Foundations and Trends® in Systems and Control

Distributed Optimization for Smart Cyber-Physical Networks

Suggested Citation: Giuseppe Notarstefano, Ivano Notarnicola and Andrea Camisa (2019), "Distributed Optimization for Smart Cyber-Physical Networks", Foundations and Trends® in Systems and Control: Vol. xx, No. xx, pp 1–16. DOI: 10.1561/XXXXXXXXX.

Giuseppe Notarstefano
Università di Bologna, Bologna, Italy
giuseppe.notarstefano@unibo.it

Ivano Notarnicola
Università di Bologna, Bologna, Italy
ivano.notarnicola@unibo.it

Andrea Camisa
Università di Bologna, Bologna, Italy
a.camisa@unibo.it

the essence of knowledge
Boston — Delft

Contents

Distributed Optimization for Smart Cyber-Physical Networks

Giuseppe Notarstefano[1], Ivano Notarnicola[2] and Andrea Camisa[3]

[1] *Università di Bologna, Bologna (Italy); giuseppe.notarstefano@unibo.it*
[2] *Università di Bologna, Bologna (Italy); ivano.notarnicola@unibo.it*
[3] *Università di Bologna, Bologna (Italy); a.camisa@unibo.it*

ABSTRACT

The presence of embedded electronics and communication capabilities as well as sensing and control in smart devices has given rise to the novel concept of cyber-physical networks, in which agents aim at cooperatively solving complex tasks by local computation and communication. Numerous estimation, learning, decision and control tasks in smart networks involve the solution of large-scale, structured optimization problems in which network agents have only a partial knowledge of the whole problem. Distributed optimization aims at designing local computation and communication rules for the network processors allowing them to cooperatively solve the global optimization problem without relying on any central unit. The purpose of this survey is to provide an introduction to distributed optimization methodologies. Principal approaches, namely (primal) consensus-based, duality-based and constraint exchange methods, are formalized. An analysis of the basic schemes is supplied, and state-of-the-art extensions are reviewed.

Giuseppe Notarstefano, Ivano Notarnicola and Andrea Camisa (2019), "Distributed Optimization for Smart Cyber-Physical Networks", Foundations and Trends® in Systems and Control: Vol. xx, No. xx, pp 1–16. DOI: 10.1561/XXXXXXXXX.

Introduction

Motivation

In recent years, the breakthroughs in embedded electronics are giving the opportunity to include computation and communication capabilities in almost any device of several domains as factories, farms, buildings, grids and cities. Communication among devices has enabled a number of new challenges along the direction of turning smart devices into smart (cooperating) systems. The keyword "cyber-physical networks" is being adopted to refer to this permeating reality, whose distinctive feature is that a great advantage can be obtained if its interconnected, complex nature is exploited. A novel peer-to-peer *distributed* computational framework is emerging as a new opportunity in which peer processors, communicating over a network, cooperatively solve a task without resorting to a unique provider that knows and owns all the data.

Several challenges arising in cyber-physical networks can be stated as optimization problems. Examples are estimation, decision, learning and control applications. To solve optimization problems over cyber-physical networks, it is not possible to apply the classical optimization algorithms (that we call *centralized*), which require the data to be managed by a single entity. In fact, the problem data are spread over the network, and it is undesirable (or even impossible) to collect them at a unique node. To this end, parallel computing serves as a source of inspiration. In order to speed up the solution of large-scale optimization problems, several effort has been made in designing *parallel* algorithms by splitting the computational burden among several processors. However, for typical parallel optimization algorithms, a central coordinating node is required and the communication topology is designed ad hoc. In distributed computation the communication topology cannot be thought of as a design parameter. Rather, it is a given part of the problem. Thus, in cyber-physical networks, the goal is to design algorithms, based on the exchange of information among the processors, that take advantage of the aggregated computational power. All the agents must be treated as peers and each of them must perform the same tasks and no "master" node must be present. Moreover, information privacy is often a requirement (i.e., private problem data at each node must not be shared with the other nodes). These challenges call for tailored strategies and have given rise to a novel, growing research branch termed *distributed optimization.*

Scope of the Monograph

The purpose of this survey is to give a comprehensive overview of the most common approaches used to design distributed optimization algorithms, together with the theoretical analysis of the main schemes in their basic version. We identify and formalize classes of problem set-ups that arise in motivating application scenarios. For each set-up, in order to give the main tools for analysis, we review tailored distributed algorithms in simplified cases. Extensions and generalizations of the basic schemes are also discussed at the end of each chapter. The algorithms have been developed by combining mathematical tools from optimization theory (e.g., duality) and network control theory (e.g., average consensus). For some of the discussed algorithms, we will present also parallel algorithms that serve as a starting point for the development of distributed methods.

We focus on three main categories of distributed optimization approaches: *(i)* primal consensus-based methods, i.e., methods combining classical gradient or subgradient steps with local averaging schemes; *(ii)* dual methods, i.e., methods which employ the Lagrangian dual of suitable equivalent formulations of the target problem to obtain a distributed routine; *(iii)* constraint exchange methods, which are based on the exchange of (active) constraints among agents to compute a solution of the considered problem.

Survey papers on distributed optimization have been proposed in the literature. An early survey paper presenting a broad class of relevant optimization problems in control is [85]. It also discusses tailored, parallel and distributed optimization algorithms based on decomposition techniques and including also the distributed subgradient method. Recent surveys analyze thoroughly average consensus [87] and the distributed subgradient method [87, 88, 91], with a literature review on other distributed optimization techniques. The book [97] provides parallel and distributed asynchronous optimization algorithms, including gradient tracking techniques. Some latest advances in distributed optimization are collected in [45].

Organization

In Chapter 1, we introduce the relevant problem set-ups, that we call *cost-coupled*, *constraint-coupled* and *common cost*, along with several motivating applications of interest arising in estimation, learning, decision and control. In Chapter 2 we provide an overview of primal approaches to solve cost-coupled problems, namely the distributed subgradient algorithm and the gradient tracking algorithm. In Chapter 3, a discussion on relevant duality forms for distributed optimization is first provided, and then distributed algorithms relying on Lagrangian approaches are reviewed. Namely, for cost-coupled problems, distributed dual decomposition and distributed ADMM algorithms are considered, while for constraint-coupled problems, a distributed dual subgradient algorithm and a method based on relaxation and successive distributed decomposition are presented. In Chapter 4, we focus on constraint exchange methods. We introduce the Constraints Consensus algorithm applied to common-cost problems, along with its most relevant extensions.

We also provide illustrative numerical examples to highlight significant properties of the considered distributed optimization methods. Since the described algorithms are designed for different problem set-ups, different, relevant simulation scenarios are considered in each chapter.

1

Distributed Optimization Framework

In this chapter we introduce the conceptual framework for distributed optimization in peer-to-peer networks. First, we describe the network model we will consider throughout the survey. Then we present and motivate the main optimization set-ups that are of interest in smart networks.

In a distributed scenario, we consider N units, called *agents* or *processors*, that have both communication and computation capabilities. Communication among agents is modeled by means of graph theory. Informally, given a graph $\mathcal{G}$ with N nodes, one for each agent, an agent i can send (receive) data to (from) another agent j, when the graph $\mathcal{G}$ contains an edge connecting i to j (j to i). In a distributed algorithm, agents initialize their local states and then start an iterative procedure in which communication and computation steps are iteratively performed, with all the nodes performing the same actions. In particular, local states are updated by using only information received by in-neighbors.

In this survey we consider a distributed framework in which agents cooperatively solve an optimization problem. The basic assumption we make is that each agent i has only a partial knowledge of the entire problem, e.g., only a portion of the cost and/or a portion of the constraints is locally available. In the rest of the chapter, depending on the specific optimization set-up, we will clarify what do we mean by cooperation among agents for the solution of a given optimization problem.

Remark 1.1. We point out that, regardless of the optimization problem structure, our standing assumption is that the distributed framework is made by cooperative agents. There is another strain of research on non-cooperative set-ups with applications to game theoretic problems. A non-exhaustive list of early references is [126, 66, 154]. □

1.1 Distributed Computation Model

In this section we formally define the communication model for a distributed algorithm. A network is modeled as a (possibly time-dependent) directed graph $\mathcal{G}^t = (\{1,\dots,N\}, \mathcal{E}^t)$, where $t \in \mathbb{N}$ is a universal (slotted) time, $\{1,\dots,N\}$ is the (fixed) set of agent identifiers and $\mathcal{E}^t \subseteq \{1,\dots,N\} \times \{1,\dots,N\}$, for all $t \geq 0$, is the (time-dependent) set of (directed) edges over the vertices $\{1,\dots,N\}$, which represents the communication links. A graphical representation of a time-varying network is given in Figure 1.1.

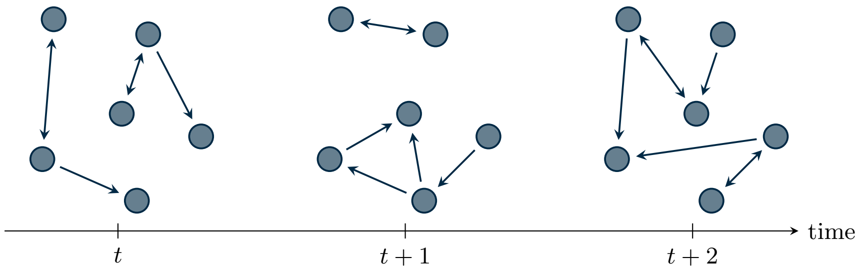


Figure 1.1: A directed time-varying graph of $N = 6$ nodes.

At each (universal) time instant t, a communication structure, i.e., a graph $\mathcal{G}^t$, is 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 of nodes 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.

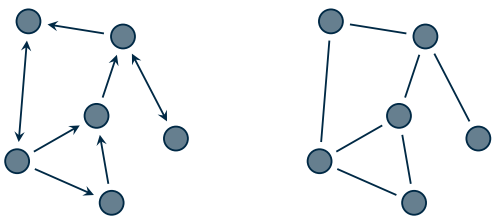


Figure 1.2: A directed (left) and an undirected (right) graph of $N = 6$ nodes.

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

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

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

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

- *jointly strongly connected* if the graph $\mathcal{G}^t_\infty \triangleq (\{1, \dots, N\}, \mathcal{E}^t_\infty)$, with $\mathcal{E}^t_\infty = \bigcup_{\tau=t}^{\infty} \mathcal{E}^\tau$, is strongly connected for all $t \geq 0$.
- *T-strongly connected* (or *uniformly jointly strongly connected*) if there exists a scalar $T > 0$ such that the graph $\mathcal{G}^t_T \triangleq (\{1, \dots, N\}, \mathcal{E}^t_T)$ 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 run distributed algorithms according to several communication protocols. When the steps of the algorithm explicitly depend on the value of t, we say that the algorithm is *synchronous*, i.e., agents must be aware of the current value of t and, thus, their local operations must be synchronized to a global clock. We will also consider a communication protocol in which agents are not aware of any global time information, i.e., their updates do not depend on t, and we term these algorithms *asynchronous*. In fact, if a distributed algorithm is designed to run over a jointly strongly connected graph, and the local computation steps do not depend on t, then the algorithm can be also implemented in an asynchronous network.

1.2 Optimization Set-ups

In this section we describe three general optimization set-ups that comprise several estimation, learning, decision and control application scenarios in smart networks. A distributed optimization algorithm for such classes of problems consists of an iterative procedure based on the distributed computation model introduced in Section 1.1. The goal for the agents is to eventually obtain a solution of the investigated problem. In each considered optimization set-up, this goal translates to different statements that will be formally specified next.

For an optimization algorithm, the aim is to minimize a scalar objective function (or cost function), usually denoted as $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^d$ is the decision variable. We may need to restrict the minimizer of f in a given constraint set $X \subseteq \mathbb{R}^d$ (or feasible set). From now on, we use the symbol min to denote that we want to minimize $f(\mathbf{x})$ subject to the constraints, and we compactly write the overall optimization problem as

$$\begin{aligned} \min_{\mathbf{x}} \; & f(\mathbf{x}) \\ \text{subj. to } & \mathbf{x} \in X. \end{aligned}$$

The generic constraint set X can also be expressed by means of equalities or inequalities as, e.g., $h_j(\mathbf{x}) = 0$ for $j \in \{1, \dots, p\}$, or $g_k(\mathbf{x}) \leq 0$ for $k \in \{1, \dots, q\}$, for some functions h_j and g_k. The equality and inequality constraints are usually compactly denoted as $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ or $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$. Centralized methods to approach this problem can be found in [9, 5].

In the remainder of this section, we introduce three structured versions of the above general optimization problem.

1.2.1 Cost-Coupled Optimization

We start by introducing an optimization set-up in which the cost function is expressed as the sum of local contributions f_i and all of them depend on a common optimization variable $\mathbf{x}$. Formally, the set-up is

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

where $\mathbf{x} \in \mathbb{R}^d$ and $X \subseteq \mathbb{R}^d$. The global constraint set X is assumed to be common to each agent, while $f_i : \mathbb{R}^d \to \mathbb{R}$ is assumed to be known by agent i only, for all $i \in \{1, \dots, N\}$. Figure 1.3 provides a graphical representation of how problem information is spread over the network.

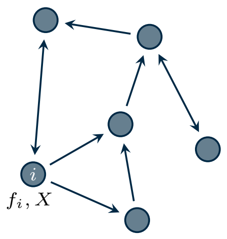


Figure 1.3: Cost-coupled set-up: each agent i only knows f_i and X and communicates according to the edges of the graph

.

More general versions of this optimization set-up assume that the constraint set is more structured, e.g., $X = \bigcap_{i=1}^{N} X_i$, where each X_i is known by agent i only.

Let $\mathbf{x}^\star$ denote an optimal solution of problem (1.1). For this optimization set-up, the goal is to design a distributed algorithm where each agent updates a local estimate $\mathbf{x}_i^t$ that converges (asymptotically or in finite time) to $\mathbf{x}^\star$, by means of local computation and neighboring communication only. An illustrative scheme is depicted in Figure 1.4.

Remark 1.2. An interesting optimization set-up arising in several applications is the so-called *partitioned*, or *partition-based*, set-up, first introduced in [39]. The problem is in the form (1.1), but the cost function and the constraints of each agent do not involve all the components

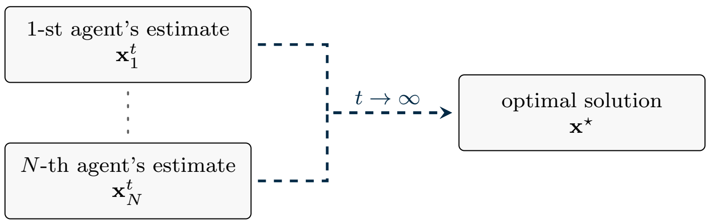

1-st agent's estimate $\mathbf{x}_1^t$
N-th agent's estimate $\mathbf{x}_N^t$
$t \to \infty$
optimal solution $\mathbf{x}^\star$

Figure 1.4: Illustrative scheme of the goal for the cost-coupled set-up and for the common cost set-up.

of the decision variable, but rather they depend only on some of its components. This sparsity in the problem can be modeled using a graph. Formally, the partitioned optimization set-up is

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

where $\mathbf{x}$ denotes the vector stacking $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$, while the notation $f_i(\mathbf{x}_i, \{\mathbf{x}_j\}_{j \in \mathcal{N}_i})$ highlights the fact that f_i actually depends only on the components of $\mathbf{x}$ indexed by $\{i\} \cup \mathcal{N}_i$. Distributed algorithms have been developed to solve partitioned problems. Remark 3.2 discusses how to tailor algorithms based on dual decomposition in order to take into account the partitioned structure. $\square$

1.2.2 Common Cost Optimization

Another important set-up arising in several applications is given by

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

where $f : \mathbb{R}^d \to \mathbb{R}$ is known by all the agents while each constraint $X_i \subseteq \mathbb{R}^d$ is known by agent i only, for all $i \in \{1, \ldots, N\}$. Figure 1.5 provides a graphical representation of how information is spread over the network.

