\}$, the set $X_i \subseteq \mathbb{R}^{n_i}$ with $n_i \in \mathbb{N}$, the functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ and $\mathbf{g}_i : \mathbb{R}^{n_i} \to \mathbb{R}^S$ with $S \in \mathbb{N}$. The symbol $\preceq$ (and, consistently, for other sides) means that the inequality holds componentwise and we denote by $\mathbf{0}$ the vector $[0,\dots,0]^\top \in \mathbb{R}^S$.

Assumption 5.2.1. *For all $i \in \{1,\dots,N\}$, each function f_i is convex, and each X_i is a non-empty, compact, convex set. Moreover, $\mathbf{g}_i$ is a componentwise convex function, i.e., for all $s \in \{1,\dots,S\}$, each component $\mathbf{g}_{is} : \mathbb{R}^{n_i} \to \mathbb{R}$ of $\mathbf{g}_i$ is a convex function.* □

The following assumption is the well-known Slater's constraint qualification.

Assumption 5.2.2. *There exist $\bar{\mathbf{x}}_1 \in \mathbf{relint}(X_1), \dots, \bar{\mathbf{x}}_N \in \mathbf{relint}(X_N)$ such that $\sum_{i=1}^{N} \mathbf{g}_i(\bar{\mathbf{x}}_i) \prec \mathbf{0}$.* □

These assumptions are quite standard and guarantee that problem (5.1) admits at least an optimal solution $\mathbf{x}^\star = (\mathbf{x}_1^\star, \dots, \mathbf{x}_N^\star)$ such that its optimal cost is $\sum_{i=1}^{N} f_i(\mathbf{x}_i^\star) = f^\star$. Moreover, a dual approach can be applied since strong duality holds. It is worth mentioning that we have assumed that $\sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}$ admits a strictly feasible point, while each constraint $\mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}$ may not.

We consider a network of N processors communicating according to a *connected* and *undirected* graph $\mathcal{G} = (\{1,\dots,N\}, \mathcal{E})$, where $\mathcal{E} \subseteq \{1,\dots,N\} \times \{1,\dots,N\}$ is the set of edges. Edge (i,j) models the fact that node i sends information to j. Note that, being the graph undirected, for each $(i,j) \in \mathcal{E}$, then also $(j,i) \in \mathcal{E}$. We denote by $|\mathcal{E}|$ the cardinality of $\mathcal{E}$ and by $\mathcal{N}_i$ the set of *neighbors* of node i in $\mathcal{G}$, i.e., $\mathcal{N}_i = \{j \in \{1,\dots,N\} \mid (i,j) \in \mathcal{E}\}$.

Each node i knows only f_i, $\mathbf{g}_i$ and X_i, and aims at estimating its portion $\mathbf{x}_i^\star$ of an optimal solution $\mathbf{x}^\star$ of (5.1) by means of local communication only. The distributed algorithm proposed in this chapter is a strategy to achieve this goal.

We are now ready to to state our Relaxation and Successive Distributed Decomposition method (RSDD).

Informally, the algorithm consists of an iterative two-step procedure. First, each node $i \in \{1, \dots, N\}$ stores a set of variables $((\mathbf{x}_i, \rho_i), \boldsymbol{\mu}_i)$ obtained as a primal-dual optimal solution pair of problem (5.2), which mimics a local version of the centralized problem (5.1). Here, the coupling with the other nodes in the original formulation is replaced by a local term depending only on neighboring variables $\boldsymbol{\lambda}_{ij}$ and $\boldsymbol{\lambda}_{ji}$, $j \in \mathcal{N}_i$. Moreover, this local version of the coupling constraint is also relaxed, i.e., a positive violation $\rho_i \mathbf{1}$ is allowed, where we denote by $\mathbf{1}$ the vector $[1, \dots, 1]^\top \in \mathbb{R}^S$. Finally, the local cost f_i plus a scaled violation $M\rho_i$, $M > 0$, is minimized. The variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$, are updated in the second step according to a suitable linear law that combines neighboring $\boldsymbol{\mu}_i$ as in (5.3). Nodes use a step-size denoted by γ^t and can initialize the variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$ to arbitrary values. In the Distributed Algorihtm 10 we formally state the distributed optimization algorithm from the perspective of node i.

Distributed Algorithm 10 RSDD

Processor states: $\mathbf{x}_i$, ρ_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\lambda}_{ij}$ for $j \in \mathcal{N}_i$

Evolution:

Gather $\boldsymbol{\lambda}_{ji}^t$ from $j \in \mathcal{N}_i$

Compute $\big((\mathbf{x}_i^{t+1}, \rho_i^{t+1}), \boldsymbol{\mu}_i^{t+1}\big)$ as a primal-dual optimal solution pair of

$$\begin{aligned} \min_{\mathbf{x}_i, \rho_i} \; & f_i(\mathbf{x}_i) + M\rho_i \\ \text{subj. to } & \rho_i \geq 0, \;\; \mathbf{x}_i \in X_i \\ & \mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \big(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t\big) \preceq \rho_i \mathbf{1} \end{aligned} \tag{5.2}$$

Gather μ_j^{t+1} from $j \in \mathcal{N}_i$

Update for all $j \in \mathcal{N}_i$

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t - \gamma^t \big(\boldsymbol{\mu}_i^{t+1} - \boldsymbol{\mu}_j^{t+1}\big) \tag{5.3}$$

As already mentioned, each agent i aims at computing an optimal strategy by means of local interaction only. The proposed distributed algorithm is a protocol in which agents exchange only the vectors $\boldsymbol{\mu}_i^t$ and $\boldsymbol{\lambda}_{ij}^t$ without explicitly communicating the current estimates of their local decision variables, $\mathbf{x}_i^t$. This is an important, appealing feature of the RSDD distributed algorithm since it guarantees privacy in the network.

In order to gain more intuition about the algorithmic evolution, at this point we provide an informal interpretation, supported by Figure 5.1, of the local optimization step in (5.2).

Agent i, due to its partial knowledge, can only optimize with respect to its own decision variable $\mathbf{x}_i$. Thus, it can solve an instance of problem (5.1) in which all the other variables in the network have a given value $\mathbf{x}_j^t$ for $j \in \{1, \dots, N\} \setminus \{i\}$. Thus, the cost function reduces to f_i only. As for the coupling constraint, it describes how the resources are allocated to all the agents at each iteration t. In the figure, we show a possible instance of a feasible allocation: in blue we depicted the resource assigned to agent i while in red the resources given to all the other agents. When agent i optimizes its local variable $\mathbf{x}_i$ only, it can "play" with the "available resource slot" given by $-\sum_{j \neq i} \mathbf{g}_j(\mathbf{x}_j^t)$. Since, at each iteration, the current allocation is in general not optimal, this constraint might be too restrictive. In fact, it can slow down (and even stall) the optimization process by easily leading to infeasibility of the local problem (5.2) when ρ_i is set to 0. On this regard, recall that we do not assume feasibility of every $\mathbf{g}_i$ independently. Also, it is worth noting that such "available resource slot" depends on the entire network's variables, and, thus, it is not an easily available information in a distributed scenario. Thus, we propose a strategy in which at each iteration t, each agent i replaces the term $-\sum_{j \neq i} \mathbf{g}_j(\mathbf{x}_j^t)$ in the coupling with $\sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t\right)$. Notice that this term can be computed locally at each node by communicating with neighboring agents only. Moreover, each agent i is allowed for a violation $\rho_i \mathbf{1}$ of the local version of the coupling constraint. At the same time, this violation is penalized in order to encourage it to eventually converge to zero. This intuitive description will be rigorously derived and proven in the following sections.

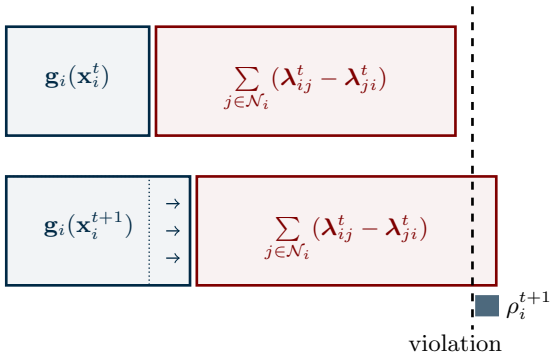


Figure 5.1: Graphical representation of the coupling relaxation.

We are now ready to state the main result of the chapter, namely the convergence of the RSDD distributed algorithm. We start by formalizing the assumption that the step-size should satisfy.

Assumption 5.2.3. *The sequence $\{\gamma^t\}_{t \geq 0}$, with each $\gamma^t \geq 0$, satisfies the conditions $\sum_{t=0}^{\infty} \gamma^t = \infty$, $\sum_{t=0}^{\infty} \left(\gamma^t\right)^2 < \infty$.* □

The convergence theorem is stated below.

Theorem 5.2.4. *Let Assumption 5.2.1 and 5.2.2 hold, and let the step-size γ^t satisfy Assumption 5.2.3. Moreover, letting $\boldsymbol{\mu}^\star$ be an optimal solution of the dual of problem (5.1), assume $M > \|\boldsymbol{\mu}^\star\|_1$. Consider a sequence $\{\mathbf{x}_i^t, \rho_i^t\}_{t\geq 0}$, $i \in \{1, \dots, N\}$, generated by the RSDD distributed algorithm. Then, the following holds:*

1. *$\left\{\sum_{i=1}^N \left(f_i(\mathbf{x}_i^t) + M\rho_i^t\right)\right\}_{t\geq 0}$ converges to the optimal cost $f^\star$ of (5.1);*
2. *every limit point of $\{\mathbf{x}_i^t\}_{t\geq 0}$, $i \in \{1, \dots, N\}$, is a primal optimal (feasible) solution of (5.1).* □

The proof is given in Section 5.4.2.

Remark 5.2.5. *When $f_i = 0$ for all $i \in \{1, \dots, N\}$, then the RSDD algorithm becomes a distributed algorithm for solving a feasibility problem, i.e., for finding $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ such that $\mathbf{x}_i \in X_i$, for all $i \in \{1, \dots, N\}$ and $\sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}$.* □

5.3 Algorithm Analysis: Relaxation and Duality Tour

In this section we present a constructive derivation of our distributed algorithm. The methodology relies on a proper relaxation of the original problem and on the derivation of a sequence of equivalent dual problems. We point out that, although based on a relaxation, the proposed algorithm exploits such a relaxation to solve *exactly* the original problem formulation in a distributed way.

5.3.1 First Duality Step and Relaxation Approach

We start our duality tour by deriving the dual problem of (5.1) and a restricted version necessary for the algorithm derivation.

Let $\boldsymbol{\mu} \succeq \mathbf{0} \in \mathbb{R}^S$, be a Lagrange multiplier associated to the inequality constraint $\sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}$ in (5.1). Then, the partial Lagrangian[1] of problem (5.1) is given by

$$\mathcal{L}_1(\mathbf{x}_1, \dots, \mathbf{x}_N, \boldsymbol{\mu}) = \sum_{i=1}^N f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top \sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i) = \sum_{i=1}^N \Big(f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top \mathbf{g}_i(\mathbf{x}_i)\Big),$$

[1] We have a "partial Lagrangian" when we do not dualize all the constraints. Here the local constraints $\mathbf{x}_i \in X_i$, $i \in \{1, \dots, N\}$, are not dualized.

and its dual function $q : \mathbb{R}^S \to \mathbb{R}$ is defined as

$$q(\boldsymbol{\mu}) = \min_{\mathbf{x}_1 \in X_1, \ldots, \mathbf{x}_N \in X_N} \mathcal{L}_1(\mathbf{x}_1, \ldots, \mathbf{x}_N, \boldsymbol{\mu}).$$

The presence of compact constraints $\mathbf{x}_i \in X_i$ for all $i \in \{1, ..., N\}$ guarantees that the domain of q is the entire $\mathbb{R}^S$.

The minimization with respect to $\mathbf{x}_i$, $i \in \{1, \ldots, N\}$, splits over i, so the dual problem of (5.1) can be written as

$$\begin{aligned} &\max_{\boldsymbol{\mu} \in \mathbb{R}^S} \ \sum_{i=1}^{N} q_i(\boldsymbol{\mu}) \\ &\text{subj. to } \boldsymbol{\mu} \succeq \mathbf{0} \end{aligned} \tag{5.4}$$

with

$$q_i(\boldsymbol{\mu}) = \min_{\mathbf{x}_i \in X_i} \Big(f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top \mathbf{g}_i(\mathbf{x}_i) \Big), \tag{5.5}$$

for all $i \in \{1, \ldots, N\}$.

In light of Assumptions 5.2.1 and 5.2.2, problem (5.1) is feasible and has finite optimal cost, say $f^\star$. Moreover, the Slater's condition holds and, thus, the strong duality theorem for convex inequality constraints, [9, Proposition 5.3.1], applies, ensuring that strong duality holds, i.e., that problems (5.1) and (5.4) have the same optimal cost $f^\star = q^\star$.

It is worth noting that being the optimal cost of (5.4) a finite value $q^\star$, then it is attained at some $\boldsymbol{\mu}^\star \succeq \mathbf{0}$, i.e., $q(\boldsymbol{\mu}^\star) = q^\star$.

Finally, we recall that, since $q(\boldsymbol{\mu}) = \sum_{i=1}^{N} q_i(\boldsymbol{\mu})$ is the dual function of (5.1), problem (5.4) is concave on its convex domain, which can be shown to be the entire $\boldsymbol{\mu} \succeq \mathbf{0}$.

With the dual problem at hand, several existing algorithms can be applied to directly solve (5.4) in a distributed way, see e.g., the distributed projected subgradient [64]. However, we point out that such approaches do not guarantee primal recovery and additional tricks must be employed to regain it, e.g., running average mechanisms.