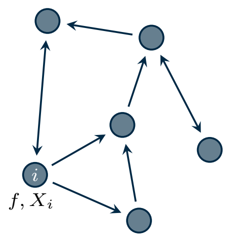


Figure 1.5: Common cost set-up: each agent i only knows f and X_i and communicates according to the edges of the graph.

The common cost set-up (1.2) is somehow similar to the cost-coupled set-up (1.1), since in both cases the optimization variable is shared among the processors. However, in the common cost set-up (1.2), the cost function is shared, and the coupling among the agents is due to the fact that the optimization variable must belong to all the local constraint sets. It is possible to think of problem (1.2) as a special case of the cost-coupled set-up (1.1) (with $X = \bigcap_{i=1}^{N} X_i$) by setting each $f_i(\mathbf{x}) = 1/N \cdot f(\mathbf{x})$. However, notice that a commonly known cost function explicitly allows for tailored distributed optimization algorithms such as, e.g., constraint exchange methods (cf. Chapter 4).

Let $\mathbf{x}^\star$ denote an optimal solution of problem (1.2). For such optimization set-up, the goal is to design a distributed algorithm where each agent updates a local estimate $\mathbf{x}_i^t$ that converges (asymptotically or in finite time) to $\mathbf{x}^\star$, by means of local computation and neighboring communication only (cf. Figure 1.4).

1.2.3 Constraint-Coupled Optimization

In this subsection, we present a different set-up which we call constraint-coupled. Agents in a network want to minimize the sum of local cost functions, each one depending only on a local vector satisfying local constraints. The decision vectors are then coupled to each other by means of separable coupling constraints. This feature leads easily to the so-called big-data problems having a very highly dimensional decision variable that grows with the network size. However, since agents are typically interested in computing only their (small) portion of an optimal solution, novel tailored methods need to be developed to address these challenges.

Formally, the constraint-coupled optimization problem is

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to} \quad & \mathbf{x}_i \in X_i, \qquad i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \le \mathbf{0}, \end{aligned} \tag{1.3}$$

where $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ is the global optimization vector stacking all the local variables, $X_i \subseteq \mathbb{R}^{d_i}$, $f_i : \mathbb{R}^{d_i} \to \mathbb{R}$ and $\mathbf{g}_i : \mathbb{R}^{d_i} \to \mathbb{R}^S$ are known by agent i only, for all $i \in \{1, \dots, N\}$. Notice that problem (1.3) is challenging because of the coupling constraints $\sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \le \mathbf{0}$. If there were no coupling constraints, the optimization would trivially split into N independent problems. Figure 1.6 provides a graphical representation of how information is spread over the network.

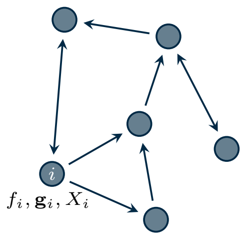


Figure 1.6: Constraint-coupled set-up: each agent i only knows f_i, X_i and $\mathbf{g}_i$ and communicates according to the edges of the graph.

Let $(\mathbf{x}_1^\star, \dots, \mathbf{x}_N^\star)$ denote an optimal solution of problem (1.3). The goal is to design a distributed algorithm where each agent updates a local estimate $\mathbf{x}_i^t$ that converges (asymptotically or in finite time) to $\mathbf{x}_i^\star$, the i-th portion of $(\mathbf{x}_1^\star, \dots, \mathbf{x}_N^\star)$, by means of local computation and neighboring communication only. An illustrative scheme is depicted in Figure 1.7.

A special instance of this set-up has been investigated in the context of resource allocation, where the coupling constraint is linear, e.g., $\sum_{i=1}^{N} \mathbf{x}_i = b$, and there are no local constraints. In this survey, we

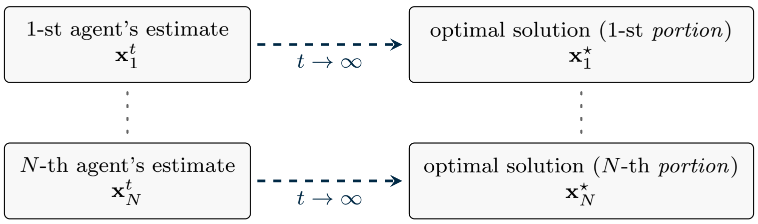

1-st agent's estimate $\mathbf{x}_1^t$
$t \to \infty$
optimal solution (1-st *portion*) $\mathbf{x}_1^\star$
N-th agent's estimate $\mathbf{x}_N^t$
$t \to \infty$
optimal solution (N-th *portion*) $\mathbf{x}_N^\star$

Figure 1.7: Illustrative scheme of the goal for the constraint-coupled set-up.

consider more general problems where the coupling may be nonlinear and local constraints are explicitly taken into account.

Remark 1.3 (Comparison with the cost-coupled set-up)**.** We notice that problem (1.1) can be cast as (1.3) by introducing copies $\mathbf{x}_1, \dots, \mathbf{x}_N$ of the decision vector $\mathbf{x}$ and appropriate coherence (coupling) constraints, i.e.,

$$\begin{aligned}
\min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\
\text{subj. to} \quad & \mathbf{x}_i \in X, \qquad i \in \{1,\dots,N\} \\
& \mathbf{x}_1 = \mathbf{x}_2 \\
& \quad \vdots \\
& \mathbf{x}_{N-1} = \mathbf{x}_N
\end{aligned}$$

However, it is worth noticing that the coupling constraint of such reformulation enjoys a special, sparse structure while the constraints in (1.3) are more general (since they involve all the agents in the network). □

1.3 Optimization Set-ups for Learning and Control

In this section, we motivate the study of the optimization set-ups introduced in Section (1.2) by describing important application scenarios that are of interest in control and robotics as well as communication and signal processing.

1.3.1 Regression for Data Analytics

Let us consider an important task for several applications, namely the linear *regression* problem, in which we assume that a set of points in a training dataset is used to estimate the parameters of a model (assumed to be linear in the parameters). The model can be exploited, e.g., to predict new generated samples. Figure 1.8 proposes a pictorial representation of a simple scenario in $\mathbb{R}^2$.

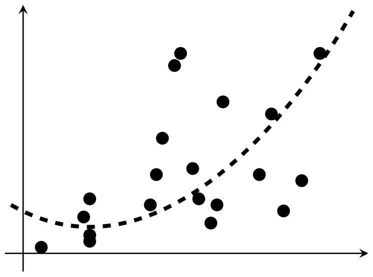


Figure 1.8: Set of data points in $\mathbb{R}^2$ that can be fit using a polynomial model (i.e., linear in the parameters). The coefficients of the polynomial are obtained with a regression approach.

Nowadays, especially in big-data contexts, a natural scenario is to assume that the training data are not (or cannot be) gathered at a main collection center. Rather, it is reasonable to assume that the samples are (spatially) distributed in a network, as shown in Figure 1.9.
Now, let us focus on Least Squares (LS), a popular regression approach. Assume that N processors in a network want to solve a regression problem, where $\mathbf{x} \in \mathbb{R}^d$ denotes the parameter vector that has to be estimated, and each agent i has n_i observations. The (unweighted) LS problem can be formulated as

$$\min_{\mathbf{x}} \sum_{i=1}^{N} \|\mathbf{D}_i \mathbf{x} - \mathbf{b}_i\|^2 \tag{1.4}$$

where, for all $i \in \{1, \ldots, N\}$, $\mathbf{D}_i \in \mathbb{R}^{n_i \times d}$ is the regression matrix and $\mathbf{b}_i \in \mathbb{R}^{n_i}$ is the label vector.

A typical challenge arising in regression problems is due to the fact that problem (1.4) may be ill-posed and can easily lead to over-fitting

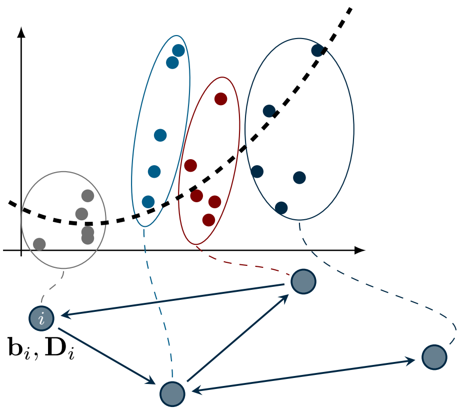


Figure 1.9: Regression problem over a network of 4 agents.

phenomena. A viable technique to prevent over-fitting consists in adding a suitable *regularization* term $r(\mathbf{x})$ in the cost function, leading to

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

where $r : \mathbb{R}^d \to \mathbb{R}$ is assumed to be known by all the agents in the network. Several possibilities for the regularizer $r(\mathbf{x})$ can be chosen. For instance, by using ℓ_1-norm, we obtain the so-called LASSO (Least Absolute Shrinkage and Selection Operator) problem, i.e.,

$$\min_{\mathbf{x}} \sum_{i=1}^{N} \|\mathbf{D}_i\mathbf{x} - \mathbf{b}_i\|^2 + \rho\|\mathbf{x}\|_1 \tag{1.5}$$

where ρ is a positive scalar used to strengthen or weaken the effects of the regularizer. Problem (1.5) can be classified as cost-coupled, i.e., of the form (1.1), with $X = \mathbb{R}^d$ and local functions given by $f_i(\mathbf{x}) = \|\mathbf{D}_i\mathbf{x} - \mathbf{b}_i\|^2 + \rho/N \cdot \|\mathbf{x}\|_1$.

This problem will be used to test duality-based methods for cost-coupled problems and a numerical example is shown in Section 3.6.

1.3.2 Classification via Logistic Regression

Regression problems can be also set up for a classification scenario. We recall a set-up in which linear models are trained by minimizing the so-called *logistic loss functions*. Suppose each agent has m_i points $p_{i,1}, \ldots, p_{i,m_i} \in \mathbb{R}^d$ (which represent training samples in a feature space) and suppose they are associated to binary labels, i.e., each point $p_{i,j}$ is labeled with $\ell_{i,j} \in \{-1, 1\}$, for all $j \in \{1, \ldots, m_i\}$ and $i \in \{1, \ldots, N\}$. The problem consists of building a linear classification model from the training samples by maximizing the a-posteriori probability of each class. In particular, we look for a separating hyperplane of the form $\{z \in \mathbb{R}^d \mid w^\top z + b = 0\}$, whose parameters ($w$ and b) can be determined by solving the convex optimization problem

$$\min_{w,b} \sum_{i=1}^{N} \sum_{j=1}^{m_i} \log\Big[1 + e^{-(w^\top p_{i,j} + b)\ell_{i,j}}\Big] + \frac{C}{2}\|w\|^2, \tag{1.6}$$

where $C > 0$ is a parameter affecting regularization. We now make some observations on problem (1.6). First, we see that it is an unconstrained optimization problem, so that an optimal solution can always be found (even though it may be meaningless for the classification problem). Second, we point out that the cost function is strictly convex, so that the optimal solution is unique. Finally, notice that the problem is cost-coupled, i.e., it is of the form (1.1), with $X = \mathbb{R}^d$ and each f_i is given by

$$f_i(w, b) = \sum_{j=1}^{m_i} \log\Big[1 + e^{-(w^\top p_{i,j} + b)\ell_{i,j}}\Big] + \frac{C}{2N}\|w\|^2, \quad i \in \{1, \ldots, N\}.$$

In a distributed setting, the goal is to make agents agree on a common solution $(w^\star, b^\star)$, so that all of them can compute the separating hyperplane as $\{z \in \mathbb{R}^d \mid (w^\star)^\top z + b^\star = 0\}$.

This problem is suited for the application of consensus-based primal methods (cf. Section 2) and a numerical example is shown in Section 2.5.

1.3.3 Classification via Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a popular tool used in (supervised) learning to build classification models. Suppose we have N points $p_1, \ldots, p_N \in \mathbb{R}^d$ (which represent training samples in a feature space) and suppose they are associated to binary labels, i.e., each p_i is labeled with $\ell_i \in \{-1, 1\}$, for all $i \in \{1, \ldots, N\}$. For simplicity, we consider linear SVM (more complex set-ups can be handled with appropriate transformations [17]). The problem consists of building a classification model from the training samples. In particular, we look for a separating hyperplane of the form $\{z \in \mathbb{R}^d \mid w^\top z + b = 0\}$ such that it separates all the points with $\ell_i = -1$ from all the points with $\ell_i = 1$. In symbols:

$$\begin{aligned} w^\top p_i + b &> 0, \quad \forall i \text{ such that } \ell_i = 1, \text{ and} \\ w^\top p_i + b &< 0, \quad \forall i \text{ such that } \ell_i = -1. \end{aligned}$$

In Figure 1.10, a classification example is shown.

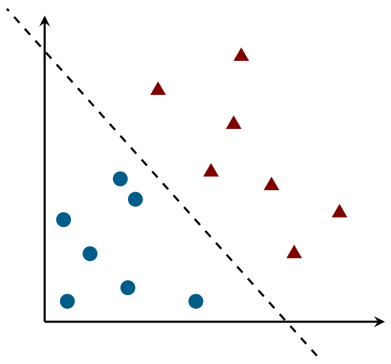


Figure 1.10: Graphical representation of a linear SVM problem in $\mathbb{R}^2$. The triangles and the dots represent points with different labels and the goal is to compute a separating hyperplane, denoted here by a dashed line.

In order to maximize the distance of the separating hyperplane from the training points, one can solve the following (convex) quadratic program:

$$\begin{aligned} &\min_{w,b} \ \frac{1}{2} w^\top w \\ &\text{subj. to } \ell_i(w^\top p_i + b) \geq 1, \quad i \in \{1, \ldots, N\}. \end{aligned} \tag{1.7}$$

Problem (1.7) is known in the literature as *hard-margin* SVM problem, and can be solved only if a separating hyperplane exists. However, if problem (1.7) is infeasible (e.g., when there are outliers), one can solve a *soft-margin* SVM problem in which some of the training samples are allowed to be on the "wrong side" of the hyperplane. Formally, we consider the following relaxation of problem (1.7):

$$\begin{aligned} \min_{w,b,\boldsymbol{\xi}} \;& \frac{1}{2} w^\top w + C \sum_{i=1}^{N} \xi_i \\ \text{subj. to } \;& \ell_i (w^\top p_i + b) \geq 1 - \xi_i, \qquad i \in \{1, \ldots, N\}, \\ & \boldsymbol{\xi} \geq 0, \end{aligned} \tag{1.8}$$

where we denote by $\boldsymbol{\xi}$ the vector stacking the violations $\xi_1, \ldots, \xi_N$ and $C > 0$ weighs the effect of the relaxation. Notice that problem (1.8) can be viewed either as a cost-coupled problem of the form (1.1), or as a common cost problem of the form (1.2).

In a distributed set-up, problem (1.8) must be solved by agents in a network. We suppose that each agent i is assigned exactly one training tuple (p_i, ℓ_i), so that each agent knows one constraint of the optimization problem. Agents eventually agree on an optimal solution $(w^\star, b^\star, \boldsymbol{\xi}^\star)$, so that the separating hyperplane can be computed as $\{z \in \mathbb{R}^d \mid (w^\star)^\top z + b^\star = 0\}$.

This problem is suited, e.g., for the application of constraint exchange methods (cf. Section 4.2) and a numerical example is shown in Section 4.4.

1.3.4 Target Localization in Sensor Networks

An interesting application in the field of sensor and robotic networks is the problem of estimating the position of a target, while having information on the position of sensors that can detect the unknown target within their field of sensing. A representational example of the problem is given in Figure 1.11. Formally, we suppose that N sensors are used to estimate in a distributed way the position of a target. Each sensor i knows its position $\mathbf{v}_i \in \mathbb{R}^2$ and the unknown target position is denoted by $\mathbf{x} \in \mathbb{R}^2$. We assume that sensors in the network detect the presence of the unknown target with two sensing mechanisms: *(i)* laser transmitters which scan through some angle, leading to a bounded cone set that can be expressed by three linear constraints, two bounding the angle and one bounding the distance, compactly written as $A_i \mathbf{x} \leq b_i$, with $A_i \in \mathbb{R}^{3\times 2}$ and $b_i \in \mathbb{R}^3$, and *(ii)* the range of the RF transmitter, leading to circular constraints of the form $\|\mathbf{x} - \mathbf{v}_i\|_2 \leq r_i$, where r_i denotes the

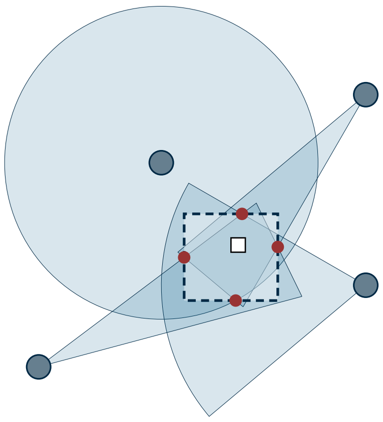


Figure 1.11: Localization of the target (white square) by set estimates of four sensors (blue nodes). The target is confined in the bounding box (dashed rectangle), which is determined by four extreme points (in red). The extreme points can be found by solving four instances of problem (1.9) with different cost vector c.

maximum sensing distance. Depending on the sensing mechanisms that each sensor i is equipped with, it is possible to bound the position of the unknown target to be contained in the intersection of convex sets X_i, each one known only by agent i, defined as $X_i \triangleq \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{v}_i\|_2 \leq r_i\}$ if the constraint is a disk, $X_i \triangleq \{\mathbf{x} \mid A_i\mathbf{x} \leq b_i\}$ if the constraint is a cone, $X_i \triangleq \{\mathbf{x} \mid A_i\mathbf{x} \leq b_i, \ \|\mathbf{x} - \mathbf{v}_i\|_2 \leq r_i\}$ if the constraint is a quadrant.

Now, the goal for the agents is to compute the smallest bounding box $\{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$, for suitable $\mathbf{x}^L, \mathbf{x}^U \in \mathbb{R}^2$, that is guaranteed to contain the unknown position of the additional target. This can be addressed by solving four optimization problems, one for each component of $\mathbf{x}^L, \mathbf{x}^U$. For instance, to compute the first component of $\mathbf{x}^L$, agents define the objective vector $c = [1, 0]^\top$ and they cooperatively solve the optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \ & c^\top \mathbf{x} \\ \text{subj. to } & \mathbf{x} \in \bigcap_{i=1}^{N} X_i, \end{aligned} \tag{1.9}$$

which is in the common cost form (1.2). After an optimal solution $\mathbf{x}^\star$ is found, each agent computes the first component of $\mathbf{x}^L$ by using the first component of $\mathbf{x}^\star$, and similarly for the other coordinates.

1.3.5 Task allocation/assignment

Task allocation is a building block for decision making problems in which a certain number of agents must be assigned given tasks. The goal is to find the best matching of agents and tasks according to a given performance criterion. Here, we consider N agents and N tasks and we look for a one-to-one assignment. Define the variable $x_{i\kappa}$, which is 1 if agent i is assigned to task κ and 0 otherwise. Also, define the set E_A, which contains the tuple (i,κ) if agent i can be assigned to task κ. Finally, let $c_{i\kappa}$ be the cost occurring if agent i is assigned to task κ. In Figure 1.12, we show an illustrative example of the set-up.

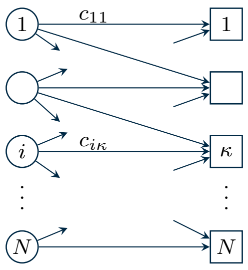


Figure 1.12: Graphical representation of the task assignment problem. Agents are represented by circles, while tasks are represented by squares. An arrow from agent i to task κ means that agent i can perform task κ (i.e., $(i,\kappa) \in E_A$) with corresponding cost equal to $c_{i\kappa}$.

Since the objective is to minimize the total cost, the task allocation problem can be formulated as an integer program. However, as pointed out in [8], integrality constraints can be dropped to obtain the linear program

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \sum_{(i,\kappa)\in E_A} c_{i\kappa} x_{i\kappa} \\
\text{subj. to} \quad & \mathbf{0} \le \mathbf{x} \le \mathbf{1}, \\
& \sum_{\{\kappa \mid (i,\kappa)\in E_A\}} x_{i\kappa} = 1 \quad \forall\, i \in \{1,\dots,N\}, \\
& \sum_{\{i \mid (i,\kappa)\in E_A\}} x_{i\kappa} = 1 \quad \forall\, \kappa \in \{1,\dots,N\},
\end{aligned} \tag{1.10}$$

where $\mathbf{x}$ is the variable stacking all $x_{i\kappa}$. If problem (1.10) is feasible, it can be shown that it admits an optimal solution such that $x_{i\kappa} \in \{0,1\}$ for all $(i,\kappa) \in E_A$ (see, e.g., [8]). Moreover, all the optimal assignments belong to the optimal solution set of problem (1.10).

Problem (1.10) can be cast to the constraint-coupled form (1.3). To see this, let us define K_i as the number of tasks that agent i can perform (i.e., $K_i = |\{\kappa \mid (i,\kappa) \in E_A\}|$). We assume that agent i deals with the variable $\mathbf{x}_i \in \mathbb{R}^{K_i}$, stacking the $x_{i\kappa}$ for all κ such that $(i,\kappa) \in E_A$. Then, the local sets X_i can be written as

$$X_i = \{\mathbf{x}_i \in \mathbb{R}^{K_i} \mid \mathbf{0} \le \mathbf{x}_i \le \mathbf{1} \text{ and } \mathbf{x}_i^\top \mathbf{1} = 1\}, \quad i \in \{1,\dots,N\}. \tag{1.11}$$

The coupling constraints can be written by defining, for all $i \in \{1,\dots,N\}$, the matrix $H_i \in \mathbb{R}^{N \times K_i}$, obtained by extracting from the $N \times N$ identity matrix the subset of columns corresponding to the tasks that agent i can perform. Problem (1.10) becomes

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \quad & \sum_{i=1}^N c_i^\top \mathbf{x}_i \\ \text{subj. to} \quad & \mathbf{x}_i \in X_i, \quad i \in \{1,\dots,N\} \\ & \sum_{i=1}^N H_i \mathbf{x}_i = \mathbf{1}, \end{aligned}$$

where each c_i stacks the costs $c_{i\kappa}$, for all κ such that $(i,\kappa) \in E_A$. Notice that problem (1.10) can be also tackled by resorting to its dual, which can be solved by using distributed optimization algorithms for common-cost problems.

In a distributed context, the goal for the agents is to find an optimal solution $\mathbf{x}^\star$, but each agent i is only interested in computing its portion $\mathbf{x}_i^\star$ of optimal solution, which contains only one entry $x_{i\kappa} = 1$, corresponding to the task κ that agent i is eventually assigned.

1.3.6 Cooperative Distributed Model Predictive Control

Model Predictive Control (MPC) is a widely studied technique in the control community, and is also used in distributed contexts. The goal is to design an optimization-based feedback control law for a (spatially distributed) network of dynamical systems. The leading idea is the principle of *receding horizon* control, which informally speaking consists of solving at each time step an optimization problem (usually termed *optimal control problem*), in which the system model is used to predict the system trajectory over a fixed time window. After an optimal solution of the optimal control problem is found, the input associated to the current time instant is applied and the process is repeated (for a survey on MPC methods, see, e.g., [116]).

Now, we describe a typical distributed MPC framework applied to a network of linear systems with linear coupling constraints. Formally, assume we have N discrete-time linear dynamical systems with independent dynamics of the form $\mathbf{z}_i(s+1) = A_i \mathbf{z}_i(s) + B_i \mathbf{u}_i(s)$, where $s \in \mathbb{Z}$ represents time; $\mathbf{z}_i(s) \in \mathbb{R}^{q_i}$ is the system state at time s; $\mathbf{u}_i(s) \in \mathbb{R}^{m_i}$ is the input fed to the system at time s; and A_i, B_i are given matrices of appropriate dimensions, for all $i \in \{1, \dots, N\}$. We suppose that the states and the inputs must satisfy local constraints $\mathbf{z}_i(s) \in Z_i$ and $\mathbf{u}_i(s) \in U_i$ for all $i \in \{1, \dots, N\}$, and that the agents' states are coupled to each other by means of coupling constraints of the form $\sum_{i=1}^N H_i \mathbf{z}_i(s) \le h$, for a given $h \in \mathbb{R}^P$. Given the initial conditions of the systems $\mathbf{z}_1^0, \dots, \mathbf{z}_N^0$, the optimal control problem to be solved is

$$
\begin{aligned}
\min_{\substack{\mathbf{z}_1,\dots,\mathbf{z}_N \\ \mathbf{u}_1,\dots,\mathbf{u}_N}} \; & \sum_{i=1}^N \bigg(\sum_{s=0}^{S-1} \ell_i(\mathbf{z}_i(s), \mathbf{u}_i(s)) + V_i(\mathbf{z}_i(S)) \bigg) \\
\text{subj. to } \; & \mathbf{z}_i(s+1) = A_i \mathbf{z}_i(s) + B_i \mathbf{u}_i(s), \quad s \in \{0, \dots, S-1\}, \forall\, i, \\
& \mathbf{z}_i(s) \in Z_i, \mathbf{u}_i(s-1) \in U_i \qquad s \in \{1, \dots, S\}, \qquad \forall\, i, \\
& \mathbf{z}_i(0) = \mathbf{z}_i^0, \qquad \forall\, i, \\
& \sum_{i=1}^N H_i \mathbf{z}_i(s) \le h, \qquad s \in \{1, \dots, S\},
\end{aligned}
\tag{1.12}
$$

where S is the *prediction horizon*, $\mathbf{z}_i = [\mathbf{z}_i(0)^\top, \dots, \mathbf{z}_i(S)^\top]^\top$ and $\mathbf{u}_i = [\mathbf{u}_i(0)^\top, \dots, \mathbf{u}_i(S-1)^\top]^\top$ are the optimization vectors, $\ell_i : \mathbb{R}^{q_i + m_i} \to \mathbb{R}$ is the *stage cost* and $V_i : \mathbb{R}^{q_i} \to \mathbb{R}$ is the *terminal cost*, for all $i \in \{1, \dots, N\}$. Problem (1.12) can be fit into the constraint-coupled set-up (1.3) by setting

$$
f_i(\mathbf{x}_i) = \sum_{s=0}^{S-1} \ell_i(\mathbf{z}_i(s), \mathbf{u}_i(s)) + V_i(\mathbf{z}_i(S)),
$$

$$\mathbf{g}_i(\mathbf{x}_i) = \begin{bmatrix} H_i\mathbf{z}_i(1) - \frac{h}{N} \\ \vdots \\ H_i\mathbf{z}_i(S) - \frac{h}{N} \end{bmatrix},$$

for all $i \in \{1, \dots, N\}$, with the local optimization variables being $\mathbf{x}_i = [\mathbf{z}_i^\top, \mathbf{u}_i^\top]^\top$ and the local constraint set X_i being

$$X_i \triangleq \Big\{(\mathbf{z}_i, \mathbf{u}_i) \in \mathbb{R}^{(S+1)q_i+Sm_i} \mid \mathbf{z}_i(s+1) = A_i\mathbf{z}_i(s) + B_i\mathbf{u}_i(s), \\ \mathbf{z}_i(s+1) \in Z_i, \mathbf{u}_i(s) \in U_i, \ \forall\, s\Big\},$$

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

Next, we describe an example of microgrid control scenario that can be fit into our distributed optimization framework. 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 GEN. At each time instant s in a given horizon $[0, S]$, they generate power, denoted by $p^s_{\mathtt{gen},i}$, 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 $i \in \mathtt{GEN}$, $\underline{p} \le p^s_{\mathtt{gen},i} \le \bar{p}$, with $s \in [0, S]$, and $\underline{r} \le p^{s+1}_{\mathtt{gen},i} - p^s_{\mathtt{gen},i} \le \bar{r}$, with $s \in [0, S-1]$. The cost to produce power by a generator is modeled as a quadratic function $f^s_{\mathtt{gen},i} = \alpha_1 p^s_{\mathtt{gen},i} + \alpha_2 (p^s_{\mathtt{gen},i})^2$ with α_1 and α_2 positive scalars. Storage devices are collected in STOR and their power is denoted by $p^s_{\mathtt{stor},i}$ and satisfies bounds and a dynamical constraint given by $-d_{\mathtt{stor}} \le p^s_{\mathtt{stor},i} \le c_{\mathtt{stor}}$, $s \in [0, S]$, $q^{s+1}_{\mathtt{stor},i} = q^s_{\mathtt{stor},i} + p^s_{\mathtt{stor},i}$, $s \in [0, S-1]$, and $0 \le q^s_{\mathtt{stor},i} \le q_{\max}$, $s \in [0, S]$, where the initial capacity $q^0_{\mathtt{stor},i}$ is given and $d_{\mathtt{stor}}$, $c_{\mathtt{stor}}$ and $q_{\max}$ are positive scalars. There are no costs associated with the stored power. Controllable loads are collected in CONL and their power is denoted by $p^s_{\mathtt{conl},i}$. The power must satisfy box constraints, i.e., $-P \le p^s_{\mathtt{conl},i} \le P$, $s \in [0, S]$. A desired load profile $p^s_{\mathtt{des},i}$ for $p^s_{\mathtt{conl},i}$ is given and the controllable load incurs in a cost $f^s_{\mathtt{conl},i} = \beta \max\{0, p^s_{\mathtt{des},i} - p^s_{\mathtt{conl},i}\}$, $\beta \ge 0$, 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 $p^s_{\mathtt{tr}}$ and must satisfy $|p^s_{\mathtt{tr}}| \le E$, $s \in [0, S]$. The power-trading cost is modeled as $f^s_{\mathtt{tr}} = -c_1 p^s_{\mathtt{tr}} + c_2 |p^s_{\mathtt{tr}}|$, with c_1 and c_2 positive scalars corresponding to the price and a general transaction cost respectively.