We propose an alternative approach that will give rise to a distributed algorithm which overtakes these issues. Let us introduce an optimization problem similar to (5.4), given by

$$\begin{aligned} &\max_{\boldsymbol{\mu} \in \mathbb{R}^S} \ \sum_{i=1}^{N} q_i(\boldsymbol{\mu}) \\ &\text{subj. to } \boldsymbol{\mu} \succeq \mathbf{0},\ \boldsymbol{\mu}^\top \mathbf{1} \leq M, \end{aligned} \tag{5.6}$$

where M is a positive scalar and $\mathbf{1} = [1, \ldots, 1]^\top$. This problem is a *restricted* version of problem (5.4). Here, in fact, an additional constraint,

namely $\boldsymbol{\mu}^\top \mathbf{1} \leq M$, has been added to (5.4). It is worth mentioning that this restriction makes the constraint set of (5.6) a compact set. Although this step may seem not motivated at this point of the discussion, its necessity will be clear from the following steps of the analysis (see also Section 5.4.3 for a dedicated discussion).

Notice that, if M is sufficiently large, the presence of the constraint $\boldsymbol{\mu}^\top \mathbf{1} \leq M$ in (5.6) will not alter the optimal solutions of the unrestricted problem (5.4). The next result formally establishes the relation between problem (5.6) and problem (5.4).

Lemma 5.3.1. *Let $\boldsymbol{\mu}^\star$ be an optimal solution of problem* (5.4) *and M be a positive scalar satisfying $M > \|\boldsymbol{\mu}^\star\|_1$. Then, problem* (5.6) *and problem* (5.4) *have the same optimal cost, namely $q^\star = f^\star$. Moreover, $\boldsymbol{\mu}^\star$ is an optimal solution also for problem* (5.6)*.*

Proof. The constraint set $\{\boldsymbol{\mu} \succeq \mathbf{0} \mid \boldsymbol{\mu}^\top \mathbf{1} \leq M\}$ is a restriction of the constraint set $\boldsymbol{\mu} \succeq \mathbf{0}$ of problem (5.4), thus the optimal cost of (5.6) is greater than or equal to the optimal cost of (5.4). But, since by construction the constraint set $\{\boldsymbol{\mu} \succeq \mathbf{0} \mid \boldsymbol{\mu}^\top \mathbf{1} \leq M\}$ contains at least one optimal solution of problem (5.4), namely $\boldsymbol{\mu}^\star$, then the optimal cost of problem (5.6) is $q^\star$ and is at least attained at $\boldsymbol{\mu}^\star$, so that the proof follows. □

With the dual problem (5.4) and its restricted version (5.6) within our reach, one may wonder about the connection between their primal counterparts. Next, we show that the restricted problem (5.6) is the dual of a *relaxed* version of original primal problem (5.1).

Lemma 5.3.2. *The following optimization problem*

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N,\rho} \ & \sum_{i=1}^{N} f_i(\mathbf{x}_i) + M\rho \\ \text{subj. to } & \rho \geq 0,\ \mathbf{x}_i \in X_i,\ i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \preceq \rho \mathbf{1} \end{aligned} \tag{5.7}$$

is the dual of problem (5.6)*. Moreover, strong duality holds, so that it has optimal cost $f^\star$.*

Proof. We show that problem (5.6) is the dual of (5.7). Clearly the vice-versa holds because the cost function is (convex and) closed [9, Chapter 5].

Consider the (partial) Lagrangian of (5.7) given by

$$\begin{aligned}\mathcal{L}_2(\mathbf{x}_1,\dots,\mathbf{x}_N,\rho,\boldsymbol{\mu}) &= \sum_{i=1}^N f_i(\mathbf{x}_i) + M\rho + \boldsymbol{\mu}^\top\Big(\sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i) - \rho\mathbf{1}\Big)\\ &= \sum_{i=1}^N \Big(f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top\mathbf{g}_i(\mathbf{x}_i)\Big) + \rho\big(M - \boldsymbol{\mu}^\top\mathbf{1}\big).\end{aligned}$$

Then, the dual function is

$$\begin{aligned}q_R(\boldsymbol{\mu}) &= \min_{\mathbf{x}_1\in X_1,\dots,\mathbf{x}_N\in X_N,\rho\geq 0} \mathcal{L}_2(\mathbf{x}_1,\dots,\mathbf{x}_N,\rho,\boldsymbol{\mu})\\ &= \begin{cases}\displaystyle\sum_{i=1}^N \min_{\mathbf{x}_i\in X_i}\Big(f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top\mathbf{g}_i(\mathbf{x}_i)\Big) & \text{if } M-\boldsymbol{\mu}^\top\mathbf{1}\geq 0\\ -\infty & \text{otherwise}\end{cases}\\ &= \begin{cases}\displaystyle\sum_{i=1}^N q_i(\boldsymbol{\mu}) & \text{if } M-\boldsymbol{\mu}^\top\mathbf{1}\geq 0\\ -\infty & \text{otherwise,}\end{cases}\end{aligned}$$

where each $q_i(\boldsymbol{\mu})$ is the same defined in (5.5). The maximization of the dual function $q_R(\boldsymbol{\mu})$ on its domain turns out to be the maximization of $\sum_{i=1}^N q_i(\boldsymbol{\mu})$ over $\{\boldsymbol{\mu}\succeq\mathbf{0}\mid\boldsymbol{\mu}^\top\mathbf{1}\leq M\}$, which is problem (5.6), and the proof follows. □

Notice that problem (5.7) is a relaxation of problem (5.1) since we allow for a positive violation of the coupling constraint. At the same time, the violation ρ is penalized with a scaling factor M in order to discourage it. The variable ρ resembles the ρ_i introduced in the distributed algorithm. However, as it will be clear from the forthcoming analysis, ρ_i is *not* a local estimate of ρ, but it rather represents the local contribution of agent i to the common violation ρ.

The following result characterizes how the original primal problem (5.1) and its relaxed version (5.7) are related.

Proposition 5.3.3. *Let M be such that $M > \|\boldsymbol{\mu}^\star\|_1$ with $\boldsymbol{\mu}^\star$ an optimal solution of the dual of problem* (5.1)*. The optimal solutions of the relaxed problem* (5.7) *are in the form $(\mathbf{x}_1^\star,\dots,\mathbf{x}_N^\star,0)$, where $(\mathbf{x}_1^\star,\dots,\mathbf{x}_N^\star)$ is an optimal solution of* (5.1)*, i.e., solutions of* (5.7) *must have $\rho^\star = 0$.*

Proof. To prove the statement we first notice that problem (5.7) is the epigraph formulation of

$$\begin{aligned}\min_{\mathbf{x}_1,\dots,\mathbf{x}_N}\ & \sum_{i=1}^N f_i(\mathbf{x}_i) + M\max\Big\{0,\sum_{i=1}^N \mathbf{g}_{i1}(\mathbf{x}_i),\dots,\sum_{i=1}^N \mathbf{g}_{iS}(\mathbf{x}_i)\Big\}\\ \text{subj. to}\ & \mathbf{x}_i\in X_i,\ i\in\{1,\dots,N\}.\end{aligned} \tag{5.8}$$

where $\mathbf{g}_{is}$ denotes the s-th component of $\mathbf{g}_i$. Problems (5.1) and (5.8) enjoy the same structure as the ones considered in Proposition A.3.1. By Assumption 5.2.2, problem (5.7) (and thus (5.8)) satisfies the assumptions for strong duality, thus we can invoke Proposition A.3.1, with $c = M$, to conclude that problems (5.1) and (5.8) have the same optimal solutions, thus completing the proof. □

Remark 5.3.4 (Alternative restrictions)**.** *Other choices for the restriction of the domain* $\boldsymbol{\mu} \succeq \mathbf{0}$ *of* (5.4) *can be considered. For instance, one can consider upper bounds in the form* $\boldsymbol{\mu} \preceq M\mathbf{1}$ *or* $\boldsymbol{\mu} \preceq [M_1, \dots, M_S]^\top$*. As one might expect, the specific constraint restriction gives rise to different forms of the relaxed primal problem* (5.7)*.* □

5.3.2 Second Dual Problem Derivation

At this point, we continue our duality tour in order to design an algorithm that solves problem (5.6) instead of the unrestricted dual problem (5.4).

In order to make problem (5.6) amenable for a distributed solution, we enforce a sparsity structure that matches the network. To this end, we introduce copies of the common optimization variable $\boldsymbol{\mu}$ and we copy also its domain. Moreover, we enforce coherence constraints among the copies $\boldsymbol{\mu}_i$ having the sparsity of the connected graph $\mathcal{G}$, thus obtaining

$$\begin{aligned} \max_{\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_N} \ & \sum_{i=1}^{N} q_i(\boldsymbol{\mu}_i) \\ \text{subj. to } & \boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M, \quad i \in \{1, \dots, N\} \\ & \boldsymbol{\mu}_i = \boldsymbol{\mu}_j, \qquad\qquad (i,j) \in \mathcal{E}. \end{aligned} \tag{5.9}$$

Being problem (5.9) an equivalent version of problem (5.6), it has the same optimal cost $q^\star = f^\star$.

In order to solve problem (5.9), we would like to use a dual decomposition approach with the aim of obtaining a distributed algorithm. That is, the leading idea is to derive the dual of problem (5.9) and apply a subgradient method to solve it.

We start deriving the dual problem of (5.9) by dualizing only the coherence constraints. Thus, we write the partial Lagrangian

$$\mathcal{L}_3(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N, \boldsymbol{\Lambda}) = \sum_{i=1}^{N} \Big(q_i(\boldsymbol{\mu}_i) + \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ij}^\top (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \Big), \tag{5.10}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{S \cdot |\mathcal{E}|}$ is the vector stacking each Lagrange multiplier $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^S$, with $(i,j) \in \mathcal{E}$, associated to the constraint $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j = \mathbf{0}$. Notice that here we did not dualize the constraints $\{\boldsymbol{\mu}_i \succeq \mathbf{0} \mid \boldsymbol{\mu}_i^\top \mathbf{1} \leq M\}$, since they are local for the agents.

Since the communication graph $\mathcal{G}$ is undirected, we can exploit the symmetry of the constraints. In fact, for each $(i,j) \in \mathcal{E}$ we also have $(j,i) \in \mathcal{E}$, and, expanding all the terms in (5.10), for given i and j, we always have both the terms $\boldsymbol{\lambda}_{ij}^\top(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$ and $\boldsymbol{\lambda}_{ji}^\top(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)$. Thus, after some simple algebraic manipulations, we can rephrase (5.10) as

$$\mathcal{L}_3(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N, \boldsymbol{\Lambda}) = \sum_{i=1}^{N} \Big(q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\top \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) \Big),$$

which is separable with respect to $\boldsymbol{\mu}_i$, $i \in \{1, \dots, N\}$. Thus, the dual function is

$$\begin{aligned} \eta(\boldsymbol{\Lambda}) &= \sup_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M, \forall\, i \in \{1,\dots,N\}} \mathcal{L}_3(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N, \boldsymbol{\Lambda}) \\ &= \sum_{i=1}^{N} \eta_i\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}\big), \end{aligned} \tag{5.11}$$

where, for all $i \in \{1, \dots, N\}$,

$$\eta_i\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}\big) = \sup_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M} \Big(q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\top \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) \Big). \tag{5.12}$$

Finally, by denoting the domain of η as

$$D_\eta = \Big\{ \boldsymbol{\Lambda} \in \mathbb{R}^{S \cdot |\mathcal{E}|} \mid \eta(\boldsymbol{\Lambda}) < +\infty \Big\},$$

the dual of problem (5.9) is

$$\min_{\boldsymbol{\Lambda} \in D_\eta} \eta(\boldsymbol{\Lambda}) = \min_{\boldsymbol{\Lambda} \in D_\eta} \sum_{i=1}^{N} \eta_i\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}\big). \tag{5.13}$$

Since problem (5.13) is the dual of a concave program, then it is a convex (constrained) problem. Moreover, its cost function $\eta(\boldsymbol{\Lambda})$ is very structured since it is a sum of contributions η_i and each of them depends only on neighboring variables.

In the next lemma we characterize the domain of problem (5.13).

Lemma 5.3.5. *The domain D_η of η in* (5.11) *is $\mathbb{R}^{S \cdot |\mathcal{E}|}$. Thus optimization problem* (5.13) *is unconstrained.*

Proof. We show that each $\eta_i\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}\big)$ is finite for all $\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}$. Each function $q_i(\boldsymbol{\mu}_i)$ is concave on its domain $\boldsymbol{\mu}_i \succeq \mathbf{0}$ for all $i \in \{1, \dots, N\}$. In fact, from the definition of q_i in (5.5), we notice that it is obtained as the minimization over a nonempty compact set X_i of the function $f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \mathbf{g}_i(\mathbf{x}_i)$. Such a function is concave (in fact linear) in $\boldsymbol{\mu}_i$, thus, following

the proof of [9, Proposition 5.1.2], we can conclude that every q_i is concave over its convex domain $\boldsymbol{\mu}_i \succeq \mathbf{0}$. For each $i \in \{1, \dots, N\}$, the function η_i as defined in (5.12) is obtained by maximizing a (concave) continuous function (q_i plus a linear term) over a compact set (containing the domain of q_i, namely $\boldsymbol{\mu}_i \succeq \mathbf{0}$), and thus it is always finite, so that the proof follows. □

Remark 5.3.6. *Although it represents a minor aspect, we notice that q_i may be seen as the dual function of*

$$\begin{aligned} \min_{\mathbf{x}_i} \ & f_i(\mathbf{x}_i) \\ \text{subj. to } & \mathbf{x}_i \in X_i, \\ & \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}, \end{aligned}$$

which might be not even feasible since we do not assume any condition on the feasibility of $\mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}$. Infeasibility of this problem does not affect the structure of D_η. □

It is worth noting that Lemma 5.3.5 strongly relies on the compactness of $\{\boldsymbol{\mu}_i \succeq \mathbf{0} \mid \boldsymbol{\mu}_i^\top \mathbf{1} \leq M\}$. This means that, without the primal relaxation, D_η is not guaranteed to be $\mathbb{R}^{S \cdot |\mathcal{E}|}$. In Section 5.4.3, we better clarify this aspect.

In the following we characterize the optimization problem (5.13).