The power network must provide at least a given power demand D^s, which can be modeled by a *coupling constraint* among the units

$$\sum_{i\in\texttt{GEN}} p^s_{\texttt{gen},i} + \sum_{i\in\texttt{STOR}} p^s_{\texttt{stor},i} + \sum_{i\in\texttt{CONL}} p^s_{\texttt{conl},i} + p^s_{\texttt{tr}} \geq D^s, \tag{1.13}$$

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

Notice that the microgrid control problem can be cast in the constraint-coupled form (1.3). To this end, we let each $\mathbf{x}_i$ be the whole trajectory over the prediction horizon $[0, S]$, i.e.,

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

for all the generators $i \in \texttt{GEN}$ and, consistently, for the other device types. As for the cost functions, we define

$$f_i(\mathbf{x}_i) \triangleq \sum_{s=0}^{S} f^s_{\texttt{gen},i}(p^s_{\texttt{gen},i})$$

for all the generators $i \in \texttt{GEN}$ and, consistently, for the other device types. The local constraint sets X_i are given by

$$\begin{aligned} X_i \triangleq \Big\{[p^0_{\texttt{gen},i}, \ldots, p^S_{\texttt{gen},i}]^\top \mid \underline{p} \leq p^s_{\texttt{gen},i} \leq \bar{p},\ \tau \in [0, S], \\ \underline{r} \leq p^{s+1}_{\texttt{gen},i} - p^s_{\texttt{gen},i} \leq \bar{r}, \quad \tau \in [0, S-1]\Big\}, \end{aligned}$$

for all the generators $i \in \texttt{GEN}$ and, consistently, for the other device types. The coupling constraints are as in (1.13).

This problem is suited, e.g., for the application of duality-based methods for constraint-coupled problems (cf. Section 3.4) and a numerical example is shown in Section 3.6.

2

Consensus-Based Primal Methods

In this chapter we focus on primal approaches to design distributed algorithms for cost-coupled problems. We start by describing the so-called distributed subgradient method, based on a combination of the average consensus protocol with the subgradient method. Then, we present a recent improvement of such consensus-based scheme, named gradient tracking, that relies on the novel idea of tracking the gradient of the global cost function via a dynamic consensus scheme. Then, we provide extensions to the basic schemes. Finally, we show a numerical example to compare the two presented algorithms.

2.1 Distributed Subgradient Method

In this section we review the distributed subgradient method that has been proposed in the pioneering works [95, 96] (see also the tutorial papers [87, 88, 91]). In this survey, we report a proof based on the analysis proposed in the references above.

As already described in Section 1.2.1, we consider a network of N

agents that aim to cooperatively solve the cost-coupled problem

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_{i=1}^{N} f_i(\mathbf{x}) \\ \text{subj. to } & \mathbf{x} \in \mathbb{R}^d, \end{aligned} \tag{2.1}$$

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

A natural way to devise a distributed algorithm for problem (2.1) is to study how it would be solved through a centralized gradient-based approach. We recall that a subgradient method applied to (2.1) consists of an iterative procedure in which the current solution estimate, denoted by $\mathbf{x}^t$, is updated according to

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \gamma^t \sum_{i=1}^{N} \widetilde{\nabla} f_i(\mathbf{x}^t),$$

where γ^t is the step-size and $\sum_{i=1}^{N} \widetilde{\nabla} f_i(\mathbf{x}^t)$ is a subgradient of the cost function at $\mathbf{x}^t$. The initial value $\mathbf{x}^0$ can be set to any element of $\mathbb{R}^d$.

Next, we introduce the distributed subgradient algorithm proposed in [95, 96]. Each agent i maintains its own estimate $\mathbf{x}_i^t$ of the decision variable $\mathbf{x}$, initialized to any value in $\mathbb{R}^d$ and iteratively updated until it eventually converges to an optimal solution of (2.1). The distributed subgradient algorithm is based on the combination of a consensus protocol (cf. Appendix B) with the subgradient optimization method (cf. Appendix A.1) to move each local solution estimate toward an optimal (common) solution of problem (2.1). Algorithm 1 summarizes the distributed subgradient algorithm from the perspective of node i.

For presentation purposes, in this survey we consider a simplified network configuration, so that the core idea of the scheme can be easily caught. That is, the network is modeled as a fixed, connected and undirected graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$. The weights a_{ij} in (2.2a) inherit the typical assumptions on consensus protocols, formally reported next.

Assumption 2.1. Let the weights a_{ij}, $i, j \in \{1, \dots, N\}$ be nonnegative entries of $A \in \mathbb{R}^{N \times N}$ that match the graph $\mathcal{G}$, i.e., $a_{ij} \neq 0$ for all $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. Moreover, they satisfy

- $\sum_{j=1}^{N} a_{ij} = 1$, for all $i \in \{1, \dots, N\}$;

Algorithm 1 Distributed Subgradient

```
Initialization: $\mathbf{x}_i^0 \in \mathbb{R}^d$
Evolution: for $t = 0, 1, \ldots$
  Gather $\mathbf{x}_j^t$ from neighbors $j \in \mathcal{N}_i$
  Compute
      $\mathbf{v}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij}\, \mathbf{x}_j^t$    (2.2a)
  Update
      $\mathbf{x}_i^{t+1} = \mathbf{v}_i^{t+1} - \gamma^t\, \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1})$    (2.2b)
```

- $\sum_{i=1}^{N} a_{ij} = 1$, for all $j \in \{1, \ldots, N\}$;
- for all $i \in \{1, \ldots, N\}$, $a_{ii} > 0$. □

We point out that one may also consider strongly connected directed graphs that admits a doubly-stochastic weighted adjacency matrix. Detailed convergence analyses of distributed subgradient schemes have been provided, e.g., in [95, 96, 87, 91, 88]. For the sake of completeness, this survey provides a proof for the convergence of Algorithm 1 that is mainly inspired by the references above.

We start by stating the condition on the step-size γ^t used in the update (2.2b). As in the standard (centralized) subgradient method, it must satisfy a diminishing property.

Assumption 2.2. The step-size sequence $\{\gamma^t\}_{t \geq 0}$, with $\gamma^t \geq 0$, satisfies the conditions $\sum_{t=0}^{\infty} \gamma^t = \infty$, $\sum_{t=0}^{\infty} \left(\gamma^t\right)^2 < \infty$. □

As a consequence of the square summability in Assumption 2.2, the step-size vanishes as the algorithm proceeds, i.e., $\lim_{t \to \infty} \gamma^t = 0$.

Next, we state regularity requirements for problem (2.1).

Assumption 2.3. Let the following conditions hold:

(i) each $f_i : \mathbb{R}^d \to \mathbb{R}$ is convex and has bounded subgradients, i.e., there exists a scalar $C_i > 0$ such that $\|\widetilde{\nabla} f_i(\mathbf{x})\| \leq C_i$ for any subgradient $\widetilde{\nabla} f_i(\mathbf{x})$ of f_i at any $\mathbf{x} \in \mathbb{R}^d$;

(ii) problem (2.1) has at least one optimal solution, i.e., the optimal solution set $X^\star = \{\mathbf{x} \in \mathbb{R}^d \mid f(\mathbf{x}) = f^\star\}$ is nonempty, where $f^\star$ denotes the optimal value of problem (2.1). $\square$

Usually, in the analysis of consensus-based algorithms, it is useful to introduce the average of the quantities that are required to be asymptotically consensual. Here, we introduce the average of the current solution estimates, i.e., for all $t \geq 0$ we define

$$\bar{\mathbf{x}}^t \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^t. \tag{2.3}$$

We point out that $\bar{\mathbf{x}}^t \in \mathbb{R}^d$ has the same dimension of the local solution estimates $\mathbf{x}_i^t$ and is introduced only for the sake of analysis. Of course, it cannot be computed by any agent and, nevertheless, it does not need to be known. We observe that $\bar{\mathbf{x}}^t$ evolves according to its own dynamics, which can be obtained by combining the local updates of the agents. Formally, it holds

$$\begin{aligned} \bar{\mathbf{x}}^{t+1} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^{t+1} = \frac{1}{N} \sum_{i=1}^{N} \Big(\mathbf{v}_i^{t+1} - \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1}) \Big) \\ &= \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}\, \mathbf{x}_j^t - \gamma^t \frac{1}{N} \sum_{i=1}^{N} \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1}) \\ &= \bar{\mathbf{x}}^t - \gamma^t \frac{1}{N} \sum_{i=1}^{N} \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1}), \end{aligned} \tag{2.4}$$

where we used the (row) stochasticity of the weights a_{ij}.

The following result (see [11] for a proof) is an important building block for the forthcoming proof of the convergence of Algorithm 1.

Lemma 2.4. Let $\{Y^t\}_{t\geq 0}$, $\{W^t\}_{t\geq 0}$, and $\{Z^t\}_{t\geq 0}$ be three scalar sequences such that $W^t t$ is nonnegative for all t. Assume the following

$$\begin{aligned} &Y^{t+1} \leq Y^t - W^t + Z^t, \qquad t \geq 0, \\ &\sum_{t=0}^{\infty} Z^t < \infty. \end{aligned}$$

Then either $\lim_{t\to\infty} Y^t = -\infty$ or else $\{Y^t\}_{t\geq 0}$ converges to a finite value and $\sum_{t=0}^{\infty} W^t < \infty$. $\square$

The following theorem, also provided, e.g., in [95, 96, 87, 88, 91], formally states the convergence properties of Algorithm 1. For ease of notation, we consider a scalar optimization problem, i.e., $d = 1$.

Theorem 2.5. Let Assumptions 2.1, 2.2 and 2.3 hold and let the communication graph be undirected and connected. Then, the sequences of local solution estimates $\{\mathbf{x}_i^t\}_{t\geq 0}$, $i \in \{1, \ldots, N\}$, generated by Algorithm 1, converge to a (common) solution of problem (2.1), i.e., for all $i \in \{1, \ldots, N\}$, it holds

$$\lim_{t\to\infty} \|\mathbf{x}_i^t - \mathbf{x}^\star\| = 0, \tag{2.5}$$

for some $\mathbf{x}^\star \in X^\star$.

Proof. The proof provided in this manuscript is mainly based on the ones given in [95, 96, 87, 88, 91]. It is based on showing the following three steps:

1. asymptotic consensus of the local solution estimates to their average, i.e.,

$$\lim_{t\to\infty} \|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\| = 0, \tag{2.6}$$

   for all $i \in \{1, \ldots, N\}$;

2. summability of the consensus error weighted by the step-size, i.e.,

$$\lim_{T\to\infty} \sum_{t=0}^{T} \gamma^t \|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\| < \infty; \tag{2.7}$$

3. convergence of the average sequence $\{\bar{\mathbf{x}}^t\}_{t\geq 0}$ to an optimal solution of problem (2.1), i.e.,

$$\lim_{t\to\infty} \|\bar{\mathbf{x}}^t - \mathbf{x}^\star\| = 0, \tag{2.8}$$

   for some $\mathbf{x}^\star \in X^\star$.

Let $\mathbf{x}^t$ be the vector stacking the local solution estimates $\mathbf{x}_i^t$. Then, the consensus error evolution is given by

$$\mathbf{x}^{t+1} - \bar{\mathbf{x}}^{t+1}\mathbf{1} = A\mathbf{x}^t + \boldsymbol{\epsilon}^t - \bar{\mathbf{x}}^t\mathbf{1} - \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{\epsilon}_j^t$$

$$
\begin{aligned}
&= (A - \mathbf{1}\mathbf{1}^\top/N)\mathbf{x}^t + \boldsymbol{\epsilon}^t - \frac{1}{N}\sum_{j=1}^N \boldsymbol{\epsilon}_j^t \\
&= (A - \mathbf{1}\mathbf{1}^\top/N)(\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}) + \boldsymbol{\epsilon}^t - \frac{1}{N}\sum_{j=1}^N \boldsymbol{\epsilon}_j^t \\
&= (A - \mathbf{1}\mathbf{1}^\top/N)(\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}) + (I - \mathbf{1}\mathbf{1}^\top/N)\boldsymbol{\epsilon}^t,
\end{aligned}
$$

where $\boldsymbol{\epsilon}^t$ denotes the vector stacking all the $\boldsymbol{\epsilon}_i^t$ with the short-hand $\boldsymbol{\epsilon}_i^t$ for $-\gamma^t\widetilde{\nabla} f_i(\mathbf{v}_i^{t+1})$, for all $i \in \{1, \dots, N\}$.

Taking the norm of both sides in the last equation and applying the triangle inequality leads to

$$
\begin{aligned}
\|\mathbf{x}^{t+1} - \bar{\mathbf{x}}^{t+1}\mathbf{1}\| &\leq \|A - \mathbf{1}\mathbf{1}^\top/N\|\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \|I - \mathbf{1}\mathbf{1}^\top/N\|\|\boldsymbol{\epsilon}^t\| \\
&\leq \sigma_A\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \|\boldsymbol{\epsilon}^t\|,
\end{aligned}
$$

where we used the sub-multiplicative property of the 2-norm, we set $\sigma_A = \|A - \mathbf{1}\mathbf{1}^\top/N\|$ (i.e., the contraction factor associated to the matrix A, cf. Appendix B.1), and we used the bound $\|I - \mathbf{1}\mathbf{1}^\top/N\| \leq 1$.

By using the explicit solution for the free evolution and the forced evolution of a linear time-invariant system, the term $\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\|$ can be bounded as follows

$$
\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| \leq \sigma_A^t\|\mathbf{x}^0 - \bar{\mathbf{x}}^0\mathbf{1}\| + \sum_{\tau=0}^{t-1}\sigma_A^{t-1-\tau}\|\boldsymbol{\epsilon}^\tau\|.
$$

Since by assumption $\|\boldsymbol{\epsilon}^\tau\| \to 0$ (cf. Assumption 2.2 and 2.3(i)) and $\sigma_A \in (0, 1)$, it can be proven that

$$
\lim_{t\to\infty}\sum_{\tau=0}^{t-1}\sigma_A^{t-1-\tau}\|\boldsymbol{\epsilon}^\tau\| = 0, \tag{2.9}
$$

which, in turns, implies that

$$
\lim_{t\to\infty}\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| = 0. \tag{2.10}
$$

Next we show the summability condition. It holds

$$\begin{aligned}\lim_{T\to\infty}\sum_{t=0}^{T}\gamma^t\|\mathbf{x}^t-\bar{\mathbf{x}}^t\mathbf{1}\| &\leq \lim_{T\to\infty}\sum_{t=0}^{T}\gamma^t\sigma_A^t\|\mathbf{x}^0-\bar{\mathbf{x}}^0\mathbf{1}\|\\ &\quad+\lim_{T\to\infty}\sum_{t=0}^{T}\gamma^t\sum_{\tau=0}^{t-1}\sigma_A^{t-1-\tau}\|\boldsymbol{\epsilon}^\tau\|\\ &\overset{(a)}{=}\lim_{T\to\infty}\sum_{t=0}^{T}\gamma^t\sigma_A^t\|\mathbf{x}^0-\bar{\mathbf{x}}^0\mathbf{1}\| \qquad (2.11)\\ &\quad+\lim_{T\to\infty}\sum_{t=0}^{T}\gamma^t\sum_{\tau=0}^{t-1}\sigma_A^{t-1-\tau}\gamma^\tau C\\ &\overset{(b)}{\leq}\kappa,\end{aligned}$$

for some finite κ, where in (a) we rearranged terms; in (b) the first term is bounded due to geometric series properties (cf. Assumption 2.2 and recall $\sigma_A\in(0,1)$), while the second one can be shown to be bounded by using the Young's inequality[1] to write

$$\sum_{t=0}^{T}\gamma^t\sum_{\tau=0}^{t-1}\sigma_A^{t-1-\tau}\gamma^\tau\leq\sum_{t=0}^{T}\sum_{\tau=0}^{t-1}\sigma_A^{t-1-\tau}\Big((\gamma^t)^2+(\gamma^\tau)^2\Big)/2,$$

and, then, exploiting subgradient boundedness (cf. Assumption 2.3), geometric series properties (recall $\sigma_A\in(0,1)$) and the step-size properties (cf. square summability of γ^t in Assumption 2.2).