Lemma 5.3.7. *Let M be such that $M > \|\boldsymbol{\mu}^\star\|_1$ with $\boldsymbol{\mu}^\star$ an optimal solution of the dual of problem* (5.1)*. Then, problem* (5.13)*, which is the dual of problem* (5.9)*, has a bounded optimal cost, call it $\eta^\star$, and strong duality holds. Moreover,*

$$\eta^\star = f^\star,$$

with $f^\star$ the optimal solution of the original primal problem (5.1)*.*

Proof. By Lemma 5.3.1 (5.9) is equivalent to (5.4) and has finite optimal cost equal to $q^\star$. Since each $q_i(\boldsymbol{\mu}_i)$ is concave with respect to its variable $\boldsymbol{\mu}_i$, as shown in the proof of Lemma 5.3.5, then also $\sum_{i=1}^N q_i(\boldsymbol{\mu}_i)$ is concave with respect to $(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N)$. Thus, since the domain $\{(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N) \mid \boldsymbol{\mu}_i \succeq \mathbf{0} \mid \boldsymbol{\mu}_i^\top \mathbf{1} \leq M, i \in \{1, \dots, N\}\}$, is polyhedral, by [11, Theorem 6.4.2] strong duality between problem (5.9) and its dual (5.13) holds. Thus, $\eta^\star = q^\star$ is bounded. The second statement follows by strong duality between problems (5.1) and (5.4), which concludes the proof. □

5.3.3 Distributed Subgradient Method

We detail in this subsection how to explicitly design a distributed dual decomposition algorithm to solve problem (5.6) based on a subgradient iteration applied to problem (5.13).

Exploiting the separability of η in (5.11), we recall how to compute each component of a subgradient of η at a given $\boldsymbol{\Lambda} \in \mathbb{R}^{S\cdot|\mathcal{E}|}$, see e.g., [11, Section 8.1]. That is, it holds

$$\frac{\tilde{\partial}\eta(\boldsymbol{\Lambda})}{\partial \boldsymbol{\lambda}_{ij}} = \boldsymbol{\mu}_i^\star - \boldsymbol{\mu}_j^\star, \tag{5.14}$$

where $\frac{\tilde{\partial}\eta(\cdot)}{\partial \boldsymbol{\lambda}_{ij}}$ denotes the component associated to the variable $\boldsymbol{\lambda}_{ij}$ of a subgradient of η, and

$$\boldsymbol{\mu}_k^\star \in \underset{\boldsymbol{\mu}_k \succeq \mathbf{0}, \boldsymbol{\mu}_k^\top \mathbf{1} \leq M}{\operatorname{argmax}} \Big(q_k(\boldsymbol{\mu}_k) + \boldsymbol{\mu}_k^\top \sum_{h \in \mathcal{N}_k} (\boldsymbol{\lambda}_{kh} - \boldsymbol{\lambda}_{hk}) \Big), \tag{5.15}$$

for $k = i, j$.

Having recalled how to compute subgradients of η, we are ready to summarize how the subgradient method reads when applied to problem (5.13).

At each iteration t, each node $i \in \{1, \dots, N\}$:

(S1) receives $\boldsymbol{\lambda}_{ji}^t$, $j \in \mathcal{N}_i$, and computes a subgradient $\boldsymbol{\mu}_i^{t+1}$ by solving

$$\max_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M} \Big(q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\top \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \Big); \tag{5.16}$$

(S2) receives the updated $\boldsymbol{\mu}_j^{t+1}$, $j \in \mathcal{N}_i$ and updates $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$, via

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t - \gamma^t (\boldsymbol{\mu}_i^{t+1} - \boldsymbol{\mu}_j^{t+1}),$$

where γ^t is the step-size.

Notice that (S1)–(S2) is a distributed algorithm, i.e., it can be implemented by means of local computation and communication steps without any centralized stage. However, we want to stress that the algorithm is *not* implementable as it is written, since functions q_i are still not explicitly available.

The next lemma gives a condition on the subgradient of the convex cost function η.

Lemma 5.3.8. *For every $\boldsymbol{\Lambda}^t \in \mathbb{R}^{S\cdot|\mathcal{E}|}$ with components $\boldsymbol{\lambda}_{ij}^t \in \mathbb{R}^S$, $\forall\, (i,j) \in \mathcal{E}$, any subgradient of the function η at $\boldsymbol{\Lambda}^t$ is bounded.*

Proof. To prove the lemma, we show that each component $\frac{\tilde{\partial}\eta(\boldsymbol{\Lambda}^t)}{\partial \boldsymbol{\lambda}_{ij}}$ of $\frac{\tilde{\partial}\eta(\boldsymbol{\Lambda}^t)}{\partial \boldsymbol{\Lambda}}$ is bounded. Using (5.14), this is equivalent to showing that $\boldsymbol{\mu}_i^{\star t}$ and $\boldsymbol{\mu}_j^{\star t}$ are bounded and, hence, their difference. Since, from equation (5.15) the two are obtained as maxima of a concave function over a compact domain, they are always finite for all $\boldsymbol{\Lambda}^t$, so that the proof follows. □

The above lemma shows that argmax in (5.15) is well posed.

5.4 Algorithm Analysis: Convergence Proof

This section is devoted to proving the convergence result, formally stated in Theorem 5.2.4, of the RSDD distributed algorithm.

5.4.1 Preparatory Results

In this subsection we give two intermediate results that represent building blocks for the convergence proof given in Section 5.4.2. The next lemma is instrumental to the second one.

Lemma 5.4.1. *Consider the optimization problem*

$$\begin{aligned} \max_{\boldsymbol{\mu}_i} \; & f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \Big(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \Big) \\ \text{subj. to } & \boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M, \end{aligned} \tag{5.17}$$

with given $\mathbf{x}_i$, $\boldsymbol{\lambda}_{ij}^t$ *and* $\boldsymbol{\lambda}_{ji}^t$, $j \in \mathcal{N}_i$, *and* $M > 0$. *Then, its dual problem is*

$$\begin{aligned} \min_{\rho_i} \; & f_i(\mathbf{x}_i) + M\rho_i \\ \text{subj. to } & \rho_i \geq 0 \\ & \mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \preceq \rho_i \mathbf{1}, \end{aligned} \tag{5.18}$$

and strong duality holds.

Proof. First, since $\mathbf{x}_i$, $\boldsymbol{\lambda}_{ij}^t$ and $\boldsymbol{\lambda}_{ji}^t$ are given, problem (5.17) is a feasible *linear* program (the box constraint is nonempty) with compact domain. Thus, both problem (5.17) and its dual have finite optimal cost and strong duality holds.

In order to show that (5.18) is the dual of (5.17), we introduce a multiplier $\rho_i \geq 0$ associated to the constraint $M - \boldsymbol{\mu}_i^\top \mathbf{1} \geq 0$, and we write the partial Lagrangian of (5.17)

$$\mathcal{L}_4(\boldsymbol{\mu}_i, \rho_i) = f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \Big(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \Big) + \rho_i \Big(M - \boldsymbol{\mu}_i^\top \mathbf{1} \Big).$$

Then, we can rearrange the terms to obtain

$$\mathcal{L}_4(\boldsymbol{\mu}_i, \rho_i) = f_i(\mathbf{x}_i) + M\rho_i + \boldsymbol{\mu}_i^\top \Big(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) - \rho_i \mathbf{1} \Big).$$

The dual function $\max_{\boldsymbol{\mu}_i \succeq 0} \mathcal{L}_4(\boldsymbol{\mu}_i, \rho_i)$ is equal to $f_i(\mathbf{x}_i) + M\rho_i$ if $\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) - \rho_i \mathbf{1} \preceq \mathbf{0}$ and $+\infty$ otherwise. Finally, the minimization of the dual function on its domain with respect to $\rho_i \geq 0$ gives problem (5.18) and concludes the proof. □

In the following, we propose a technique to make step (5.16) explicit. By plugging in (5.16) the definition of q_i, given in (5.5), the following max-min optimization problem is obtained:

$$\max_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M} \min_{\mathbf{x}_i \in X_i} \left(f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \left(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \right) \right). \tag{5.19}$$

The next lemma allows us to recast problem (5.19) in terms of the primal-dual optimal solution pair of a suitable optimization problem involving only decision variables of each node i.

Lemma 5.4.2. *Consider the optimization problem*

$$\begin{aligned} \min_{\mathbf{x}_i, \rho_i} \; & f_i(\mathbf{x}_i) + M\rho_i \\ \text{subj. to } & \rho_i \geq 0, \;\; \mathbf{x}_i \in X_i \\ & \mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t \right) \preceq \rho_i \mathbf{1}. \end{aligned} \tag{5.20}$$

A finite primal-dual optimal solution pair of (5.20)*, call it* $\left((\mathbf{x}_i^{t+1}, \rho_i^{t+1}), \boldsymbol{\mu}_i^{t+1} \right)$*, does exist and* $(\mathbf{x}_i^{t+1}, \boldsymbol{\mu}_i^{t+1})$ *is a solution of* (5.19)*.*

Proof. Problem (5.20) is a feasible convex program, in fact $f_i(\mathbf{x}_i) + M\rho_i$ is convex, the set X_i is nonempty, convex and compact, the constraint $\rho_i \geq 0$ is convex as well as the inequality constraint $\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t \right) - \rho_i \mathbf{1} \preceq \mathbf{0}$). Then, by choosing a sufficiently large ρ_i, we can show that the Slater's constraint qualification is satisfied and, thus, strong duality holds. Therefore, a primal-dual optimal solution pair $(\mathbf{x}_i^{t+1}, \rho_i^{t+1}, \boldsymbol{\mu}_i^{t+1})$ of (5.20) exists. Moreover, problem (5.20) can be recast as

$$\min_{\mathbf{x}_i \in X_i} \left(\min_{\rho_i \geq 0, \, \mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \preceq \rho_i \mathbf{1}} f_i(\mathbf{x}_i) + M\rho_i \right).$$

By Lemma 5.4.1, we can substitute the inner minimization with its equivalent dual maximization obtaining

$$\min_{\mathbf{x}_i \in X_i} \max_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M} \left(f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \left(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \right) \right). \tag{5.21}$$

Let $\phi(\mathbf{x}_i, \boldsymbol{\mu}_i) = f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \left(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \right)$ and observe that (i) $\phi(\cdot, \boldsymbol{\mu}_i)$ is closed and convex for all $\boldsymbol{\mu}_i \succeq \mathbf{0}$ (affine transformation of a convex function with compact domain X_i) and (ii) $\phi(\mathbf{x}_i, \cdot)$ is closed and concave since it is a linear function with compact domain ($\{\boldsymbol{\mu}_i \succeq \mathbf{0} \mid \boldsymbol{\mu}_i^\top \mathbf{1} \leq M\}$),

for all $\mathbf{x}_i \in \mathbb{R}^S$. Thus, we can invoke Proposition A.1.2 to switch min and max operators in (5.21), and write

$$\begin{aligned}\min_{\mathbf{x}_i \in X_i} \max_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M} \left(f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \left(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \right) \right) \\ = \max_{\boldsymbol{\mu}_i \succeq \mathbf{0}, \boldsymbol{\mu}_i^\top \mathbf{1} \leq M} \min_{\mathbf{x}_i \in X_i} \left(f_i(\mathbf{x}_i) + \boldsymbol{\mu}_i^\top \left(\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \right) \right).\end{aligned} \tag{5.22}$$

which is (5.19), thus concluding the proof. □

We highlight that problem (5.20) is the local optimization step (5.2) in RSDD distributed algorithm.

Finally, the next corollary makes a connection between the optimal cost of the i-th problem (5.20) and the value of the i-th local term η_i (defined in (5.12)) in the second dual function η (defined in (5.11)).

Corollary 5.4.3. *Let $(\mathbf{x}_i^{t+1}, \rho_i^{t+1})$ be a solution of (5.20) with given $\boldsymbol{\lambda}_{ij}^t$ and $\boldsymbol{\lambda}_{ij}^t$ for $j \in \mathcal{N}_i$. Then*

$$\eta_i\big(\{\boldsymbol{\lambda}_{ij}^t, \boldsymbol{\lambda}_{ji}^t\}_{j \in \mathcal{N}_i}\big) = f_i(\mathbf{x}_i^{t+1}) + M\rho_i^{t+1}, \tag{5.23}$$

with η_i defined in (5.12).

Proof. In the proof of Lemma 5.4.2, we have shown that condition (5.22) holds for all $t \geq 0$. Its left hand side has optimal cost $f_i(\mathbf{x}_i{}^{t+1}) + M\rho_i^{t+1}$, while the one of the right hand side is exactly the definition of $\eta_i\big(\{\boldsymbol{\lambda}_{ij}^t, \boldsymbol{\lambda}_{ji}^t\}_{j \in \mathcal{N}_i}\big)$ in (5.12). Thus, equation (5.22) can be rewritten as

$$f_i(\mathbf{x}_i^{t+1}) + M\rho_i^{t+1} = \eta_i\big(\{\boldsymbol{\lambda}_{ij}^t, \boldsymbol{\lambda}_{ji}^t\}_{j \in \mathcal{N}_i}\big),$$

for all $i \in \{1, \ldots, N\}$, concluding the proof. □

5.4.2 Proof of the Convergence Theorem

To prove statement (i) of the theorem, we show that the RSDD distributed algorithm is an operative way to implement the subgradient method (S1)–(S2) and, that (S1)–(S2) solves problem (5.6).

First, for each $i \in \{1, \ldots, N\}$, we let $\{\boldsymbol{\mu}_i^t\}_{t \geq 0}$ and $\{\boldsymbol{\lambda}_{ij}^t\}_{t \geq 0}$, $j \in \mathcal{N}_i$, be the auxiliary sequences defined in the RSDD distributed algorithm associated to $\{(\mathbf{x}_i^t, \rho_i^t)\}_{t \geq 0}$. From Lemma 5.4.2, at each iteration t a primal-dual optimal solution pair $\big((\mathbf{x}_i^t, \rho_i^t), \boldsymbol{\mu}_i^t\big)$ of (5.2) in fact exists, so that the algorithm is well-posed.

Second, to show that RSDD implements (S1)–(S2) we notice that update (5.3) and (S2) are trivially identical. As for (S1), we have shown that equation (5.19) is an explicit expression for (5.16) in (S1). Thus, by invoking

Lemma 5.4.2, we can conclude that finding the dual part of a primal-dual optimal solution pair of (5.2) corresponds to performing (S1). Therefore, the sequences $\{\boldsymbol{\lambda}_{ij}^t\}_{t\geq 0}$, $(i,j) \in \mathcal{E}$ generated by RSDD and by (S1)–(S2) coincide.

Third, we show that RSDD solves problem (5.13). By Lemma 5.3.8 the subgradients of η are bounded at each $\{\boldsymbol{\lambda}_{ij}^t\}_{t\geq 0}$, $(i,j) \in \mathcal{E}$. Moreover, since the step-size γ^t satisfies Assumption 5.2.3, we can invoke Proposition A.2.1 to conclude that the sequence $\{\boldsymbol{\lambda}_{ij}^t\}_{t\geq 0}$, $(i,j) \in \mathcal{E}$ generated by RSDD (or equivalently by (S1)–(S2)) converges in objective value to the optimal cost $\eta^\star$ of (5.13). To complete the convergence in objective value, we use (5.23) in Corollary 5.4.3 and take the limit as $t \to \infty$, thus obtaining

$$\lim_{t\to\infty} \sum_{i=1}^N \left(f_i(\mathbf{x}_i^{t+1}) + M\rho_i^{t+1} \right) = \lim_{t\to\infty} \sum_{i=1}^N \eta_i(\{\boldsymbol{\lambda}_{ij}^t, \boldsymbol{\lambda}_{ji}^t\}_{j\in\mathcal{N}_i}) = \eta^\star = f^\star,$$

where the last equality follows by Lemma 5.3.7, so that the proof of the first statement is complete.

To prove statement (ii) of the theorem, i.e., the primal recovery property, we start by studying the properties of $(\mathbf{x}_1^t, \dots, \mathbf{x}_N^t, \rho_1^t, \dots, \rho_N^t)$. By construction, for all $t \geq 0$ each pair $(\mathbf{x}_i^t, \rho_i^t)$ satisfies $\mathbf{x}_i^t \in X_i$, $\rho_i^t \geq 0$ and

$$\mathbf{g}_i(\mathbf{x}_i^t) + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t \right) \preceq \rho_i^t \mathbf{1}. \tag{5.24}$$

Thus, by summing (5.24) over $i \in \{1, \dots, N\}$, it follows

$$\sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i^t) + \sum_{i=1}^N \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t \right) \preceq \sum_{i=1}^N \rho_i^t \mathbf{1}, \tag{5.25}$$

for all $t \geq 0$. Let us denote by a_{ij} the (i,j)-th entry of the adjacency matrix associated to the undirected graph $\mathcal{G}$. Then, we can write

$$\begin{aligned} \sum_{i=1}^N \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) &\overset{(a)}{=} \sum_{i=1}^N \sum_{j=1}^N a_{ij}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \\ &= \sum_{i=1}^N \sum_{j=1}^N a_{ij}\boldsymbol{\lambda}_{ij}^t - \sum_{i=1}^N \sum_{j=1}^N a_{ij}\boldsymbol{\lambda}_{ji}^t \overset{(b)}{=} 0, \end{aligned}$$

where (a) follows by writing the sum over neighboring agents in terms of the adjacency matrix, while (b) holds since $\mathcal{G}$ is undirected (so that $a_{ij} = a_{ji}$ for all $(i,j) \in \mathcal{E}$), which implies that the two summations in the second line are identical for all $t \geq 0$. Hence, (5.25) reduces to

$$\sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i^t) \preceq \sum_{i=1}^N \rho_i^t \mathbf{1}, \tag{5.26}$$

for all $t \geq 0$.

Equation (5.26) shows that for all $t \geq 0$ the vector

$$(\mathbf{x}_1^t, \dots, \mathbf{x}_N^t, \rho_1^t, \dots, \rho_N^t)$$

is feasible for the following optimization problem

$$\begin{aligned} \min_{\substack{\mathbf{x}_1,\dots,\mathbf{x}_N \\ \rho_1,\dots,\rho_N}} \; & \sum_{i=1}^N f_i(\mathbf{x}_i) + M \sum_{i=1}^N \rho_i \\ \text{subj. to } & \rho_i \geq 0, \; \mathbf{x}_i \in X_i, \; i \in \{1, \dots, N\} \\ & \sum_{i=1}^N \mathbf{g}_i(\mathbf{x}_i) \preceq \sum_{i=1}^N \rho_i \mathbf{1}. \end{aligned} \tag{5.27}$$

Notice that problem (5.27) is equivalent to problem (5.7) by simply defining $\rho = \sum_{i=1}^N \rho_i$. Thus, at each iteration t the vector $(\mathbf{x}_1^t, \dots, \mathbf{x}_N^t, \sum_{i=1}^N \rho_i^t)$ is feasible for problem (5.7).

We now show that every limit point of the sequence $\{\mathbf{x}_1^t, \dots, \mathbf{x}_N^t, \sum_{i=1}^N \rho_i^t\}_{t\geq 0}$ is feasible, other than optimal, for problem (5.7). By construction, each $\mathbf{x}_i^t \in X_i$ for all $i \in \{1, \dots, N\}$, so that $\{\mathbf{x}_i^t\}_{t\geq 0}$ is bounded. Moreover, from the first statement of the theorem, also the sequence $\{\sum_{i=1}^N \rho_i^t\}_{t\geq 0}$ is bounded since $\{\sum_{i=1}^N f_i(\mathbf{x}_i^t) + M \sum_{i=1}^N \rho_i^t\}_{t\geq 0}$ converges to a finite value $f^\star$. Since the sequence of vectors $\{(\mathbf{x}_1^t, \dots, \mathbf{x}_N^t, \sum_{i=1}^N \rho_i^t)\}_{t\geq 0}$ is bounded, then there exists a sub-sequence of indices $\{t_n\}_{n\geq 0} \subseteq \{t\}_{t\geq 0}$ such that the sub-sequence $\{(\mathbf{x}_1^{t_n}, \dots, \mathbf{x}_N^{t_n}, \sum_{i=1}^N \rho_i^{t_n})\}_{n\geq 0}$ converges to a limit point $(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N, \bar{\rho})$. From the first statement of the theorem we have that $(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N, \bar{\rho})$ satisfies

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

Moreover, since each component of $\mathbf{g}_i$ is a (finite) convex function over $\mathbb{R}^{n_i}$, it is also continuous over any compact subset of $\mathbb{R}^{n_i}$. Thus, by taking the limit as $n \to \infty$ in (5.26) with $t = t_n$, it also holds

$$\sum_{i=1}^N \mathbf{g}_i(\bar{\mathbf{x}}_i) \preceq \bar{\rho}\mathbf{1}. \tag{5.28}$$

By Proposition 5.3.3 it must hold that $(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N, \bar{\rho}) = (\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N, 0)$, i.e., $\bar{\rho} = 0$. Thus, (5.28) holds with $\bar{\rho} = 0$ and, thus, this guarantees that every limit point of $(\mathbf{x}_1^t, \dots, \mathbf{x}_N^t)$ is feasible for the (non-relaxed) coupling constraint in the original problem (5.1) and thus optimal for that problem. So that the proof follows.

5.4.3 Discussion on the Necessity of the Relaxation

We briefly summarize why the dual restriction, and thus its primal relaxation counterpart, is needed. Suppose we do not restrict the original dual problem (5.4), but we still apply the same formal derivation given in the previous sections. Then, the counterpart of (5.11) is $\eta^{\mathtt{NR}}(\boldsymbol{\Lambda}) = \sum_{i=1}^{N} \eta_i^{\mathtt{NR}}\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j\in\mathcal{N}_i}\big)$ with

$$\eta_i^{\mathtt{NR}}\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j\in\mathcal{N}_i}\big) = \sup_{\boldsymbol{\mu}_i \succeq \mathbf{0}} \Big(q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\top \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) \Big),$$

for all $i \in \{1, \dots, N\}$. Finally, by denoting the domain of $\eta^{\mathtt{NR}}$ as $D_\eta^{\mathtt{NR}}$, we have that the counterpart of problem (5.13) is $\min_{\boldsymbol{\Lambda}\in D_{\boldsymbol{\Lambda}}^{\mathtt{NR}}} \eta^{\mathtt{NR}}(\boldsymbol{\Lambda})$.

We notice that the latter problem is a constrained minimization since, differently from the relaxed case, the domain $D_\eta^{\mathtt{NR}}$ does not always coincide with the entire space $\mathbb{R}^{S\cdot|\mathcal{E}|}$ (Cf. Lemma 5.3.5). Thus, to apply the subgradient method we need to adapt (S1)–(S2) by adding a projection step $\boldsymbol{\Lambda}^{t+1} = \big[\widetilde{\boldsymbol{\Lambda}}^{t+1}\big]_{D_\eta^{\mathtt{NR}}}$, where $\widetilde{\boldsymbol{\lambda}}_{ij}^{t+1}$ is the result of (S2) and $[\,\cdot\,]_{D_\eta^{\mathtt{NR}}}$ denotes the Euclidean projection onto $D_\eta^{\mathtt{NR}}$. Notice that the projection onto $D_\eta^{\mathtt{NR}}$ of the entire $\widetilde{\boldsymbol{\Lambda}}^{t+1}$ prevents the distributed implementation. The projection is also needed because otherwise there would be no guarantees about the boundedness of subgradients of $\eta^{\mathtt{NR}}$ (differently from the relaxed case, see Lemma 5.3.8). Analogously, being the set $\{\boldsymbol{\mu}_i \in \mathbb{R}^S \mid \boldsymbol{\mu}_i \succeq \mathbf{0}\}$ not compact, then Lemma 5.4.2 would not hold. The theoretical issue is that switching min and max operators in (5.22) may not be possible due to the non-compact domains (Cf. Proposition A.1.2). In fact, the minimization in (5.20) (which is equation (5.2) in the algorithm) without relaxation (i.e., setting $\rho_i = 0$) may lead to an unfeasible problem due to the constraint $\mathbf{g}_i(\mathbf{x}_i) + \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \preceq \mathbf{0}$ (recall that we assumed only the feasibility of $\sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \preceq \mathbf{0}$).

5.5 Application to Distributed Model Predictive Control

In this section we present a computational study of our RSDD distributed algorithm tailored to an instance of a distributed MPC scheme for microgrid control. Most existing control strategies use a centralized approach, see, e.g., [104]. However, for several reasons, distributed control strategies, that do not require to collect all data at a central coordinator, are highly desirable.

A microgrid consists of several generators, controllable loads, storage devices and a connection to the main grid. In the following, we use the notational convention that energy generation corresponds to positive variables, while energy consumption corresponds to negative variables. In the

following subsection, we specify a model for each component and set-up the optimization problem we aim at solving distributedly.

5.5.1 Microgrid Model

Generators are collected in the set $\texttt{GEN}$. At each time instant τ in a given horizon $[0, T]$, they generate power, denoted by $p^\tau_{\texttt{gen},i}$, that must satisfy magnitude and rate bounds, i.e., for given positive scalars $\underline{p}$, $\bar{p}$, $\underline{r}$ and $\bar{r}$, it must hold, for all $i \in \texttt{GEN}$, $\underline{p} \leq p^\tau_{\texttt{gen},i} \leq \bar{p}$, with $\tau \in [0, T]$, and $\underline{r} \leq p^{\tau+1}_{\texttt{gen},i} - p^\tau_{\texttt{gen},i} \leq \bar{r}$, with $\tau \in [0, T-1]$. The cost to produce power by a generator is modeled as a quadratic function $f^\tau_{\texttt{gen},i} = \alpha_1 p^\tau_{\texttt{gen},i} + \alpha_2 (p^\tau_{\texttt{gen},i})^2$ with α_1 and α_2 positive scalars. Storage devices are collected in $\texttt{STOR}$ and their power is denoted by $p^\tau_{\texttt{stor},i}$ and satisfies bounds and a dynamical constraint given by $-d_{\texttt{stor}} \leq p^\tau_{\texttt{stor},i} \leq c_{\texttt{stor}}$, $\tau \in [0, T]$, $q^{\tau+1}_{\texttt{stor},i} = q^\tau_{\texttt{stor},i} + p^\tau_{\texttt{stor},i}$, $\tau \in [0, T-1]$, and $0 \leq q^\tau_{\texttt{stor},i} \leq q_{\max}$, $\tau \in [0, T]$, where the initial capacity $q^0_{\texttt{stor},i}$ is given and $d_{\texttt{stor}}$, $c_{\texttt{stor}}$ and $q_{\max}$ are positive scalars. There are no costs associated to the stored power. Controllable loads are collected in $\texttt{CONL}$ and their power is denoted by $p^\tau_{\texttt{conl},i}$. A desired load profile $p^\tau_{\texttt{des},i}$ for $p^\tau_{\texttt{conl},i}$ is given and the controllable load incurs in a cost $f^\tau_{\texttt{conl},i} = \beta \max\{0, p^\tau_{\texttt{des},i} - p^\tau_{\texttt{conl},i}\}$, $\beta \geq 0$, if the desired load is not satisfied. Finally, the device $i = N$ is the connection node with the main grid; its power is denoted as $p^\tau_{\texttt{tr}}$ and must satisfy $|p^\tau_{\texttt{tr}}| \leq E$, $\tau \in [0, T]$. The power trading cost is modeled as $f^\tau_{\texttt{tr}} = -c_1 p^\tau_{\texttt{tr}} + c_2 |p^\tau_{\texttt{tr}}|$, with c_1 and c_2 positive scalars corresponding to the price and a general transaction cost respectively.

The power network must satisfy a given power demand D^τ modeled by a coupling constraint among the units

$$\sum_{i \in \texttt{GEN}} p^\tau_{\texttt{gen},i} + \sum_{i \in \texttt{STOR}} p^\tau_{\texttt{stor},i} + \sum_{i \in \texttt{CONL}} p^\tau_{\texttt{conl},i} + p^\tau_{\texttt{tr}} - D^\tau = 0,$$

for all $\tau \in [0, T]$. Reasonably, we assume D^τ to be known only by the connection node $\texttt{tr}$.