Finally, we study convergence to the optimum by showing that a proper candidate function, say V^t, decreases along the algorithmic evolution. Let V^t be a measure of the distance between the local solution estimates $\mathbf{x}_i^t$, $i\in\{1,\dots,N\}$, and an optimal solution to problem (2.1), i.e.,

$$V^t\triangleq\sum_{i=1}^{N}\|\mathbf{x}_i^t-\mathbf{x}^\star\|^2, \qquad (2.12)$$

where $\mathbf{x}^\star\in X^\star$. Due to convexity of each f_i and subgradient bounded-

[1]For all $a\geq 0$, and $b\geq 0$, it holds $2ab\leq a^2+b^2$.

ness (cf. Assumption 2.3), it follows that

$$\begin{aligned}
V^{t+1} = \sum_{i=1}^{N} \|\mathbf{x}_i^{t+1} - \mathbf{x}^\star\|^2 &= \sum_{i=1}^{N} \|\mathbf{v}_i^{t+1} - \gamma^t \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1}) - \mathbf{x}^\star\|^2 \\
&= \sum_{i=1}^{N} \|\mathbf{v}_i^{t+1} - \mathbf{x}^\star\|^2 + \sum_{i=1}^{N} \left\|\gamma^t \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1})\right\|^2 \\
&\quad - 2\gamma^t \sum_{i=1}^{N} \widetilde{\nabla} f_i(\mathbf{v}_i^{t+1})^\top (\mathbf{v}_i^{t+1} - \mathbf{x}^\star) \\
&\overset{(a)}{\leq} \sum_{j=1}^{N} \left(\sum_{i=1}^{N} a_{ij}\right) \|\mathbf{x}_j^t - \mathbf{x}^\star\|^2 + (\gamma^t)^2 \sum_{i=1}^{N} C_i^2 \\
&\quad - 2\gamma^t \sum_{i=1}^{N} \Big(f_i(\mathbf{v}_i^{t+1}) - f_i(\mathbf{x}^\star)\Big),
\end{aligned}$$

where in (a) we exploited convexity of the square norm $\|\cdot\|^2$ and weights properties (cf. Assumption 2.1) to write $\sum_{i=1}^N \|\mathbf{v}_i^{t+1} - \mathbf{x}^\star\|^2 = \sum_{i=1}^N \|\sum_{j\in\mathcal{N}_i} a_{ij}(\mathbf{x}_j^t - \mathbf{x}^\star)\|^2 \leq \sum_{j=1}^N (\sum_{i=1}^N a_{ij})\|\mathbf{x}_j^t - \mathbf{x}^\star\|^2$; subgradient boundedness (cf. Assumption 2.3); and the subgradient definition (cf. Appendix A.2. Compactly it holds

$$V^{t+1} \leq V^t - 2\,\gamma^t \sum_{i=1}^{N} \Big(f_i(\mathbf{v}_i^{t+1}) - f_i(\mathbf{x}^\star)\Big) + (\gamma^t)^2 C^2, \tag{2.13}$$

where $C^2 \triangleq \sum_{i=1}^N C_i^2$. Adding and subtracting $2\,\gamma^t \sum_{i=1}^N f_i(\bar{\mathbf{x}}^{t+1})$ yields

$$\begin{aligned}
V^{t+1} &\leq V^t - 2\gamma^t \sum_{i=1}^{N} \Big(f_i(\bar{\mathbf{x}}^{t+1}) - f_i(\mathbf{x}^\star)\Big) \\
&\quad + 2\gamma^t \sum_{i=1}^{N} \Big(f_i(\bar{\mathbf{x}}^{t+1}) - f_i(\mathbf{v}_i^{t+1})\Big) + (\gamma^t)^2 C^2 \\
&\overset{(a)}{\leq} V^t - 2\gamma^t \sum_{i=1}^{N} \Big(f_i(\bar{\mathbf{x}}^{t+1}) - f_i(\mathbf{x}^\star)\Big) \\
&\quad + 2C\gamma^t \sum_{i=1}^{N} \|\bar{\mathbf{x}}^{t+1} - \mathbf{x}_i^t\| + (\gamma^t)^2 C^2,
\end{aligned} \tag{2.14}$$

where in (a) we used subgradient boundedness to write

$$\begin{aligned}
\sum_{i=1}^{N} \left|f_i(\bar{\mathbf{x}}^{t+1}) - f_i(\mathbf{v}_i^{t+1})\right| &\leq C \sum_{i=1}^{N} \|\bar{\mathbf{x}}^{t+1} - \sum_{j\in\mathcal{N}_i} a_{ij}\mathbf{x}_j^t\| \\
&\leq C \sum_{i=1}^{N} \|\bar{\mathbf{x}}^{t+1} - \mathbf{x}_i^t\|.
\end{aligned}$$

Notice that $\sum_{i=1}^{N}(f_i(\bar{\mathbf{x}}^{t+1}) - f_i(\mathbf{x}^\star)) > 0$ since $\mathbf{x}^\star$ is a minimum of (2.1). Using Lemma 2.4 we can conclude that:

- the sequence $\{V^t\}_{t\geq 0}$ converges to a finite value, say $\bar{V}$, for every $\mathbf{x}^\star \in X^*$, and
- the average sequence $\{\bar{\mathbf{x}}^t\}_{t\geq 0}$ satisfies

$$\liminf_{t\to\infty} \sum_{i=1}^{N} f_i(\bar{\mathbf{x}}^{t+1}) = \sum_{i=1}^{N} f_i(\mathbf{x}^\star) = f^\star. \tag{2.15}$$

Since the sequence $\{V^t\}_{t\geq 0}$ (cf. its definition in (2.12)) converges, then also the sequence $\{\sum_{i=1}^{N} \|\mathbf{x}_i^t - \mathbf{x}^\star\|\}_{t\geq 0}$ converges, for every $\mathbf{x}^\star \in X^\star$. Moreover, recall that by consensus achievement (2.10), it holds $\lim_{t\to\infty} \|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\| = 0$. Therefore, also $\{\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\|\}_{t\geq 0}$ must converge.

In view of (2.15) and of continuity of f (due to its convexity), one of the limit points of $\{\bar{\mathbf{x}}^t\}_{t\geq 0}$ must belong to $X^\star$; thus, consider a subsequence $\{\bar{\mathbf{x}}^{t_k}\}_{k\geq 0}$ of $\{\bar{\mathbf{x}}^t\}_{t\geq 0}$ converging to an optimum, i.e., such that $\lim_{k\to\infty} \|\bar{\mathbf{x}}^{t_k} - \mathbf{x}^\infty\| = 0$, with $\mathbf{x}^\infty \in X^\star$. Convergence of $\bar{\mathbf{x}}^{t_k}$ with the asymptotic consensus property (cf. eq. (2.10)) implies that also $\lim_{k\to\infty} \|\mathbf{x}_i^{t_k} - \mathbf{x}^\infty\| = 0$, for all $i \in \{1, \ldots, N\}$. But in view of convergence of $V^t = \sum_{i=1}^{N} \|\mathbf{x}_i^t - \mathbf{x}^\star\|^2$, it must be that the (entire) sequence $\{\mathbf{x}_i^t\}_{\geq 0}$ converges to $\mathbf{x}^\star \in X^\star$. □

It is worth mentioning that convergence of the distributed subgradient algorithm to an optimum can only be guaranteed with a diminishing step-size. This is mainly due to the fact that at each iteration, each agent i considers an update direction depending only on its local objective function f_i, rather than on the entire cost function $\sum_{i=1}^{N} f_i$.

Convergence rates have been proven for the distributed subgradient method and its variants. In [89], a convergence rate of $\mathcal{O}(\ln t/\sqrt{t})$ is proved for an extension of the distributed subgradient algorithm for directed graphs.

2.2 Gradient Tracking Algorithm

In this section we review a recent method for cost-coupled problems (cf. Section 1.2.1) that exhibits a faster convergence rate because it allows for the use of a constant step-size. The underlying idea of this novel approach is to implement a distributed consensus-based mechanism to track the gradient of the whole cost function. Thanks to this tracking mechanism, a linear convergence rate has been shown for this scheme, matching the rate of the centralized gradient method.

Formally, we consider a cost-coupled problem in the form (2.1), where the cost functions f_i satisfy suitable regularity properties that will be specified next.

In order to understand the concept underlying the gradient tracking algorithm, let us consider the (centralized) gradient method applied to (2.1). If we denote by $\mathbf{x}^t$ the (centralized) solution estimate, the method reads

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \gamma \sum_{h=1}^{N} \nabla f_h(\mathbf{x}^t). \tag{2.16}$$

In a distributed context, each agent i has its own version $\mathbf{x}_i^t$ of the current solution estimate $\mathbf{x}^t$. Thus, the gradient scheme (2.16) can be adapted as follows

$$\mathbf{x}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij}\mathbf{x}_j^t - \gamma \sum_{h=1}^{N} \nabla f_h(\mathbf{x}_h^t),$$

where the consensus iteration $\sum_{j \in \mathcal{N}_i} a_{ij}\mathbf{x}_j^t$ is meant to enforce an agreement among the agents. However, still the descent direction $\sum_{h=1}^{N} \nabla f_h(\mathbf{x}_h^t)$ requires a global knowledge that is not locally available. To overcome this issue, the exact (centralized) descent direction is replaced by a local descent direction, say $\mathbf{y}_i^t$, which is updated through a *dynamic* average consensus iteration to eventually track $\sum_{h=1}^{N} \nabla f_h(\mathbf{x}_h^t)$. Informally, the dynamic average consensus is a distributed algorithm in which each agent i has access only to its local (possibly time-varying) signal, say $\mathbf{r}_i^t$, and wants to track the (time-varying) average signal $1/N \cdot \sum_{h=1}^{N} \mathbf{r}_h^t$ by exchanging information only with neighbors. See Appendix B.3 for further details. In the context of gradient tracking, each agent's signal is the local gradient at the current estimate, i.e., $\mathbf{r}_i^t = \nabla f_i(\mathbf{x}_i^t)$. The following table (Algorithm 2) formally summarizes the gradient tracking algorithm from the perspective of agent i, where eq. (2.18) describes the dynamic average consensus iteration for the tracking of $\sum_{h=1}^{N} \nabla f_h(\mathbf{x}_h^t)$, the local solution estimate $\mathbf{x}_i^t$ is initialized to any vector in $\mathbb{R}^d$ and the gradient tracker $\mathbf{y}_i^t$ is initialized to $\nabla f_i(\mathbf{x}_i^0)$.

Algorithm 2 Gradient Tracking

```
  **Initialization**: $\mathbf{x}_i^0$ and $\mathbf{y}_i^0 = \nabla f_i(\mathbf{x}_i^0)$
  **Evolution**: for $t = 0, 1, ...$
    **Gather** $\mathbf{x}_j^t$ from neighbors $j \in \mathcal{N}_i$
    **Update**
        $\mathbf{x}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij}\,\mathbf{x}_j^t - \gamma\,\mathbf{y}_i^t$    (2.17)
    **Gather** $\mathbf{y}_j^t$ from neighbors $j \in \mathcal{N}_i$
    **Update**
        $\mathbf{y}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij}\,\mathbf{y}_j^t + (\nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\mathbf{x}_i^t))$    (2.18)
```

Gradient tracking algorithms have been proposed with several names and versions in the literature, but with a common underlying idea. Early works [157, 158, 137] propose the novel idea of distributively tracking a Newton-Raphson direction by means of suitable average consensus ratios. In [23] the same approach has been extended to deal with directed, asynchronous networks with lossy communication. More recently, the idea of gradient tracking has been independently proposed by several research groups. In [32, 33] the authors consider constrained nonsmooth and nonconvex problems, while in [151, 153] strongly convex, unconstrained, smooth optimization problems are addressed with agent-specific stepsizes. Works [128, 127] extend the algorithms to (possibly) time-varying digraphs (still in a nonconvex setting). A convergence rate analysis of the scheme was later developed in [92, 93, 111, 113, 153], where [92, 93] consider time-varying (directed) graphs. Several other recent works investigate the same scheme under numerous variants, such as [94, 112, 147, 150].

In order to highlight the key tools needed for the analysis of such class of algorithms, in this survey we investigate a simplified scenario that is characterized afterwards.

Assumption 2.6. For all $i \in \{1, \ldots, N\}$, each cost function $f_i : \mathbb{R}^d \to \mathbb{R}$ satisfies the following conditions

- it is α-strongly convex, i.e.,

$$f_i(\mathbf{w}) \geq f_i(\mathbf{z}) + \nabla f_i(\mathbf{z})^\top (\mathbf{w} - \mathbf{z}) + \frac{\alpha}{2}\|\mathbf{w} - \mathbf{z}\|^2,$$

  for all $\mathbf{w}, \mathbf{z} \in \mathbb{R}^d$ and $\alpha > 0$;

- it has Lipschitz continuous gradient with constant $L > 0$, i.e.,

$$\|\nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{z})\| \leq L\|\mathbf{w} - \mathbf{z}\|,$$

  for all $\mathbf{w}, \mathbf{z} \in \mathbb{R}^d$. □

Since each f_i is a strongly convex function, then also their sum is strongly convex. Thus under Assumption 2.6, problem (2.1) has a unique optimal solution, denoted by $\mathbf{x}^\star$. Notice that it holds $\alpha \leq L$. We point out that one can also consider a more general case in which each f_i has L_i-Lipschitz continuous gradient. The results proved next still hold by setting in the analysis $L = \sum_{i=1}^N L_i$.

Similarly to the distributed subgradient algorithm in Section 2.1, we consider a simple network scenario modeled as a fixed, connected and undirected graph $\mathcal{G} = (\{1, \ldots, N\}, \mathcal{E})$. We assume the weights a_{ij} satisfy a double stochasticity property as formalized in Assumption 2.1.

The gradient tracking scheme has been proposed in [33, 127] with a diminishing step-size γ^t. As in the distributed subgradient algorithm (cf. Section 2.1), this choice allows one to decouple the convergence analysis in two independent parts, i.e., consensus achievement and asymptotic convergence of the consensual value to the optimum. When a constant step-size γ is used, as done in this survey, the proof cannot be split in two parts anymore, but consensus and optimality need to be handled simultaneously.