5.5.2 Numerical Results

We consider a heterogeneous network of $N = 10$ units with 4 generators, 3 storage devices, 2 controllable loads and 1 connection to the main grid. We assume that in the distributed MPC scheme each unit predicts its power generation strategy over a horizon of $T = 12$ slots. In order to fit the microgrid control problem in our set-up, we let each $\mathbf{x}_i$ be the whole trajectory over the prediction horizon $[0, T]$, e.g., $\mathbf{x}_i = [p^0_{\texttt{gen},i}, \dots, p^T_{\texttt{gen},i}]^\top$, for all the generators $i \in \texttt{GEN}$ and, consistently, for the other device types. As for the cost functions we define $f_i(\mathbf{x}_i) = \sum_{\tau=0}^{T} f^\tau_{\texttt{gen},i}(p^\tau_{\texttt{gen},i})$ for $i \in \texttt{GEN}$ and, consistently, for the other device types. The local X_i are given by the constraint described above for each unit type.

In Figure 5.2 we show the algorithmic evolution of the sum of the penalty parameters ρ_i^t and the maximum violation of the coupling constraint at each iteration t. The $\sum_{i=1}^N \rho_i^t$ asymptotically goes to zero as claimed in Theorem 5.2.4. In this particular instance we also notice that, after the very first iterations, the generated points are strictly feasible for the coupling constraints and in the limit they hit the boundary. However, we point out that strict feasibility of the coupling constraints is obtained even if $\sum_{i=1}^N \rho_i^t$ becomes zero only asymptotically.

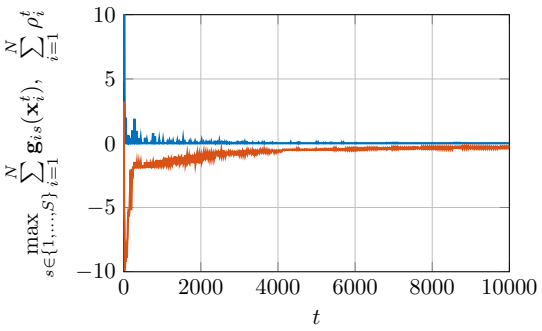


Figure 5.2: Evolution of $\max_{s\in\{1,\dots,S\}} \sum_{i=1}^N g_{is}(\mathbf{x}_i^t)$ (red) and $\sum_{i=1}^N \rho_i^t$ (blue).

In Figure 5.3 we show how $\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$ compares with the unknown part of the coupling constraints of each agent i, namely $\sum_{j\neq i} \mathbf{g}_j(\mathbf{x}_j^t)$. The picture highlights that $\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$ actually "tracks" the contribution in the coupling constraint due to all the other agents in the network obtained by means of neighboring information only. Specifically only the most informative value for the coupling is estimated with this procedure, i.e., the maximum component of the vector $\sum_{j\neq i} \mathbf{g}_j(\mathbf{x}_j^t)$.

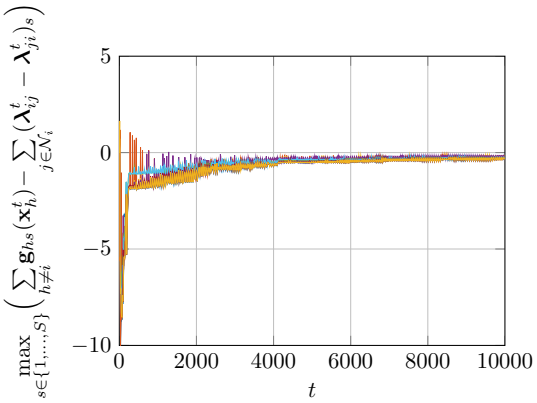


Figure 5.3: Evolution of the "constraint tracking" error of the coupling constraint for all $i \in \{1, \dots, N\}$.

Remark 5.5.1. *Since the value of* $\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$ *is tracking the missing part of the coupling constraint due to all the other agents, it is reasonable to use it as a local measure about the (global) feasibility of the current solution estimate. Roughly speaking, when* $\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$ *converges to a stationary value, agents can consider their current solution estimates feasible for the coupling constraint, and declare their optimization process completed.* □

Finally, in Figure 5.4 the convergence rate of the distributed algorithm is shown, i.e., the difference between the centralized optimal cost $\eta^\star = f^\star$ and the sum of the local costs $\sum_{i=1}^N (f_i(\mathbf{x}_i^t) + M\rho_i^t)$, in logarithmic scale. It can be seen that the proposed algorithm converges in a nonmonotonic fashion to the optimal cost with a sublinear rate $O(1/\sqrt{t})$ as expected for a subgradient method.

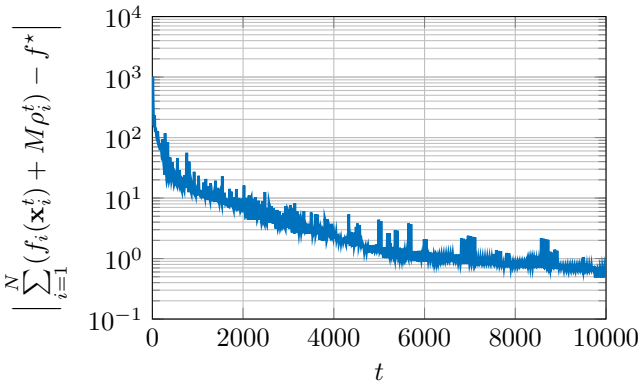


Figure 5.4: Evolution of the cost error $\left|\sum_{i=1}^{N}(f_i(\mathbf{x}_i^t)+M\rho_i^t)-f^\star\right|$ in logarithmic scale.

5.6 Peak-Minimization Problem

In this section we consider a constraint-coupled set-up arising in Demand Side Management (DSM) applications that have been described in Section 1.5.2.

The duality-based methodology described in the previous sections applies also to the peak minimization set-up and in the following we adapt the Relaxation and Successive Distributed Decomposition to such a scenario.

We consider a network of N processors that want to solve a peak-

minimization problem

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N,P} \; & P \\ \text{subj. to } & \mathbf{x}_i \in X_i, && i \in \{1,\dots,N\}, \\ & \sum_{i=1}^{N} g_{i,s}(x_{i,s}) \le P, && s \in \{1,\dots,S\}, \end{aligned} \tag{5.29}$$

where, for all $i \in \{1,\dots,N\}$, each constraint set $X_i \subseteq \mathbb{R}^S$ is nonempty, convex and compact, and the functions $g_{i,s} : \mathbb{R} \to \mathbb{R}$, $s \in \{1,\dots,S\}$, are convex.

5.6.1 Distributed Peak Minimization via Duality Tour

In this subsection we sketch how to derive a distributed algorithm for solving the peak-minimization problem (5.29) using the same methodology described in Section 5.3.

We start by writing the dual problem of (5.29). Let $\boldsymbol{\mu} = [\mu_1,\dots,\mu_S]^\top \in \mathbb{R}^S$, then the partial Lagrangian of (5.29) is given by

$$\begin{aligned} \mathcal{L}_1(\mathbf{x}_1,\dots,\mathbf{x}_N,P,\boldsymbol{\mu}) &= P + \sum_{s=1}^{S} \mu_s \Big(\sum_{i=1}^{N} g_{i,s}(x_{i,s}) - P \Big) \\ &= P\Big(1 - \sum_{s=1}^{S} \boldsymbol{\mu}_s\Big) + \sum_{i=1}^{N}\sum_{s=1}^{S} \mu_s g_{i,s}(x_{i,s}). \end{aligned}$$

The dual function $q : \mathbb{R}^S \to \mathbb{R}$ is defined as

$$q(\boldsymbol{\mu}) := \min_{\mathbf{x}_1 \in X_1,\dots,\mathbf{x}_N \in X_N, P} \mathcal{L}_1(\mathbf{x}_1,\dots,\mathbf{x}_N,P,\boldsymbol{\mu}),$$

where the presence of the local constraints $\mathbf{x}_i \in X_i$ for all $i \in \{1,\dots,N\}$ is due to the fact that we did not dualize them. The minimization of $\mathcal{L}_1$ with respect to P gives rise to the simplex constraint $\{\boldsymbol{\mu} \in \mathbb{R}^S \mid \mathbf{1}^\top \boldsymbol{\mu} = 1\}$ on the dual problem, while the minimization with respect to $\mathbf{x}_i$, $i \in \{1,\dots,N\}$, splits over i. Thus, the dual problem can be written as

$$\begin{aligned} \max_{\boldsymbol{\mu} \in \mathbb{R}^S} \; & \sum_{i=1}^{N} q_i(\boldsymbol{\mu}) \\ \text{subj. to } & \mathbf{1}^\top \boldsymbol{\mu} = 1,\ \boldsymbol{\mu} \succeq \mathbf{0}, \end{aligned} \tag{5.30}$$

with $\mathbf{1} := [1,\dots,1]^\top \in \mathbb{R}^S$, $\mathbf{0} := [0,\dots,0]^\top \in \mathbb{R}^S$ and

$$q_i(\boldsymbol{\mu}) = \min_{\mathbf{x}_i \in X_i} \sum_{s=1}^{S} \mu_s g_{i,s}(x_{i,s}), \quad i \in \{1,\dots,N\}.$$

Differently from problem (5.4), problem (5.30) has a compact domain $\{\boldsymbol{\mu} \in \mathbb{R}^S \mid \mathbf{1}^\top \boldsymbol{\mu} = 1,\, \boldsymbol{\mu} \succeq \mathbf{0}\}$. This, is a key feature which simplifies the application of the duality tour methodology. In fact, by introducing copies of $\boldsymbol{\mu}$ and the consistency constraints among them, we can write

$$\begin{aligned} \max_{\boldsymbol{\mu} \in \mathbb{R}^S} \; & \sum_{i=1}^{N} q_i(\boldsymbol{\mu}_i) \\ \text{subj. to } & \mathbf{1}^\top \boldsymbol{\mu}_i = 1,\, \boldsymbol{\mu}_i \succeq \mathbf{0}, \qquad i \in \{1, \dots, N\}, \\ & \boldsymbol{\mu}_i = \boldsymbol{\mu}_j \qquad\qquad (i,j) \in \mathcal{E}, \end{aligned} \tag{5.31}$$

where we have also added redundant local simplex constraints on each copy. Finally, to complete the tour we write the dual of problem (5.31) which is given by

$$\min_{\boldsymbol{\Lambda} \in \mathbb{R}^{S \cdot |\mathcal{E}|}} \; \sum_{i=1}^{N} \eta_i\big(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}\big). \tag{5.32}$$

where each $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^S$ is the Lagrange multiplier associated to the constraint $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j$ and

$$\eta_i = \max_{\mathbf{1}^\top \boldsymbol{\mu}_i = 1, \boldsymbol{\mu}_i \succeq \mathbf{0}} \Big(q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\top \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) \Big),$$

for all $i \in \{1, \dots, N\}$.

Retracing the same line of proof carried out in the previous sections, a subgradient method applied to problem (5.32) turns out to be a distributed optimization algorithm. Informally, the algorithm consists of a two-step iterative procedure. First, each node $i \in \{1, \dots, N\}$ stores a set of variables $((\mathbf{x}_i, \rho_i), \boldsymbol{\mu}_i)$ obtained as a primal-dual optimal solution pair of a local optimization problem with an epigraph structure as the centralized problem (5.29). The coupling with the other nodes in the original formulation is replaced by a term depending on neighboring variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$. These variables are updated in the second step according to a suitable linear law weighting the difference of neighboring $\boldsymbol{\mu}_i$. Nodes use a diminishing step-size denoted by γ^t and can initialize the variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$, to arbitrary values. In the Distributed Algorithm 11 we formally state the Distributed Duality-Based Peak Minimization algorithm from the perspective of node i.

Distributed Algorithm 11 DDPM

Processor states: $(\mathbf{x}_i, \rho_i)$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\lambda}_{ij}$ for $j \in \mathcal{N}_i$

Evolution:

Gather $\boldsymbol{\lambda}_{ji}^t$ from $j \in \mathcal{N}_i$

Compute $\big((\mathbf{x}_i^{t+1}, \rho_i^{t+1}), \boldsymbol{\mu}_i^{t+1}\big)$ as a primal-dual optimal solution pair of

$$\begin{aligned} &\min_{\mathbf{x}_i, \rho_i} \ \rho_i \\ &\text{subj. to } \mathbf{x}_i \in X_i \\ &\qquad g_{i,s}(x_{i,s}) + \sum_{j \in \mathcal{N}_i} \big(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t\big)_s \leq \rho_i, \quad s \in \{1, \dots, S\} \end{aligned} \tag{5.33}$$

Gather $\boldsymbol{\mu}_j^{t+1}$ from $j \in \mathcal{N}_i$

Update for all $j \in \mathcal{N}_i$

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t - \gamma^t \big(\boldsymbol{\mu}_i^{t+1} - \boldsymbol{\mu}_j^{t+1}\big) \tag{5.34}$$

We are now ready to state a converge result, namely that DDPM generates sequences that asymptotically converge to primal feasible optimal solutions of problem (5.29).

Theorem 5.6.1. *Let $\{(\mathbf{x}_i^t, \rho_i^t)\}_{t \geq 0}$, $i \in \{1, \dots, N\}$, be a sequence generated by the DDPM distributed algorithm, with a diminishing step-size γ^t satisfying Assumption 5.2.3. Then, the following holds:*

1. *the sequence $\big\{\sum_{i=1}^N \rho_i^t\big\}_{t \geq 0}$ converges to the optimal cost $P^\star$ of problem (5.29), and*
2. *every limit point of the primal sequence $\{\mathbf{x}_i^t\}_{t \geq 0}$, with $i \in \{1, \dots, N\}$, is an optimal (feasible) solution of (5.29).* □

A detailed proof of Theorem 5.6.1 can be found in [73].