Since the gradient tracking algorithm is a consensus-based scheme, it is convenient to introduce average quantities of local agent variables. Namely, we define the average of the solution estimates and the average of the trackers as

$$\bar{\mathbf{x}}^t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i^t$$

$$\bar{\mathbf{y}}^t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i^t,$$

for all $t \geq 0$. Using simple algebraic manipulations, it can be shown that the average quantities evolve as the following linear dynamical system

$$\bar{\mathbf{x}}^{t+1} = \bar{\mathbf{x}}^t - \gamma\, \bar{\mathbf{y}}^t \tag{2.19}$$

$$\bar{\mathbf{y}}^{t+1} = \bar{\mathbf{y}}^t + \frac{1}{N}\sum_{i=1}^{N}\Big(\nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\mathbf{x}_i^t)\Big). \tag{2.20}$$

By exploiting the (column) stochasticity of consensus weights (cf. Assumption 2.1) and the initialization of the trackers, i.e., $\mathbf{y}_i^0 = \nabla f_i(\mathbf{x}_i^0)$, one can show that a conservation property for the tracker average $\bar{\mathbf{y}}^t$ holds. That is

$$\begin{aligned}\bar{\mathbf{y}}^{t+1} - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{x}_i^{t+1}) &= \bar{\mathbf{y}}^t - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{x}_i^t)\\ &= \bar{\mathbf{y}}^0 - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{x}_i^0) = 0,\end{aligned} \tag{2.21}$$

which implies $\bar{\mathbf{y}}^t = 1/N \cdot \sum_{i=1}^{N} \nabla f_i(\mathbf{x}_i^t)$, for all $t \geq 0$.

The analysis we propose is mainly a detailed version of the proof provided in [150] for the above simplified scenario. In addition, we consider a scalar optimization problem, i.e., we set $d = 1$.

The proof starts by characterizing the interconnection among the following quantities:

- consensus error $\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\|$, where $\mathbf{x}^t$ stacks all the $\mathbf{x}_i^t$;
- gradient tracking error $\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\|$, where $\mathbf{y}^t$ stacks all the $\mathbf{y}_i^t$;
- distance from optimality of the average $\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\|$, where $\mathbf{x}^\star$ is the optimal solution of problem (2.1).

We first recall a preliminary result which relies on Lipschitz continuity of the cost gradients.

Lemma 2.7. Let $\nabla f(\mathbf{x}^t)$ denote the vector stacking all the gradients $\nabla f_i(\mathbf{x}_i^t)$, $i \in \{1, \dots, N\}$. Under Assumptions 2.6 and 2.1, it holds that

$$\|\nabla f(\mathbf{x}^{t+1}) - \nabla f(\mathbf{x}^t)\| \le L\|\mathbf{x}^{t+1} - \mathbf{x}^t\|,$$
$$\left\|\frac{1}{N}\sum_{i=1}^N \left(\nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\mathbf{x}_i^t)\right)\right\| \le \frac{L}{\sqrt{N}}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|,$$
$$\left\|\frac{1}{N}\sum_{i=1}^N \left(\nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\bar{\mathbf{x}}^t)\right)\right\| \le \frac{L}{\sqrt{N}}\|\mathbf{x}^{t+1} - \bar{\mathbf{x}}^t\mathbf{1}\|,$$

where L is the Lipschitz constant of ∇f_i, $i \in \{1, \dots, N\}$. $\square$

The previous lemma can be easily shown by exploiting the basic algebraic property $\sum_{i=1}^N \|\theta_i\|_2 \le \sqrt{N}\|[\theta_1, \dots, \theta_N]^\top\|_2$, which follows by concavity of the square root function.

Next, we provide a list of intermediate results that will be used in the convergence theorem. They explicitly provide linear upper bounds for the three quantities introduced above. The following lemma characterizes the consensus error.

Lemma 2.8. Under Assumption 2.6, it holds

$$\|\mathbf{x}^{t+1} - \bar{\mathbf{x}}^{t+1}\mathbf{1}\| \le \sigma_A\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \gamma\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\|,$$

for all $t \ge 0$, where $\sigma_A \in (0, 1)$.

Proof. From (2.17) and (2.19), we can write

$$\begin{aligned}\|\mathbf{x}^{t+1} - \bar{\mathbf{x}}^{t+1}\mathbf{1}\| &= \|A\mathbf{x}^t - \gamma\mathbf{y}^t - (\bar{\mathbf{x}}^t - \gamma\bar{\mathbf{y}}^t)\mathbf{1}\| \\ &\le \|A\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \gamma\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| \\ &\le \sigma_A\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \gamma\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\|,\end{aligned}$$

where we used the triangle inequality and σ_A is the contraction factor associated to the consensus matrix A (cf. Appendix B.1). $\square$

Next, we bound the distance of the average $\bar{\mathbf{x}}^t$ from $\mathbf{x}^\star$, optimal solution of problem (2.1).

Lemma 2.9. Under Assumptions 2.1 and 2.6, it holds that

$$\|\bar{\mathbf{x}}^{t+1} - \mathbf{x}^\star\| \le \theta\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\| + \gamma\frac{L}{\sqrt{N}}\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\|, \tag{2.22}$$

where $\theta = \max(|1-\alpha\gamma/N|, |1-L\gamma/N|)$, with L and α being the Lipschitz constant of ∇f_i and the strong convexity parameter of f_i, respectively, $i \in \{1,\dots,N\}$.

Proof. Using (2.19), we can write

$$\begin{aligned}
\|\bar{\mathbf{x}}^{t+1} - \mathbf{x}^\star\| &= \|\bar{\mathbf{x}}^t - \gamma\,\bar{\mathbf{y}}^t - \mathbf{x}^\star\| \\
&\overset{(a)}{=} \left\|\bar{\mathbf{x}}^t - \gamma\frac{1}{N}\sum_{i=1}^N \nabla f_i(\bar{\mathbf{x}}^t) - \mathbf{x}^\star + \gamma\frac{1}{N}\sum_{i=1}^N \nabla f_i(\bar{\mathbf{x}}^t) - \gamma\,\bar{\mathbf{y}}^t\right\| \\
&\overset{(b)}{\le} \left\|\bar{\mathbf{x}}^t - \gamma\frac{1}{N}\sum_{i=1}^N \nabla f_i(\bar{\mathbf{x}}^t) - \mathbf{x}^\star\right\| + \gamma\left\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\bar{\mathbf{x}}^t) - \bar{\mathbf{y}}^t\right\| \\
&\overset{(c)}{\le} \theta\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\| + \gamma\left\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\bar{\mathbf{x}}^t) - \bar{\mathbf{y}}^t\right\| \\
&\overset{(d)}{\le} \theta\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\| + \gamma\frac{L}{\sqrt{N}}\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\|,
\end{aligned}$$

where in (a) we added and subtracted $\gamma/N \cdot \sum_{i=1}^N \nabla f_i(\bar{\mathbf{x}}^t)$, in (b) we used the triangle inequality, in (c) we exploited the convergence rate result for a gradient iteration applied to a smooth and strongly convex function[2] and (d) follows by the conservation property of the tracker (cf. (2.21)), the Lipschitz continuity of each ∇f_i and the algebraic property $\sum_{i=1}^N \|\xi_i\|_2 \le \sqrt{N}\|[\xi_1,\dots,\xi_N]^\top\|_2$. □

Finally, we provide an upper bound for the tracking error.

Lemma 2.10. Under Assumptions 2.1 and 2.6, it holds

$$\begin{aligned}
\|\mathbf{y}^{t+1} - \bar{\mathbf{y}}^{t+1}\mathbf{1}\| \le{}& (\sigma_A + \gamma L)\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| \\
&+ (L\|A - I\| + \gamma L^2\sqrt{N})\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| \qquad (2.23)\\
&+ \gamma L^2\sqrt{N}\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\|,
\end{aligned}$$

for all $t \ge 0$, where σ_A is the contraction factor associated to the consensus matrix A, I is the identity matrix and L is the Lipschitz constant of ∇f_i, $i \in \{1,\dots,N\}$.

[2] We recall that a (centralized) gradient iteration applied to the minimization of a L_φ-smooth and α_φ-strongly function $\varphi(z)$ satisfies (for a sufficiently small $\gamma > 0$) $\|z - \gamma\nabla\varphi(z) - z^\star\| \le \theta_\varphi\|z - z^\star\|$, where $\theta_\varphi = \max(|1-\alpha_\varphi\gamma|, |1-L_\varphi\gamma|)$ and $z^\star$ is the minimizer of φ.

Proof. Under Lipschitz continuity of ∇f, and using (2.20), it follows

$$\begin{aligned}
\|\mathbf{y}^{t+1} - \bar{\mathbf{y}}^{t+1}\mathbf{1}\| &\overset{(a)}{=} \Big\|A\mathbf{y}^t + \nabla f(\mathbf{x}^{t+1}) - \nabla f(\mathbf{x}^t) \\
&\qquad - \big(\bar{\mathbf{y}}^t + \frac{1}{N}\sum_{i=1}^{N} (\nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\mathbf{x}_i^t))\big)\mathbf{1}\Big\| \\
&\overset{(b)}{\leq} \|A\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + \left\|\left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\right)(\nabla f(\mathbf{x}^{t+1}) - \nabla f(\mathbf{x}^t))\right\| \\
&\overset{(c)}{\leq} \sigma_A\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + \left\|I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\right\| \|\nabla f(\mathbf{x}^{t+1}) - \nabla f(\mathbf{x}^t)\| \\
&\overset{(d)}{\leq} \sigma_A\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + L\|A\mathbf{x}^t - \gamma\mathbf{y}^t - \mathbf{x}^t\|,
\end{aligned}$$

where in (a) we used (2.18) and (2.20), in (b) we rearranged the terms and we used the triangle inequality, in (c) we used the contraction property of the consensus matrix A (cf. Appendix B.1) and the sub-multiplicativity of 2-norm and finally in (d) we used the fact that $\|I - \mathbf{1}\mathbf{1}^\top/N\| \leq 1$ and the Lipschitz continuity of ∇f together with the update law (2.17).

Next, we make further modifications on the terms as follows

$$\begin{aligned}
\|\mathbf{y}^{t+1} - \bar{\mathbf{y}}^{t+1}\mathbf{1}\| &\leq \sigma_A\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + L\|A\mathbf{x}^t - \gamma\mathbf{y}^t - \mathbf{x}^t\| \\
&\overset{(a)}{=} \sigma_A\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + L\|(A - I)(\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}) - \mathbf{y}^t\| \\
&\overset{(b)}{\leq} \sigma_A\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + L\|A - I\| \cdot \|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \gamma L\|\mathbf{y}^t\|,
\end{aligned}$$

where in (a) we added and subtracted $\bar{\mathbf{x}}^t$ and we exploited row stochasticity of A, and in (b) we used the sub-multiplicativity of 2-norm and the triangle inequality. Adding and subtracting $\bar{\mathbf{y}}^t$ and using the triangle inequality we can write $\|\mathbf{y}^t\| \leq \|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| + \|\bar{\mathbf{y}}^t\mathbf{1}\|$, which plugged into the last equation yields

$$\begin{aligned}
\|\mathbf{y}^{t+1} - \bar{\mathbf{y}}^{t+1}\mathbf{1}\| &\leq (\sigma_A + \gamma L)\|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| \\
&\qquad + L\|A - I\| \cdot \|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + \gamma L\|\bar{\mathbf{y}}^t\mathbf{1}\|.
\end{aligned} \tag{2.24}$$

Finally, let us manipulate the last term in (2.24) as

$$\begin{aligned}
\|\bar{\mathbf{y}}^t\mathbf{1}\| = N\|\bar{\mathbf{y}}^t\| &= \left\|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^t)\right\| \\
&\overset{(a)}{=} N\left\|\frac{1}{N}\sum_{i=1}^N (\nabla f_i(\mathbf{x}_i^t) - \nabla f_i(\mathbf{x}_i^\star))\right\| \\
&\overset{(b)}{\leq} L\sum_{i=1}^N \|\mathbf{x}_i^t - \mathbf{x}_i^\star\| \\
&\overset{(c)}{\leq} L\sqrt{N}\|\mathbf{x}^t - \mathbf{x}^\star\mathbf{1}\| \\
&\overset{(d)}{\leq} L\sqrt{N}\|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| + L\sqrt{N}\|\bar{\mathbf{x}}^t - \mathbf{x}^\star\|,
\end{aligned} \tag{2.25}$$

where in (a) we added $\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^\star) = 0$, in (b) we exploited the Lipschitz continuity of each ∇f_i, in (c) we used the algebraic property $\sum_{i=1}^N \|\xi_i\|_2 \leq \sqrt{N}\|[\xi_1, \dots, \xi_N]^\top\|_2$, and in (d) we added and subtracted $\bar{\mathbf{x}}^t\mathbf{1}$ and used the triangle inequality. Combining (2.24) with (2.25) the proof follows. □

The following theorem states the convergence result for Algorithm 2.

Theorem 2.11. Let Assumptions 2.1 and 2.6 hold and let the communication graph be undirected and connected. Then, there exists a constant $\bar{\gamma} \in (0, N/L)$ such that for all $\gamma \in (0, \bar{\gamma})$ the sequences of local solution estimates $\{\mathbf{x}_i^t\}_{t\geq 0}$, $i \in \{1, \dots, N\}$, generated by Algorithm 2 are asymptotically consensual to the optimal solution $\mathbf{x}^\star$ of problem (2.1), i.e.,

$$\lim_{t\to\infty} \|\mathbf{x}_i^t - \mathbf{x}^\star\| = 0, \tag{2.26}$$

for all $i \in \{1, \dots, N\}$. Moreover, the convergence rate is linear.[3]

Proof. The proof is based on showing a (strict) contraction property along the algorithmic evolution. Let us introduce the following vector

$$\mathbf{v}^t \triangleq \begin{bmatrix} \|\mathbf{x}^t - \bar{\mathbf{x}}^t\mathbf{1}\| \\ \|\mathbf{y}^t - \bar{\mathbf{y}}^t\mathbf{1}\| \\ \|\bar{\mathbf{x}}^t - \mathbf{x}^\star\| \end{bmatrix}.$$

[3] A (convergent) sequence $\{\mathbf{z}^t\}_{t\geq 0}$ is said to converge linearly (or geometrically) to $\mathbf{z}^\star$ if there exists $\rho \in (0, 1)$ such that $\|\mathbf{z}^{t+1} - \mathbf{z}^\star\| \leq \rho\|\mathbf{z}^t - \mathbf{z}^\star\|$, for all $t \geq 0$.

Then, combining the results given in Lemma 2.8, 2.9 and 2.10 it holds

$$\mathbf{v}^{t+1} \leq J(\gamma)\mathbf{v}^t, \tag{2.27}$$

where the matrix $J(\gamma)$ is defined as

$$J(\gamma) \triangleq \begin{bmatrix} \sigma_A & \gamma & 0 \\ \left(L\|A-I\| + \gamma L^2\sqrt{N}\right) & \sigma_A + \gamma L & \gamma L^2\sqrt{N} \\ \gamma\frac{L}{\sqrt{N}} & 0 & \theta \end{bmatrix}.$$

Recall that $\theta = \max(|1-\alpha\gamma/N|, |1-L\gamma/N|)$. Since $\alpha \leq L$ and $\gamma \leq N/L$, it follows that $\theta = 1 - \alpha\gamma/N$. Thus, we can express $J(\gamma)$ as the sum of two structured matrices as follows

$$J(\gamma) = \begin{bmatrix} \sigma_A & 0 & 0 \\ L\|A-I\| & \sigma_A & 0 \\ 0 & 0 & 1 \end{bmatrix} + \gamma \begin{bmatrix} 0 & 1 & 0 \\ L^2\sqrt{N} & 1 & L^2\sqrt{N} \\ \frac{L}{\sqrt{N}} & 0 & -\alpha/N \end{bmatrix}.$$

Being $\sigma_A < 1$ and due to the triangular structure of the left matrix, we can conclude that it has spectral radius equal to 1. Since the eigenvalues of a matrix are a continuous function of its entries, we can use a continuity argument to assert that for positive γ the spectral radius of $J(\gamma)$ becomes strictly less than 1 (see [150, Theorem 1] for a more comprehensive discussion). Hence, we have $\mathbf{v}^{t+1} \leq \rho\,\mathbf{v}^t$ with $\rho \in (0,1)$. Thus, $\|\mathbf{v}^t - [0,0,0]^\top\| \to 0$ as $t \to \infty$ with linear rate, and the proof follows. □

2.3 Variants and Extensions of the Basic Gradient Tracking

Several extensions of the gradient tracking scheme (described in Section 2.2) have been proposed in the literature. We present some of them without following their historical development but following a pure conceptual flow.