5.6.2 Application to Thermostatically Controlled Loads

In this section we propose a numerical example in which we apply the proposed method to a network of Thermostatically Controlled Loads (TCLs) (such as air conditioners, heat pumps, electric water heaters), [4]. The discrete-time dynamical model of the i-th device is given by

$$T_{i,s+1} = T_{i,s}\, e^{-\alpha \Delta\tau} + \Big(1 - e^{-\alpha \Delta\tau}\Big)\Big(\frac{Q}{\alpha} x_{i,s} + \frac{\delta_{i,s}}{\alpha} + T_{i,s}^{out}\Big), \tag{5.35}$$

where $T_{i,s}$ is the temperature, $\alpha > 0$ is a parameter depending on geometric and thermal characteristics, $\Delta\tau \geq 0$ models the sampling interval, $T_{i,s}^{out}$ is

the air temperature outside the device, δ_{is} represents a known time-varying forcing term on the internal temperature of the device, $x_{i,s} \in [0,1]$ is the control input, and $Q > 0$ is a scaling factor. Moreover, we constrain the temperature to stay within a given interval $[T_{min}, T_{max}]$ with $T_{max} > T_{min} \geq 0$. The constraints due to the dynamics and the bounds on the temperature can be written compactly as inequality constraints on the input in the form $\mathbf{A}_i \mathbf{x}_i \preceq \mathbf{b}_i$, for each agent $i \in \{1, \dots, N\}$, for some $\mathbf{A}_i$ and $\mathbf{b}_i$ depending on (5.35). Finally, we assume that the power consumption $g_{i,s}(x_{i,s})$ of the i-th device is directly proportional to $x_{i,s}$, i.e., $g_{i,s}(x_{i,s}) = c_i x_{i,s}$.

Thus, optimization problem (5.29) for this scenario is

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N,P} \; & P \\ \text{subj. to } & \mathbf{A}_i \mathbf{x}_i \preceq \mathbf{b}_i,\ \mathbf{x}_i \in [0,1]^S, \quad i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} c_i x_{i,s} \leq P, \qquad\qquad s \in \{1,\dots,S\}. \end{aligned} \tag{5.36}$$

where $\mathbf{A}_i$ and $\mathbf{b}_i$ encode the constraints due to the discrete-time dynamics, the temperature constraint $T_{is} \in [T_{min}, T_{max}]$ and the known forcing term δ_{is}. Notice that the local constraint set is $X_i := \left\{\mathbf{x}_i \in \mathbb{R}^S \mid \mathbf{A}_i \mathbf{x}_i \preceq \mathbf{b}_i \text{ and } \mathbf{x}_i \in [0,1]^S\right\}$.

We choose each δ_{is} to be constant for an interval of 5 slots and zero otherwise. The nonzero values are set in the central part of the horizon $\{1, \dots, S\}$ by randomly shifting the center. Then, we randomly choose the heterogeneous power consumption coefficient $c_i \in \mathbb{R}$ of each device from a set of five values, drawn from a uniform distribution in $[1, 3]$. Finally, we consider $N = 20$ agents communicating according to an undirected connected Erdős-Rényi random graph $\mathcal{G}$ with parameter 0.2. We consider a horizon of $S = 60$. Finally, we used a diminishing step-size in the form $\gamma^t = t^{-0.8}$.

In Figure 5.5 we show the evolution at each algorithm iteration t of the local objective functions ρ_i^t, $i \in \{1, \dots, N\}$, (solid lines) which converge to stationary values. Moreover, we also plot their sum $\sum_{i=1}^N \rho_i^t$ (dashed line) and the value $P^t = \max_{s \in \{1,\dots,S\}} \sum_{i=1}^N g_{i,s}(x_{i,s}^t)$ (dotted line). As proven in [73, Corollary 3.6] both of them asymptotically converge to the centralized optimal cost $P^\star$ of problem (5.36).

In Figure 5.6 (left) the local solutions in the last algorithm iteration are depicted. We denote them by $\mathbf{x}_i^\star$, $i \in \{1, \dots, N\}$, to highlight that they satisfy the cost optimality up to the required tolerance 10^{-3}. We also plot the resulting aggregate optimal consumption, i.e., $\sum_{i=1}^N c_i \mathbf{x}_i^\star$, which, as expected, in fact shaves off the power demand peak. Moreover, the optimal local solutions satisfy the box constraint $[0, 1]$ for each slot $s \in \{1, \dots, S\}$. In fact, as we have proven, the algorithm converges in an interior point fashion, i.e., the local constraint at each node $i \in \{1, \dots, N\}$, is satisfied at all the

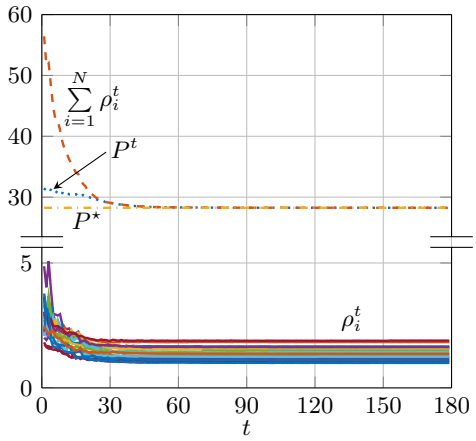


Figure 5.5: Evolution of ρ_i^t, $i \in \{1, \dots, N\}$, (solid lines), their sum $\sum_{i=1}^{N} \rho_i^t$ (dashed line), P^t (dotted line) and (centralized) optimal cost $P^\star$ (dash-dotted line).

algorithm iterations and in Figure 5.6 (right) we depict, as an example, the behavior of the components of $\mathbf{x}_1^t$.

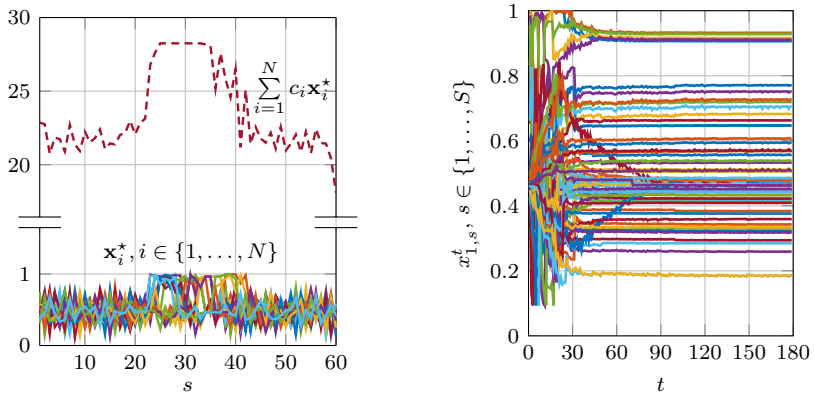


Figure 5.6: Left: profile of optimal solutions $\mathbf{x}_1^\star, \dots, \mathbf{x}_N^\star$ (solid lines), and cost $\sum_{i=1}^{N} c_i x_{i,s}^\star$ (dashed line) over the horizon $\{1, \dots, S\}$. Right: algorithmic evolution of the estimate $\mathbf{x}_1^t$.

In Figure 5.7 (left) we show the violation of the coupling constraints, for all $s \in \{1, \dots, S\}$ at each iteration t. As expected, the violations asymptotically go to nonnegative values, consistently with the claimed primal recovery feature enjoyed by our distributed algorithm, proven in Theorem 5.2.4. In Figure 5.7 (right) the difference $\sum_{i=1}^{N} (g_{i,s}(x_{i,s}^t) - \rho_i^t)$ is also shown, which is always nonnegative as theoretically expected, see the proof of Theorem 5.6.1 in [73] for further details.

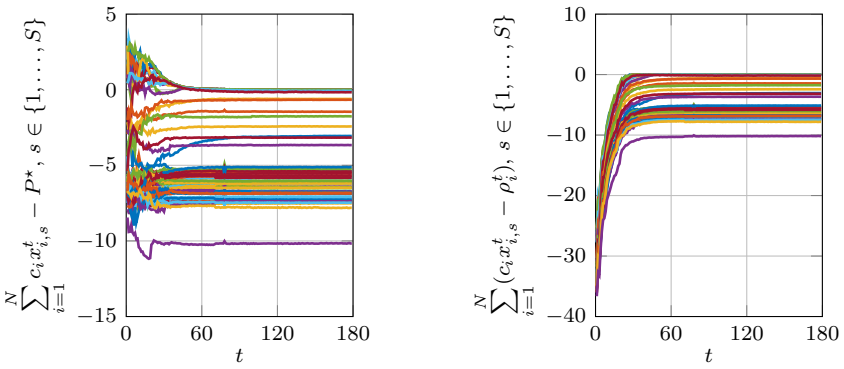


Figure 5.7: Evolution of primal violations of solutions $\mathbf{x}_i^t$, $i \in \{1,\dots,N\}$.

Finally, we propose a numerical comparison between our DDPM distributed algorithm and a plain distributed subgradient method, see [64], applied to the dual of the peak-minimization problem (5.29). In particular, agents agree on a solution of (5.30) and build the averaging primal sequence to recover a primal feasible solution. Formally, at every iteration t, each agent i locally computes a dual subgradient of its own q_i at its current dual solution estimate $\boldsymbol{\mu}_{i,\text{SUB}}^t$ as

$$\widetilde{\nabla} q_i(\boldsymbol{\mu}_{i,\text{SUB}}^t) = [g_{i,1}(\mathbf{x}_{i,1,\text{SUB}}^t) \cdots g_{i,S}(\mathbf{x}_{i,S,\text{SUB}}^t)]^\top - \rho_{i,\text{SUB}}^t \mathbf{1},$$

where

$$\mathbf{x}_{i,\text{SUB}}^t \in \operatorname*{argmin}_{\mathbf{x}_i \in X_i} \sum_{s=1}^{S} \boldsymbol{\mu}_{i,s,\text{SUB}}^t g_{i,s}(\mathbf{x}_{i,s})$$

and

$$\rho_{i,\text{SUB}}^t = \max_{s \in \{1,\dots,S\}} g_{i,s}(\mathbf{x}_{i,s,\text{SUB}}^t),$$

Then, it receives $\boldsymbol{\mu}_{j,\text{SUB}}^t$ from its neighbors $\mathcal{N}_i$ and performs the distributed subgradient iteration given by

$$\boldsymbol{\mu}_{i,\text{SUB}}^{t+1} = \mathcal{P}_{\text{SIMPLEX}}\left[\sum_{j \in \mathcal{N}_i} w_j^i \, \boldsymbol{\mu}_{j,\text{SUB}}^t + \gamma^t \widetilde{\nabla} q_i(\boldsymbol{\mu}_{i,\text{SUB}}^t) \right],$$

where $\mathcal{P}_{\text{SIMPLEX}}$ denotes the Euclidean projection onto the simplex $\{\boldsymbol{\mu} \succeq \mathbf{0} \mid \mathbf{1}^\top \boldsymbol{\mu} = 1\}$ and w_j^i are entries of a doubly stochastic matrix matching the communication graph. The primal sequences computed through the running average are $\widehat{\mathbf{x}}_i^t = \frac{1}{t} \sum_{\tau=0}^{t} \mathbf{x}_{i,\text{SUB}}^\tau$, $i \in \{1,\dots,N\}$.

We randomly generate 50 problem instances and run both algorithms to evaluate the cost errors $|P^\star - \sum_{i=1}^N \max_{s \in \{1,\dots,S\}} g_{i,s}(\mathbf{x}_{i,s}^t)|$, for the DDPM, and $|P^\star - \sum_{i=1}^N \max_{s \in \{1,\dots,S\}} g_{i,s}(\widehat{\mathbf{x}}_{i,s,\text{SUB}}^t)|$, for the dual distributed subgradient.

As shown in Figure 5.8, our DDPM algorithm is faster than the plain distributed subgradient. Moreover, DDPM appears to reach finite time convergence for this problem set-up (in the 92% of the cases within 2000 iterations, while in all the other cases in no more than 9000). On the other hand, the plain distributed subgradient after 10000 iterations is still at an accuracy of about 10^{-1}.

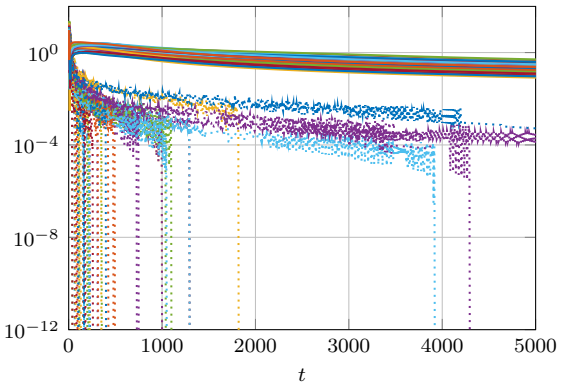


Figure 5.8: Evolution of the cost error for the DDPM (dotted lines) and for the plain distributed subgradient method (solid lines).

Conclusions & Future Developments

Conclusions

In this thesis we investigated optimization set-ups arising in control, estimation and learning applications in cyber-physical network applications. In CPNs the communication network cannot be considered a design parameter, thus algorithms need to take into account more complex schemes that can handle, e.g., asynchronous updates. Also the problems can involve huge decision variables leading immediately to "big-data" optimization problems which may be nonconvex. Finally, we focused on a key feature of distributed optimization algorithms when applied to dynamical networked systems where optimization problems with coupling among agents need to be repeatedly solved. We proposed tailored distributed optimization algorithms that tackled the mentioned challenges.

Specifically, we designed and proved the convergence of asynchronous distributed algorithms for cost-coupled optimization problems. In particular, we extended the classical distributed dual decomposition algorithm to an asynchronous version. Exploiting duality, proximal operators and block-coordinate methods, we were able to design a flexible algorithmic framework that handle composite costs and local constraints using local and constant step-sizes. Then, we have investigated the so-called "big-data" optimization problems where the size of the decision variable is typically huge. We considered a structured optimization set-up enjoying a partitioned property, in particular costs and constraints depend only on a small number of components of the entire decision vector. We proposed a dual and a primal algorithm to solve such partitioned big-data problems in asynchronous networks. The analysis is based on block-coordinate proximal methods. The dual approach allowed us to handle strongly convex programs with partitioned constraints, while with the primal approach we managed nonconvex costs with constraints depending only on a single variable component. Then, we considered a more general big-data set-up and proposed a distributed algorithm that extends the state-of-the-art distributed algorithms based on gradient

tracking schemes to a big-data scenario. By properly leveraging a block-wise optimization and communication scheme, we designed a distributed big-data optimization algorithm that can solve nonconvex cost-coupled optimization problems with huge size. Finally, we investigated a set-up that arises in dynamic optimization problems that are strictly related to important control applications. We proposed a novel distributed algorithm based on a relaxation of the original problem and an elegant exploration of Lagrangian duality theory that represents a first methodological step toward more complex schemes.