A first enhancement deals with optimization problems including both composite cost functions (i.e., with regularizers) and a common convex constraint. The main idea is to compute a feasible descent direction rather than a pure descent direction. Thus, let us consider a constrained cost-coupled optimization problem

$$\min_{\mathbf{x}\in X} \sum_{i=1}^{N} f_i(\mathbf{x}) + r(\mathbf{x}), \tag{2.28}$$

with r being a convex *regularizer* and X a convex constraint set. The modified algorithm reads as follows

$$\begin{aligned}
\Delta \mathbf{x}_i^t &= \operatorname*{argmin}_{\mathbf{x}_i \in X} \; (\mathbf{x}_i - \mathbf{x}_i^t)^\top (N \mathbf{y}_i^t) + \frac{\tau}{2}\|\mathbf{x}_i - \mathbf{x}_i^t\|^2 + \frac{r(\mathbf{x})}{N}, \\
\mathbf{x}_i^{t+1} &= \sum_{j \in \mathcal{N}_i} a_{ij} \Big((1-\beta)\mathbf{x}_j^t + \beta \Delta \mathbf{x}_j^t \Big), \\
\mathbf{y}_i^{t+1} &= \sum_{j \in \mathcal{N}_i} a_{ij}\, \mathbf{y}_j^t + \big(\nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\mathbf{x}_i^t)\big),
\end{aligned}$$

where $\tau > 0$ and $\beta \in (0, 1]$ are parameters to be suitably tuned. Notice that $N\mathbf{y}_i^t$ represents a local estimate of $\sum_{j=1}^N \nabla f_j(\mathbf{x}_j^t)$ that is used to build a linear approximation of $\sum_{j=1}^N f_j(\mathbf{x}_j^t)$ about the current iterate. Moreover, notice that $\Delta\mathbf{x}_i^t \in X$, so that, provided that $\mathbf{x}_j^t \in X$, then $\mathbf{x}_i^{t+1}$ stays feasible. This constrained version of the gradient tracking has been proposed and analyzed in [32, 33, 128, 127, 119, 97]. We notice that in these works, the authors consider a more general nonconvex optimization setting and propose more general approximation schemes than a simple linearization. Indeed, using successive convex approximations, the proposed distributed algorithms are able to solve also nonconvex instances of problem (2.28), which are of great interest in learning and estimation applications.

The gradient tracking has been extended also to time-varying and directed networks by means of the push-sum protocol (cf. Appendix B.2) in both the consensus and the tracking iterations. Formally, the algorithm reads

$$\begin{aligned}
\phi_i^{t+1} &= \sum_{j \in \mathcal{N}_i} b_{ij}^t\, \phi_i^t \\
\mathbf{x}_i^{t+1} &= \frac{1}{\phi_i^{t+1}} \left(\sum_{j \in \mathcal{N}_i} b_{ij}^t\, \phi_i^t \mathbf{x}_j^t - \gamma^t\, \mathbf{y}_i^t \right) \\
\mathbf{y}_i^{t+1} &= \frac{1}{\phi_i^{t+1}} \left(\sum_{j \in \mathcal{N}_i} b_{ij}^t\, \phi_i^t \mathbf{y}_j^t + \nabla f_i(\mathbf{x}_i^{t+1}) - \nabla f_i(\mathbf{x}_i^t) \right),
\end{aligned}$$

with $\phi_i^0 = 1$, for all $i \in \{1, \ldots, N\}$, and where the time-varying weights b_{ij}^t are entries of column stochastic matrices $B^t \in \mathbb{R}^{N \times N}$, for all $t \geq 0$. This extension has been studied in [32, 33, 128, 127, 119, 93, 150, 146, 94, 92]. Notice that the previous extensions have been combined in some of the mentioned works to design time-varying gradient algorithm for convex and nonconvex problems. Recently, a block-wise implementation of the gradient tracking algorithm has been proposed in [104, 105, 106].

2.4 Discussion and References

Early consensus-based algorithms for distributed optimization and estimation have been proposed and analyzed in [118, 55, 95, 24, 96, 72, 29, 54]. A push-sum version of the subgradient algorithm has been proposed in [89] to deal with time-varying networks. Extensions to the stochastic set-up are provided in [114, 90] A distributed algorithm using a constant step-size has been proposed in [121], with proved convergence rate $\mathcal{O}(1/t)$ (which can be strenghtened to linear for strongly convex problems). The algorithmic framework has been extended to regularized problems in [120], and a detailed convergence rate analysis has been proposed in [156]. Its extension to directed graphs is proposed in [145]. Distributed schemes to solve nonconvex optimization problems are proposed in [15, 130, 159].

As regards gradient tracking algorithms, the interested reader can find relevant up-to-date references in Section 2.2. Second-order approaches have been investigated in [141, 142, 79, 38]. Netwon-Raphson distributed approaches have been proposed and analyzed in [157, 137]. An extension to networks with packet loss is given in [16].

Distributed schemes working under asynchronous communication protocols are studied in [86, 125, 62, 63, 67, 71]. A randomized block-coordinate descent algorithm for convex optimization problems with linear constraints is proposed in [82]. In [143] an asynchronous distributed algorithm working also with communication delays is proposed.

As regards continous-time optimization, a purely primal approach is designed in [73]. A prediction-correction approach for online distributed optimization has been proposed in [124]. It is also worth mentioning the works in [139, 57, 129, 48, 115], where a control perspective is employed to analyze distributed optimization algorithms. A distributed scenario with a variable number of working nodes is proposed in [51]. A novel methodology to design continuous-time distributed optimization algorithms using techniques from geometric control theory is investigated in [37, 78].

Among the most recent contributions, a Frank-Wolfe decomposition approach for convex and nonconvex problems is analyzed in [138]. A distributed algorithm based on the proximal minimization is proposed in [75] to solve convex constrained problems. In [152], a distributed scheme using a Bregman penalization has been proposed. A distributed optimization algorithm for convex optimization with local inequality constraints has been studied in [149]. An asynchronous distributed algorithm with heterogeneous regularizations and normalizations is proposed in [49]. A specialized version of the distributed subgradient algorithm for convex feasibility problems, which allows for an infinite number of constraint sets, is presented in [41].

2.5 Numerical Example

In this section we provide a numerical study to show the behavior of the distributed optimization algorithms presented in this chapter.

We consider a network of $N = 30$ agents communicating over a fixed, undirected, connected graph generated according to an Erdős-Rényi random model with parameter $p = 0.2$. Agents are equipped with a doubly stochastic matrix built according to the Metropolis-Hastings rule [148], i.e.,

$$a_{ij} = \begin{cases} \frac{1}{\max\{d_i,d_j\}+1}, & \text{if } j \neq i \text{ and } (i,j) \in \mathcal{E}, \\ 1 - \sum_{j \in \mathcal{N}_i} \frac{1}{\max\{d_i,d_j\}+1}, & \text{if } j = i, \\ 0, & \text{otherwise.} \end{cases}$$

We focus on the logistic regression problem introduced in Section 1.3.2, where we suppose that each agent has $m_1 = \ldots = m_N = 10$ samples with feature space dimension $d = 5$. We generate the points $p_{i,j}$ according to a normal distribution with zero mean and variance equal to 2 and we generate the binary labels $\ell_{i,j}$ from a standard Bernoulli distribution. Agents must agree on the optimal solution of problem (1.6), recalled here

$$\min_{w,b} \sum_{i=1}^{N} \underbrace{\left(\sum_{j=1}^{m_i} \log\left[1 + e^{-(w^\top p_{i,j}+b)\ell_{i,j}}\right] + \frac{C}{2N}\|w\|^2 \right)}_{f_i(w,b)}.$$

The regularization parameter C is assumed to be equal to 0.01. We compare the distributed subgradient method (cf. Section 2.1), with diminishing step-size $\gamma^t = (1/t)^{0.8}$, and the gradient tracking algorithm (cf. Section 2.2), with constant step-size $\gamma = 10^{-3}$.

In Figure 2.1 we compare the convergence rate of Algorithm 1 and Algorithm 2. That is, we plot the absolute value of the difference between the optimal cost $f^\star$ and the sum of local costs $\sum_{i=1}^{N} f_i(\mathbf{x}_i^t)$. From the theoretical analysis, the cost error of both algorithms is known to asymptotically converge to zero. However, the gradient tracking algorithm has a linear convergence rate and converges more quickly than the distributed subgradient method (see Figure 2.1).

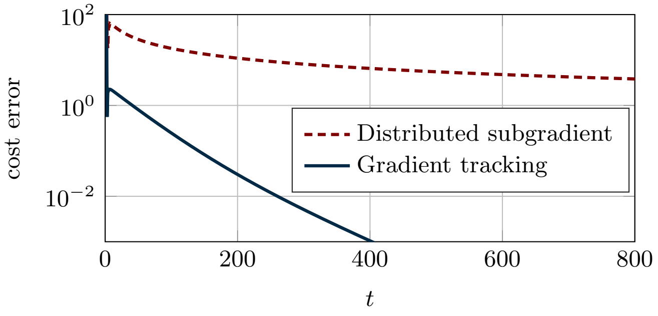


Figure 2.1: Evolution of the cost error for the distributed subgradient method and for the gradient tracking algorithm.

In Figure 2.2 and 2.3, we show the total consensus error of the local solution estimates (for both algorithms) and of the gradient trackers (for the gradient tracking algorithm), respectively.

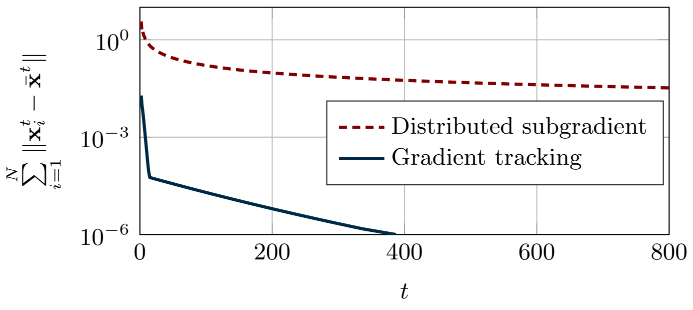


Figure 2.2: Evolution of the total consensus error of the local solution estimates $\mathbf{x}_i^t$ for the distributed subgradient method and for the gradient tracking algorithm.

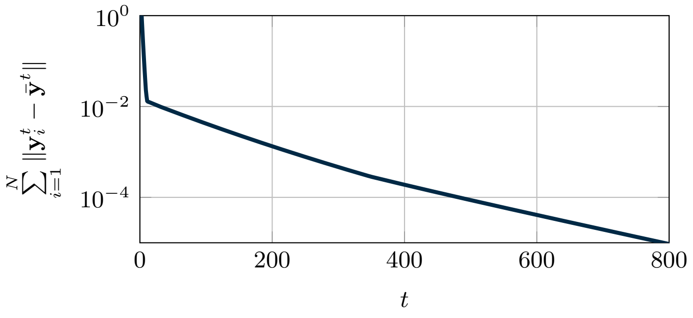


Figure 2.3: Evolution of the total consensus error of agents' gradient trackers in the gradient tracking algorithm.

3

Distributed Dual Methods

In this chapter we describe distributed optimization methods based on Lagrangian approaches. We start by discussing an illustrative example and then we present two relevant duality forms to show how duality can be exploited to reformulate cost-coupled problems as constraint-coupled problems and vice versa. We describe algorithms for cost-coupled problems based on a decomposition technique known as *dual decomposition* and on the Alternating Direction Method of Multipliers (ADMM). Then, we illustrate duality-based approaches to solve constraint-coupled problems. To conclude, we give an up-to-date set of references and we provide numerical examples to highlight the main features of the discussed algorithms.

3.1 Fenchel Duality and Graph Duality

In this section we show how a cost-coupled optimization problem can be manipulated to obtain alternative (decoupled) problem formulations that are amenable for distributed computation. First, we present a simplified scenario with two agents to illustrate how duality can be exploited in designing a distributed optimization algorithm. Then, we recall a classical duality form known as Fenchel duality (see [12]), that paved the way for a number of parallel algorithms. Finally, we introduce an alternative and effective approach, that we term graph duality, tailored for the distributed framework.

Consider a cost-coupled problem

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

where, for all $i \in \{1,\dots,N\}$, the cost function f_i is convex and the constraint set X_i is convex and bounded. These regularity assumptions are standard and guarantee that dual methods apply, i.e., that strong duality holds (cf. Appendix A.3). We will denote by $f^\star$ the optimal cost of problem (3.1).

3.1.1 Two-Agent Example

We start by considering a simple "network" of 2 agents and informally discuss how duality allows for a suitable decomposition of a cost-coupled problem. All the technical details will be provided in the forthcoming sections.

We assume that both agents cooperate to solve the cost-coupled optimization problem

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^d} \ & f_1(\mathbf{x}) + f_2(\mathbf{x}) \\ \text{subj. to } & \mathbf{x} \in X_1 \cap X_2, \end{aligned} \tag{3.2}$$

where $f_1, f_2 : \mathbb{R}^d \to \mathbb{R}$ and $X_1, X_2 \subseteq \mathbb{R}^d$. Recall that for such cost-coupled set-up, each agent is assumed to know only its own cost function and constraint (e.g., agent 1 knows only f_1 and X_1).