Future Developments

Distributed optimization poses numerous scientific challenges and in this thesis only a small part of them has been addressed. A natural development of this work may involve the extension of proposed techniques (or the combination of different ones) to simultaneously address more than one challenge and/or apply them to different set-ups. We briefly discuss some outlooks on possible future developments that can be carried out starting from this thesis.

As for the big-data optimization scenario, in Chapter 3 we studied convex programs with local constraints and exploited duality to design effective distributed algorithms. On the other hand, in Chapter 4 we investigated a primal approach to solve nonconvex problems subject to a common, convex constraint while no local constraints have been taken into account. A relevant step forward with respect to the existing literature would be to investigate primal methods, that possibly makes use of tracking mechansims, to solve nonconvex problems with local convex and nonconvex constraints.

As for asynchronous algorithms, we proposed (block-coordinate) methods for cost-coupled problems. On the other hand asynchrony is a highly desirable feature also for other set-ups as, e.g., the constraint-coupled set-up. The schemes proposed in Chapter 5 are efficient distributed methods exhibiting appealing properties such as the constraint violation estimate that decays at a given rate. However, they are designed for the undirected, static communication network. A possible extension consists in investigating how the exploration of alternative decomposition tools may lead to more general distributed subgradient algorithms that work under time-varying, asynchronous and directed networks.

Another important aspect that deserves further investigation concerns the feasibility guarantees that the proposed methodology can offer. This is a mandatory requirement for the relevant applications associated to the set-up under investigation. A major contribution of our methods is the provably asymptotic primal feasibility of the local decision variables without any (very slow) averaging mechanisms. This algorithmic feature represents a building

block to formally characterize the magnitude of the infeasibility. Thanks to this property, our method can be embedded in a distributed model predictive control scheme where a "supervised violation" of the coupling constraints can be a key feature for the controller design.

Appendix A

Optimization Basics

A.1 Lagrangian Duality

Consider a constrained optimization problem, addressed as primal problem, having the form

$$\begin{aligned} &\min_{\mathbf{x}\in X} \; f(\mathbf{x}) \\ &\text{subj. to } \; \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}, \end{aligned} \tag{A.1}$$

where $X \subseteq \mathbb{R}^d$ is a convex and compact set, $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function and $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^S$ is such that each component $\mathbf{g}_s : \mathbb{R}^d \to \mathbb{R}$, $s \in \{1, \ldots, S\}$, is a convex (scalar) function.

The following optimization problem

$$\begin{aligned} &\max_{\boldsymbol{\mu}} \; q(\boldsymbol{\mu}) \\ &\text{subj. to } \; \boldsymbol{\mu} \succeq \mathbf{0} \end{aligned} \tag{A.2}$$

is called the dual of problem (A.1), where $q : \mathbb{R}^S \to \mathbb{R}$ is obtained by minimizing with respect to $\mathbf{x} \in X$ the Lagrangian function $\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\mu}^\top \mathbf{g}(\mathbf{x})$, i.e., $q(\boldsymbol{\mu}) = \min_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu})$. Problem (A.2) is well posed since the domain of q is convex and q is concave on its domain.

A vector $\bar{\boldsymbol{\mu}} \in \mathbb{R}^S$ is said to be a Lagrange multiplier if $\bar{\boldsymbol{\mu}} \succeq \mathbf{0}$ and

$$\inf_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \bar{\boldsymbol{\mu}}) = \inf_{\mathbf{x}\in X \,:\, \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}} f(\mathbf{x}).$$

It can be shown that the following inequality holds [9]

$$\inf_{\mathbf{x}\in X} \sup_{\boldsymbol{\mu}\succeq\mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) \geq \sup_{\boldsymbol{\mu}\succeq\mathbf{0}} \inf_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}), \tag{A.3}$$

which is called weak duality. When in (A.3) the equality is verified, then we say that strong duality holds and, thus, solving the primal problem (A.1) is equivalent to solving its dual formulation (A.2). In this case the right-hand-side problem in (A.3) is referred to as *saddle-point problem* of (A.1).

Definition A.1.1. *A pair* $(\mathbf{x}^\star, \boldsymbol{\mu}^\star)$ *is called a primal-dual optimal solution of problem* (A.1) *if* $\mathbf{x}^\star \in X$ *and* $\boldsymbol{\mu}^\star \succeq \mathbf{0}$*, and* $(\mathbf{x}^\star, \boldsymbol{\mu}^\star)$ *is a saddle point of the Lagrangian, i.e.,*

$$\mathcal{L}(\mathbf{x}^\star, \boldsymbol{\mu}) \leq \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\mu}^\star) \leq \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}^\star)$$

for all $\mathbf{x} \in X$ *and* $\boldsymbol{\mu} \succeq \mathbf{0}$. □

A more general min-max property can be stated. Let $X \subseteq \mathbb{R}^d$ and $W \subseteq \mathbb{R}^S$ be nonempty convex sets. Let $\phi : X \times W \to \mathbb{R}$, then the following inequality holds true

$$\inf_{\mathbf{x}\in X} \sup_{\boldsymbol{\mu}\in W} \phi(\mathbf{x}, \boldsymbol{\mu}) \geq \sup_{\boldsymbol{\mu}\in W} \inf_{\mathbf{x}\in X} \phi(\mathbf{x}, \boldsymbol{\mu})$$

and is called the *max-min* inequality. When the equality holds, then we say that ϕ, Z and W satisfy the *strong max-min* property or the *saddle-point* property.

The following theorem gives a sufficient condition for the strong max-min property to hold.

Proposition A.1.2 ([10, Propositions 4.3])**.** *Let* ϕ *be such that (i)* $\phi(\cdot, \boldsymbol{\mu}) : X \to \mathbb{R}$ *is convex and closed for each* $\boldsymbol{\mu} \in W$*, and (ii)* $-\phi(\mathbf{x}, \cdot) : W \to \mathbb{R}$ *is convex and closed for each* $\mathbf{x} \in X$*. Assume further that* X *and* W *are convex compact sets. Then*

$$\sup_{\boldsymbol{\mu}\in W} \inf_{\mathbf{x}\in X} \phi(\mathbf{x}, \boldsymbol{\mu}) = \inf_{\mathbf{x}\in X} \sup_{\boldsymbol{\mu}\in W} \phi(\mathbf{x}, \boldsymbol{\mu})$$

and the set of saddle points is nonempty and compact. □

A.2 Subgradient Method

Consider the following constrained optimization problem

$$\min_{\mathbf{x}\in X} \; f(\mathbf{x}), \tag{A.4}$$

with $f : \mathbb{R}^d \to \mathbb{R}$ a convex function and $X \subseteq \mathbb{R}^d$ a closed, convex set.

A vector $\frac{\tilde{\partial}}{\partial \mathbf{x}} f(\mathbf{x}) \in \mathbb{R}^d$ is called a subgradient of the convex function f at $\mathbf{x} \in \mathbb{R}^d$ if $f(\mathbf{y}) \geq f(\mathbf{x}) + \frac{\tilde{\partial}}{\partial \mathbf{x}} f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$ for all $\mathbf{y} \in \mathbb{R}^d$. The (projected) subgradient method is the iterative algorithm given by

$$\mathbf{x}^{t+1} = \left[\mathbf{x}^t - \gamma^t \frac{\tilde{\partial}}{\partial \mathbf{x}} f(\mathbf{x}^t) \right]_X, \tag{A.5}$$

where $t \geq 0$ denotes the iteration counter, γ^t is the step-size, $\frac{\tilde{\partial}}{\partial \mathbf{x}} f(\mathbf{x}^t)$ denotes a subgradient of f at $\mathbf{x}^t$, and $[\,\cdot\,]_X$ is the Euclidean projection onto X.

The following proposition formally states the convergence of the subgradient method.

Proposition A.2.1 ([90]). *Assume that the subgradients of f are bounded at each $\mathbf{x} \in X$ and that the set of optimal solutions is nonempty. Let the step-size γ^t satisfy Assumption 5.2.3. Then the subgradient method in* (A.5) *applied to problem* (A.4) *converges in objective value to $f^\star$. Moreover, the sequence $\{\mathbf{x}^t\}_{t\geq 0}$ converges to an optimal solution of problem* (A.4). □

A.3 Penalty Method

Consider the inequality constrained problem (A.1) and the related penalized problem

$$\min_{\mathbf{x}\in X} \; f(\mathbf{x}) + c \cdot \max\{0, \mathbf{g}_1(\mathbf{x}), \ldots, \mathbf{g}_S(\mathbf{x})\}, \tag{A.6}$$

where $\mathbf{g}_s$ denotes the s-th component of $\mathbf{g}$ and c is a positive scalar.

Proposition A.3.1 ([8, Proposition 5.25]). *Assume that X is nonempty and convex, f and $\mathbf{g}_s$, $s \in \{1, \ldots, S\}$, are convex functions over X. Moreover, assume that problem* (A.1) *is feasible, has a finite cost and strong duality holds. Then,*

(a) in order for some optimal solution of (A.6) *to be an optimal solution of* (A.1) *there must exist a Lagrange multiplier $\bar{\boldsymbol{\mu}}$ of* (A.1) *for which*

$$\mathbf{y}^\top \bar{\boldsymbol{\mu}} < c \cdot \max\{0, \mathbf{y}_1, \ldots, \mathbf{y}_S\},$$

for all $\mathbf{y} \in \mathbb{R}^S$;

(b) in order for problem (A.1) *and* (A.6) *to have exactly the same optimal solutions, it is sufficient that*

$$\mathbf{y}^\top \bar{\boldsymbol{\mu}} < c \cdot \max\{0, \mathbf{y}_1, \ldots, \mathbf{y}_S\}, \tag{A.7}$$

for every $\mathbf{y} \in \mathbb{R}^S$ that has at least a component $\mathbf{y}_s > 0$ and for some Lagrange multiplier $\bar{\boldsymbol{\mu}}$ of (A.1). □

In the discussion after [8, Proposition 5.25], condition (A.7) is shown to hold when the penalty parameter c is greater than $\|\bar{\boldsymbol{\mu}}\|_1$.

Ringraziamenti Personali

Provo a riassumere in una pagina la mia gratitudine per le tante persone che mi hanno aiutato in questo percorso, ma per forza di cose, non potrò che essere riduttivo.

Comincio con Giuseppe a cui naturalmente devo gran parte della riuscita di questo lavoro sotto tanti aspetti: dall'aspetto scientifico (la tesi spero parli da sé), alla possibilità di frequentare numerose scuole di dottorato e conferenze internazionali, all'opportunità di coltivare collaborazioni scientifiche fino ad arrivare alle varie occasioni di interagire in prima persona con i suoi studenti. Non credo di essere stato all'altezza nella maggior parte delle circostanze, ma riconosco di essere stato decisamente e ripetutamente fortunato a poter vivere certe esperienze che mi hanno permesso di crescere sotto tutti i punti di vista.

Naturalmente non può mancare il variegato entourage che ha allietato le mie giornate lavorative e non. La nostra tavola quotidiana è stata (ed è) una cosa tutt'altro che scontata che mi ha più volte aiutato a perseverare in questo strano mestiere quando spesso non ho avuto ben chiare le ragioni per impegnarmi.

Vorrei ringraziare Aldo per avermi dato la possibilità di lavorare con lui e con il suo gruppo a Purdue, ma anche per la sua franchezza umana in tutti i nostri pranzi. Questa esperienza ha allargato la mia prospettiva, sia scientifica che umana, circa il lavoro del ricercatore. Su quest'ultimo aspetto naturalmente c'è una lunga lista di "aiutanti" che per fortuna ho incontrato strada facendo.

Ringrazio infine tutta la mia famiglia (i multiformi segni del loro contributo alla mia vita sono di dominio pubblico), Arianna e, con lei, il grande (ma grande) numero di amici che mi vogliono bene.

— Ivano

Bibliography

[1] M. Akbari, B. Gharesifard, and T. Linder, *Distributed subgradient-push online convex optimization on time-varying directed graphs*, IEEE 52nd Allerton Conference on Communication, Control, and Computing (Allerton), 2014.

[2] ———, *Distributed online convex optimization on time-varying directed graphs*, IEEE Transactions on Control of Network Systems (2015).

[3] M. Alizadeh, X. Li, Z. Wang, A. Scaglione, and R. Melton, *Demand-side management in the smart grid: Information processing for the power switch*, IEEE Signal Processing Magazine **29** (2012), no. 5, 55–67.

[4] M. Alizadeh, A. Scaglione, A. Applebaum, G. Kesidis, and K. Levitt, *Reduced-order load models for large populations of flexible appliances*, IEEE Transactions on Power Systems **30** (2015), no. 4, 1758–1774.

[5] H. H Bauschke and P. L Combettes, *Convex analysis and monotone operator theory in hilbert spaces*, Springer Science & Business Media, 2011.

[6] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM journal on imaging sciences **2** (2009), no. 1, 183–202.

[7] ———, *A fast dual proximal gradient algorithm for convex minimization and applications*, Operations Research Letters **42** (2014), no. 1, 1–6.

[8] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Academic Press, 1982.

[9] ———, *Nonlinear programming*, Athena scientific, 1999.

[10] ———, *Min common/max crossing duality: a geometric view of conjugacy in convex optimization*, Lab. for Information and Decision Systems, MIT, Tech. Rep. Report LIDS-P-2796 (2009).

[11] D. P. Bertsekas, A. Nedić, and A. Ozdaglar, *Convex analysis and optimization*, Athena Scientific, 2003.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Vol. 23, Prentice Hall Englewood Cliffs, NJ, 1989.

[13] P. Bianchi, W. Hachem, and F. Iutzeler, *A stochastic primal-dual algorithm for distributed asynchronous composite optimization*, Globalsip, 2014, pp. 732–736.

[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] G. Binetti, A. Davoudi, D. Naso, B. Turchiano, and F. L Lewis, *A distributed auction-based algorithm for the nonconvex economic dispatch problem*, IEEE Transactions on Industrial Informatics **10** (2014), no. 2, 1124–1132.

[16] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, *Convergence in multiagent coordination, consensus, and flocking*, 44th IEEE Conference on Decision and Control, and IEEE European Control Conference (CDC-ECC), 2005, pp. 2996–3000.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning **3** (2011), no. 1, 1–122.

[18] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[19] M. Bürger, G. Notarstefano, and F. Allgöwer, *From non-cooperative to cooperative distributed MPC: A simplicial approximation perspective*, European Control Conference (ECC), 2013, pp. 2795–2800.

[20] ———, *A polyhedral approximation framework for convex and robust distributed optimization*, IEEE Transactions on Automatic Control **59** (2014), no. 2, 384–395.

[21] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, *Asynchronous parallel algorithms for nonconvex big-data optimization: Model and convergence*, arXiv preprint arXiv:1607.04818 (2016).

[22] R. Carli and G. Notarstefano, *Distributed partition-based optimization via dual decomposition*, IEEE 52nd Conference on Decision and Control (CDC), 2013, pp. 2979–2984.

[23] R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, *Analysis of Newton-Raphson consensus for multi-agent convex optimization under asynchronous and lossy communications*, IEEE 54th Conference on Decision and Control (CDC), 2015, pp. 418–424.

[24] T.-H. Chang, M. Hong, and X. Wang, *Multi-agent distributed optimization via inexact consensus ADMM*, IEEE Transactions on Signal Processing **63** (2015), no. 2, 482–497.

[25] 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.

[26] N. Chatzipanagiotis and M. M Zavlanos, *On the convergence of a distributed augmented Lagrangian method for non-convex optimization*, IEEE Transactions on Automatic Control **62** (2017), no. 9, 4405–4420.

[27] A. I Chen and A. Ozdaglar, *A fast distributed proximal-gradient method*, IEEE 50th Allerton Conference on Communication, Control, and Computing (Allerton), 2012, pp. 601–608.

[28] P. D Christofides, R. Scattolini, D. M. de la Pena, and J. Liu, *Distributed model predictive control: A tutorial review and future research directions*, Computers & Chemical Engineering **51** (2013), 21–41.

[29] 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.

[30] D. V Dimarogonas, E. Frazzoli, and K. H Johansson, *Distributed event-triggered control for multi-agent systems*, IEEE Transactions on Automatic Control **57** (2012), no. 5, 1291–1297.

[31] 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.

[32] T. Erseghe, *A distributed and scalable processing method based upon ADMM*, IEEE Signal Processing Letters **19** (2012), no. 9, 563–566.

[33] F. Facchinei, G. Scutari, and S. Sagratella, *Parallel selective algorithms for nonconvex big data optimization*, IEEE Transactions on Signal Processing **63** (2015), no. 7, 1874–1889.

[34] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, *Dual decomposition for multi-agent distributed optimization with coupling constraints*, Automatica **84** (2017), 149–158.

[35] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, *An asynchronous mini-batch algorithm for regularized stochastic optimization*, IEEE Transactions on Automatic Control **61** (2016), no. 12, 3740–3754.

[36] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer, *Accelerated gradient methods and dual decomposition in distributed model predictive control*, Automatica **49** (2013), no. 3, 829–833.

[37] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms II: Advanced theory and bundle methods*, Vol. 306, Springer, 1993.

[38] J.-H. Hours and C. N Jones, *A parametric nonconvex decomposition algorithm for real-time and distributed NMPC*, IEEE Transactions on Automatic Control **61** (2016), no. 2, 287–302.

[39] B. Houska, J. Frasch, and M. Diehl, *An augmented Lagrangian based algorithm for distributed nonconvex optimization*, SIAM Journal on Optimization **26** (2016), no. 2, 1101–1127.

[40] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, *Explicit convergence rate of a distributed alternating direction method of multipliers*, IEEE Transactions on Automatic Control **61** (2016), no. 4, 892–904.

[41] D. Jakovetic, J. M. Freitas Xavier, and J. M. Moura, *Convergence rates of distributed Nesterov-like gradient methods on random networks*, IEEE Transactions on Signal Processing **62** (2014), no. 4, 868–882.

[42] G. M James, C. Paulson, and P. Rusmevichientong, *The constrained LASSO*, Citeseer, 2012.

[43] B. Johansson, M. Rabi, and M. Johansson, *A randomized incremental subgradient method for distributed optimization in networked systems*, SIAM Journal on Optimization **20** (2009), no. 3, 1157–1170.

[44] S. Kar and J. M. Moura, *Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs*, IEEE Journal of Selected Topics in Signal Processing **5** (2011), no. 4, 674–690.

[45] F. P Kelly, A. K Maulloo, and D. K. Tan, *Rate control for communication networks: shadow prices, proportional fairness and stability*, Journal of the Operational Research society **49** (1998), no. 3, 237–252.

[46] S. S Kia, J. Cortés, and S. Martínez, *Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication*, Automatica **55** (2015), 254–264.

[47] S. Lee and A. Nedić, *Distributed random projection algorithm for convex optimization*, IEEE Journal of Selected Topics in Signal Processing **7** (2013), no. 2, 221–229.

[48] ———, *Asynchronous gossip-based random projection algorithms over networks*, IEEE Transactions on Automatic Control **61** (2016), no. 4, 953–968.

[49] M. Lorenzen, M. Bürger, G. Notarstefano, and F. Allgöwer, *A distributed solution to the adjustable robust economic dispatch problem*, IFAC Proceedings Volumes **46** (2013), no. 27, 75–80.

[50] P. D. Lorenzo and G. Scutari, *Distributed nonconvex optimization over networks*, IEEE International Conference on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015, pp. 229–232.

[51] S. H Low and D. E Lapsley, *Optimization flow control, I: basic algorithm and convergence*, IEEE/ACM Transactions on Networking **7** (1999), no. 6, 861–874.

[52] K. Margellos, A. Falsone, S. Garatti, and M. Prandini, *Distributed constrained optimization and consensus in uncertain networks via proximal minimization*, IEEE Transactions on Automatic Control **PP** (2017), no. 99.

[53] D. Mateos-Núñez and J. Cortés, *Distributed saddle-point subgradient algorithms with laplacian averaging*, IEEE Transactions on Automatic Control **62** (2017), no. 6, 2720–2735.

[54] A.-H. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, *Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid*, IEEE Transactions on Smart Grid **1** (2010), no. 3, 320–331.

[55] A. Mokhtari, A. Koppel, and A. Ribeiro, *Doubly random parallel stochastic methods for large scale learning*, American Control Conference (ACC), 2016, pp. 4847–4852.

[56] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, *Distributed optimization with local domains: Applications in MPC and network flows*, IEEE Transactions on Automatic Control **60** (2015), no. 7, 2004–2009.

[57] I. Necoara, *Random coordinate descent algorithms for multi-agent convex optimization over networks*, IEEE Transactions on Automatic Control **58** (2013), no. 8, 2001–2012.

[58] I. Necoara and D. Clipici, *Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds*, SIAM Journal on Optimization **26** (2016), no. 1, 197–226.

[59] 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.

[60] A. Nedić and A. Olshevsky, *Distributed optimization over time-varying directed graphs*, IEEE Transactions on Automatic Control **60** (2015), no. 3, 601–615.

[61] A. Nedić, A. Olshevsky, and W. Shi, *A geometrically convergent method for distributed optimization over time-varying graphs*, IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 1023–1029.

[62] ———, *Achieving geometric convergence for distributed optimization over time-varying graphs*, SIAM Journal on Optimization **27** (2017), no. 4, 2597–2633.

[63] 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.

[64] A. Nedić and A. Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, IEEE Transactions on Automatic Control **54** (2009), no. 1, 48–61.

[65] A. Nedić, A. Ozdaglar, and P. A. Parrilo, *Constrained consensus and optimization in multi-agent networks*, IEEE Transactions on Automatic Control **55** (2010), no. 4, 922–938.

[66] Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal on Optimization **22** (2012), no. 2, 341–362.

[67] Y. Nesterov, *Gradient methods for minimizing composite functions*, Mathematical Programming **140** (2013), no. 1, 125–161.

[68] I. Notarnicola, R. Carli, and G. Notarstefano, *Distributed partitioned big-data optimization via asynchronous dual decomposition*, IEEE Transactions on Control of Network Systems **PP** (2017), no. 99, 1–10.

[69] I. Notarnicola and G. Notarstefano, *A randomized primal distributed algorithm for partitioned and big-data non-convex optimization*, IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 153–158.

[70] ———, *Constraint coupled distributed optimization: a relaxation and duality approach*, arXiv preprint arXiv:1711.09221 (2017).

[71] ———, *A duality-based approach for distributed optimization with coupling constraints*, IFAC World Congress, 2017, pp. 14891–14896.

[72] I. Notarnicola, M. Franceschelli, and G. Notarstefano, *A duality-based approach for distributed min-max optimization*, arXiv preprint arXiv:1611.09168 (2016).

[73] ———, *A duality-based approach for distributed min-max optimization with application to demand side management*, IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 1877–1882.

[74] I. Notarnicola and G. Notarstefano, *Randomized dual proximal gradient for large-scale distributed optimization*, IEEE 54th conference on decision and control (CDC), 2015, pp. 712–717.

[75] ———, *Asynchronous distributed optimization via randomized dual proximal gradient*, IEEE Transactions on Automatic Control **62** (2017), no. 5, 2095–2106.

[76] I. Notarnicola, Y. Sun, G. Scutari, and G. Notarstefano, *Distributed big-data optimization via block communications*, IEEE International Conference on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2017, pp. 557–561.

[77] ———, *Distributed big-data optimization via block-iterative convexification and averaging*, IEEE 56th Conference on Decision and Control (CDC), 2017, pp. 2281–2288.

[78] 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.

[79] B. O'Donoghue, G. Stathopoulos, and S. Boyd, *A splitting method for optimal control*, IEEE Transactions on Control Systems Technology **21** (2013), no. 6, 2432–2442.

[80] D. P. Palomar and M. Chiang, *A tutorial on decomposition methods for network utility maximization*, IEEE Journal on Selected Areas in Communications **24** (2006), no. 8, 1439–1451.

[81] N. Parikh and S. Boyd, *Proximal algorithms*, Foundations and Trends in Optimization **1** (2013), no. 3, 123–231.

[82] F. Pasqualetti, R. Carli, and F. Bullo, *Distributed estimation via iterative projections with application to power network monitoring*, Automatica **48** (2012), no. 5, 747–758.

[83] A. Patrascu and I. Necoara, *Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization*, Journal of Global Optimization **61** (2015), no. 1, 19–46.

[84] G. Qu and N. Li, *Harnessing smoothness to accelerate distributed optimization*, IEEE Transactions on Control of Network Systems **PP** (2017), no. 99, 1–1.

[85] 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.

[86] ———, *Parallel coordinate descent methods for big data optimization*, Mathematical Programming **156** (2016), no. 1-2, 433–484.

[87] I. D Schizas, A. Ribeiro, and G. B Giannakis, *Consensus in ad hoc wsns with noisy links–part I: Distributed estimation of deterministic signals*, IEEE Transactions on Signal Processing **56** (2008), no. 1, 350–364.

[88] F. C. Schweppe and J. Wildes, *Power system static-state estimation, part II: Approximate model.*, IEEE Transactions on Power Apparatus and Systems **89** (1970), no. 1, 125–130.

[89] 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.

[90] N. Z. Shor, *Minimization methods for non-differentiable functions*, Springer, 1985.

[91] 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.

[92] Y. Sun and G. Scutari, *Distributed nonconvex optimization for sparse representation*, IEEE International Conference on Speech and Signal Processing (ICASSP), 2017, pp. 4044–4048.

[93] Y. Sun, G. Scutari, and D. Palomar, *Distributed nonconvex multiagent optimization over time-varying networks*, IEEE 50th Asilomar Conference on Signals, Systems, and Computers, 2016, pp. 788–794.

[94] M. Todescato, G. Cavraro, R. Carli, and L. Schenato, *A robust block-Jacobi algorithm for quadratic programming under lossy communications*, IFAC-PapersOnLine, 2015, pp. 126–131.

[95] Q. Tran-Dinh, I. Necoara, and M. Diehl, *A dual decomposition algorithm for separable nonconvex optimization using the penalty function framework*, IEEE 52nd Conference on Decision and Control (CDC), 2013, pp. 2372–2377.

[96] ———, *Fast inexact decomposition algorithms for large-scale separable convex optimization*, Optimization **65** (2016), no. 2, 325–356.

[97] K. I Tsianos, S. Lawlor, and M. G Rabbat, *Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning*, IEEE 50th Allerton Conference on Communication, Control, and Computing (Allerton), 2012, pp. 1543–1550.

[98] J. Tsitsiklis, D. Bertsekas, and M. Athans, *Distributed asynchronous deterministic and stochastic gradient optimization algorithms*, IEEE Transactions on Automatic Control **31** (1986), no. 9, 803–812.

[99] E. Wei and A. Ozdaglar, *On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers*, IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2013, pp. 551–554.

[100] C. Xi, R. Xin, and U. A. Khan, *Add-opt: Accelerated distributed directed optimization*, IEEE Transactions on Automatic Control **PP** (2017), no. 99, 1–1.

[101] H. Xu, D. J. Eis, and P. J. Ramadge, *The generalized LASSO is reducible to a subspace constrained LASSO*, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013, pp. 3268–3272.

[102] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, *Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes*, IEEE 54th Conference on Decision and Control (CDC), 2015, pp. 2055–2060.

[103] B. Yang and M. Johansson, *Distributed optimization and games: A tutorial overview*, Networked control systems, 2010, pp. 109–148.

[104] R. Zamora and A. K. Srivastava, *Controls for microgrids with storage: Review, challenges, and research needs*, Renewable and Sustainable Energy Reviews **14** (2010), no. 7, 2009–2018.

[105] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, *Asynchronous Newton-Raphson consensus for distributed convex optimization*, 3rd IFAC workshop on distributed estimation and control in networked systems, 2012.

[106] 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.