The aim is to decompose problem (3.2) by exploiting Lagrangian duality. Specifically, we would like to obtain two symmetric subproblems so that each agent can solve its subproblem independently. To this end, we recast problem (3.2) into an equivalent formulation by introducing two copies, say $\mathbf{x}_1$ and $\mathbf{x}_2$, of the decision variable $\mathbf{x}$ and a coherence constraint to obtain

$$\begin{aligned} \min_{\mathbf{x}_1,\mathbf{x}_2} \;& f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) \\ \text{subj. to } & \mathbf{x}_1 \in X_1,\ \mathbf{x}_2 \in X_2, \\ & \mathbf{x}_1 = \mathbf{x}_2. \end{aligned} \tag{3.3}$$

This reformulation exhibits a convenient structure since the cost function of each agent depends only on its copy of the decision variable, while the coupling in the problem is due only to the coherence constraint $\mathbf{x}_1 = \mathbf{x}_2$. Now we write the dual of problem (3.3) (cf. Appendix A.3). Let us introduce the Lagrangian of (3.3), i.e.,

$$\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}) = f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \boldsymbol{\lambda}^\top(\mathbf{x}_1 - \mathbf{x}_2),$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ is the multiplier associated to the constraint $\mathbf{x}_1 = \mathbf{x}_2$. As it will be clear from the forthcoming discussion, the presence of a single $\boldsymbol{\lambda}$ in $\mathcal{L}$ does not allow for a symmetric decomposition. Thus, let us follow an alternative approach, more suited for distributed computation. Formally, we add another, redundant constraint and rewrite (3.3) as

$$\begin{aligned} \min_{\mathbf{x}_1,\mathbf{x}_2} \;& f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) \\ \text{subj. to } & \mathbf{x}_1 \in X_1,\ \mathbf{x}_2 \in X_2, \\ & \mathbf{x}_1 = \mathbf{x}_2, \\ & \mathbf{x}_2 = \mathbf{x}_1, \end{aligned} \tag{3.4}$$

which is trivially equivalent to problem (3.3). For this problem, the Lagrangian becomes

$$\begin{aligned} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}) &= f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \boldsymbol{\lambda}_{12}^\top(\mathbf{x}_1 - \mathbf{x}_2) + \boldsymbol{\lambda}_{21}^\top(\mathbf{x}_2 - \mathbf{x}_1) \\ &\overset{(a)}{=} f_1(\mathbf{x}_1) + (\boldsymbol{\lambda}_{12} - \boldsymbol{\lambda}_{21})^\top \mathbf{x}_1 \\ &\quad + f_2(\mathbf{x}_2) + (\boldsymbol{\lambda}_{21} - \boldsymbol{\lambda}_{12})^\top \mathbf{x}_2, \end{aligned} \tag{3.5}$$

where $\boldsymbol{\lambda}_{12}$ and $\boldsymbol{\lambda}_{21}$ are the multipliers associated to the constraints $\mathbf{x}_1 = \mathbf{x}_2$ and $\mathbf{x}_2 = \mathbf{x}_1$ respectively, and in (a) we use the problem symmetry to rearrange $\mathcal{L}$ in two similar terms, each one depending only on a single primal variable, i.e., on $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively. The dual function of problem (3.4) is obtained by minimizing the Lagrangian (3.5) with respect to the primal variables. Formally,

$$\begin{aligned} q(\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}) &= \inf_{\mathbf{x}_1 \in X_1, \mathbf{x}_2 \in X_2} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}) \\ &= \underbrace{\min_{\mathbf{x}_1 \in X_1} \big(f(\mathbf{x}_1) + (\boldsymbol{\lambda}_{12} - \boldsymbol{\lambda}_{21})^\top \mathbf{x}_1\big)}_{q_1(\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21})} + \underbrace{\min_{\mathbf{x}_2 \in X_2} \big(f(\mathbf{x}_2) + (\boldsymbol{\lambda}_{21} - \boldsymbol{\lambda}_{12})^\top \mathbf{x}_2\big)}_{q_2(\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21})}. \end{aligned}$$

Finally, we can pose the dual problem as

$$\max_{\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}} q(\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}) = \max_{\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}} q_1(\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}) + q_2(\boldsymbol{\lambda}_{12}, \boldsymbol{\lambda}_{21}). \tag{3.6}$$

Under suitable regularity assumption on the primal problem (3.2), problem (3.6) has the same optimal cost. Thus, by solving (3.6), a dual optimal solution can be exploited to recover a primal optimal solution. In Section 3.1.3, we described the extended approach for a general set-up with N agents.

The *distributed dual decomposition* algorithm consists of an iterative procedure to solve problem (3.6) by means of a subgradient algorithm (cf. Appendix A.1), and to obtain ultimately a solution of the original problem (3.2). The choice of solving (3.6) with such algorithm is convenient since a subgradient of the dual function[1] at a given $(\bar{\boldsymbol{\lambda}}_{12}, \bar{\boldsymbol{\lambda}}_{21})$ can be computed, in a distributed way, as

$$\widetilde{\nabla} q(\bar{\boldsymbol{\lambda}}_{12}, \bar{\boldsymbol{\lambda}}_{21}) = \begin{bmatrix} \widetilde{\nabla}_{\boldsymbol{\lambda}_{12}} q(\bar{\boldsymbol{\lambda}}_{12}, \bar{\boldsymbol{\lambda}}_{21}) \\ \widetilde{\nabla}_{\boldsymbol{\lambda}_{21}} q(\bar{\boldsymbol{\lambda}}_{12}, \bar{\boldsymbol{\lambda}}_{21}) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2 \\ \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1 \end{bmatrix},$$

where

$$\begin{aligned} \bar{\mathbf{x}}_1 &\in \operatorname*{argmin}_{\mathbf{x}_1 \in X_1} f_1(\mathbf{x}_1) + (\bar{\boldsymbol{\lambda}}_{12} - \bar{\boldsymbol{\lambda}}_{21})^\top \mathbf{x}_1, \\ \bar{\mathbf{x}}_2 &\in \operatorname*{argmin}_{\mathbf{x}_2 \in X_2} f_2(\mathbf{x}_2) + (\bar{\boldsymbol{\lambda}}_{21} - \bar{\boldsymbol{\lambda}}_{12})^\top \mathbf{x}_2. \end{aligned}$$

We assume that agent 1 maintains and updates $\mathbf{x}_1$ and $\boldsymbol{\lambda}_{12}$, while agent 2 maintains and updates $\mathbf{x}_2$ and $\boldsymbol{\lambda}_{21}$. At the beginning, they

[1] Notice that here we are slightly abusing terminology. Indeed, subgradients are defined for convex functions, while the dual function q is concave. Here, the notation $\widetilde{\nabla} q$ stands for the opposite of a subgradient of $-q$.

initialize $\boldsymbol{\lambda}_{12}^0$ and $\boldsymbol{\lambda}_{21}^0$ to arbitrary values. Then, at each iteration $t \geq 0$ of the algorithm, agents exchange their current value of $\boldsymbol{\lambda}_{12}^t$ and $\boldsymbol{\lambda}_{21}^t$ and compute a local estimate of the solution as

$$\begin{aligned} \mathbf{x}_1^{t+1} &\in \operatorname*{argmin}_{\mathbf{x}_1 \in X_1} \; f_1(\mathbf{x}_1) + (\boldsymbol{\lambda}_{12}^t - \boldsymbol{\lambda}_{21}^t)^\top \mathbf{x}_1, \\ \mathbf{x}_2^{t+1} &\in \operatorname*{argmin}_{\mathbf{x}_2 \in X_2} \; f_2(\mathbf{x}_2) + (\boldsymbol{\lambda}_{21}^t - \boldsymbol{\lambda}_{12}^t)^\top \mathbf{x}_2. \end{aligned} \tag{3.7}$$

Then, they exchange the updated value of $\mathbf{x}_1^{t+1}$ and $\mathbf{x}_2^{t+1}$ to adjust their local dual variable as

$$\begin{aligned} \boldsymbol{\lambda}_{12}^{t+1} &= \boldsymbol{\lambda}_{12}^t + \gamma^t \, \widetilde{\nabla}_{\boldsymbol{\lambda}_{12}} q(\boldsymbol{\lambda}_{12}^t, \boldsymbol{\lambda}_{21}^t) = \boldsymbol{\lambda}_{12}^t + \gamma^t \, (\mathbf{x}_1^{t+1} - \mathbf{x}_2^{t+1}), \\ \boldsymbol{\lambda}_{21}^{t+1} &= \boldsymbol{\lambda}_{21}^t + \gamma^t \, \widetilde{\nabla}_{\boldsymbol{\lambda}_{21}} q(\boldsymbol{\lambda}_{12}^t, \boldsymbol{\lambda}_{21}^t) = \boldsymbol{\lambda}_{21}^t + \gamma^t \, (\mathbf{x}_2^{t+1} - \mathbf{x}_1^{t+1}), \end{aligned} \tag{3.8}$$

where γ^t denotes the step-size of the gradient method. An illustration of how communication and computation interleave is shown in Figure 3.1.

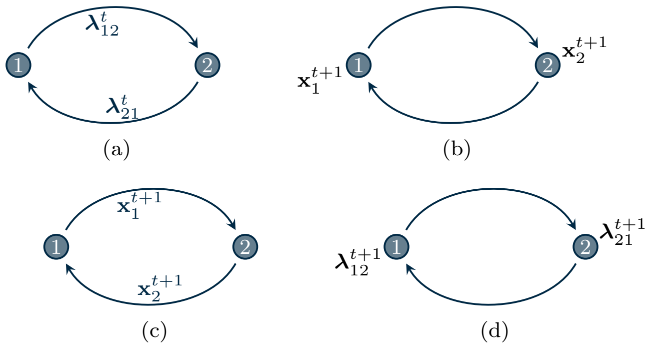

$\boldsymbol{\lambda}_{12}^t$
1
2
$\boldsymbol{\lambda}_{21}^t$
(a)
$\mathbf{x}_1^{t+1}$
$\mathbf{x}_2^{t+1}$
(b)
$\mathbf{x}_1^{t+1}$
$\mathbf{x}_2^{t+1}$
(c)
$\boldsymbol{\lambda}_{12}^{t+1}$
$\boldsymbol{\lambda}_{21}^{t+1}$
(d)

Figure 3.1: Distributed dual decomposition algorithm for the 2-agent case. In (a) agents exchange their dual variables to update in (b) the primal variables, cf. (3.7). Then, the updated primal variables are communicated in (c) to perform the dual update in (d), cf. (3.8).

In Section 3.2 we will present and analyze the general case with N agents and prove that the local solution estimates are asymptotically consensual and converge to an optimal solution of the primal problem.

3.1.2 Fenchel Duality

A classical approach to manipulate problem (3.1) consists in writing its Fenchel dual [9]. To this end, let us introduce copies $\mathbf{x}_i \in \mathbb{R}^d$ of the optimization variable $\mathbf{x}$ and an auxiliary variable $\mathbf{z} \in \mathbb{R}^d$ needed to enforce coherence among all the copies. Then, problem (3.1) can be equivalently recast as

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{z}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to} \quad & \mathbf{x}_i \in X_i, \qquad i \in \{1,\dots,N\} \\ & \mathbf{x}_i = \mathbf{z}, \qquad i \in \{1,\dots,N\}. \end{aligned} \tag{3.9}$$

The Fenchel-dual problem of (3.1) is defined as the (standard) dual of (3.9). To this end, consider the Lagrangian function of (3.9), i.e.,

$$\mathcal{L}(\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{z},\boldsymbol{\lambda}_1,\dots,\boldsymbol{\lambda}_N) = \sum_{i=1}^{N} (f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^\top(\mathbf{x}_i - \mathbf{z})).$$

The minimization of $\mathcal{L}$ with respect to the primal variables gives the dual function

$$\begin{aligned} q(\boldsymbol{\lambda}_1,\dots,\boldsymbol{\lambda}_N) &= \inf_{\mathbf{x}_1\in X_1,\dots,\mathbf{x}_N\in X_N,\mathbf{z}} \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^\top(\mathbf{x}_i - \mathbf{z})\Big) \\ &= \sum_{i=1}^{N} \inf_{\mathbf{x}_i\in X_i} \Big(f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^\top \mathbf{x}_i\Big) + \inf_{\mathbf{z}} \Big(\sum_{i=1}^{N} \boldsymbol{\lambda}_i\Big)^\top \mathbf{z} \\ &= \begin{cases} \sum_{i=1}^{N} \inf_{\mathbf{x}_i\in X_i} \Big(f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^\top \mathbf{x}_i\Big) & \text{if } \sum_{i=1}^{N} \boldsymbol{\lambda}_i = \mathbf{0} \\ -\infty & \text{otherwise.} \end{cases} \end{aligned}$$

Then, the Fenchel-dual problem of (3.1) is given by the maximization of q over its domain, i.e.,

$$\begin{aligned} \max_{\boldsymbol{\lambda}_1,\dots,\boldsymbol{\lambda}_N} \quad & \sum_{i=1}^{N} q_i(\boldsymbol{\lambda}_i) \\ \text{subj. to} \quad & \sum_{i=1}^{N} \boldsymbol{\lambda}_i = \mathbf{0}, \end{aligned} \tag{3.10}$$

where each q_i is defined as

$$q_i(\boldsymbol{\lambda}_i) \triangleq \min_{\mathbf{x}_i\in X_i} \; f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^\top \mathbf{x}_i.$$

Problems in the form (3.10) are often referred to as *resource allocation* problems. We point out that (3.10) has a constraint-coupled structure, similar to problem (1.3) in Section 1.2.3. A (centralized) projected gradient method applied to (3.10) reads as follows,

$$\mathbf{x}_i^{t+1} \in \operatorname*{argmin}_{\mathbf{x}_i \in X_i} \; f_i(\mathbf{x}_i) + \left(\boldsymbol{\lambda}_i^t\right)^\top \mathbf{x}_i, \qquad i \in \{1, \dots, N\}, \tag{3.11a}$$

$$\begin{bmatrix} \boldsymbol{\lambda}_1^{t+1} \\ \vdots \\ \boldsymbol{\lambda}_N^{t+1} \end{bmatrix} = \mathcal{P}_{\mathcal{D}} \left(\begin{bmatrix} \boldsymbol{\lambda}_1^t + \gamma\, \mathbf{x}_1^{t+1} \\ \vdots \\ \boldsymbol{\lambda}_N^t + \gamma\, \mathbf{x}_N^{t+1} \end{bmatrix} \right), \tag{3.11b}$$

where $\mathcal{D} = \{(\boldsymbol{\lambda}_1, \dots \boldsymbol{\lambda}_N) \mid \sum_{i=1}^N \boldsymbol{\lambda}_i = \mathbf{0}\}$ and $\mathcal{P}_{\mathcal{D}}$ denotes the Euclidean projection onto $\mathcal{D}$. We assume that the algorithm is initialized such that $(\boldsymbol{\lambda}_1^0, \dots \boldsymbol{\lambda}_N^0) \in \mathcal{D}$, e.g., $\boldsymbol{\lambda}_i = \mathbf{0}$ for all $i \in \{1, \dots, N\}$. The projection step (3.11b) admits the following explicit expression

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{\lambda}_1^{t+1} \\ \vdots \\ \boldsymbol{\lambda}_N^{t+1} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\lambda}_1^t + \gamma\, \mathbf{x}_1^{t+1} \\ \vdots \\ \boldsymbol{\lambda}_N^t + \gamma\, \mathbf{x}_N^{t+1} \end{bmatrix} - \begin{bmatrix} \frac{1}{N} \sum_{i=1}^N \left(\boldsymbol{\lambda}_i^t + \gamma\, \mathbf{x}_i^{t+1}\right) \\ \vdots \\ \frac{1}{N} \sum_{i=1}^N \left(\boldsymbol{\lambda}_i^t + \gamma\, \mathbf{x}_i^{t+1}\right) \end{bmatrix} \\
&\overset{(a)}{=} \begin{bmatrix} \boldsymbol{\lambda}_1^t + \gamma \left(\mathbf{x}_1^{t+1} - \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^{t+1} \right) \\ \vdots \\ \boldsymbol{\lambda}_N^t + \gamma \left(\mathbf{x}_N^{t+1} - \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i^{t+1} \right) \end{bmatrix},
\end{aligned}$$

where in (a) we exploited the (recursive) feasibility of the previous iterate $(\boldsymbol{\lambda}_1^t, \dots, \boldsymbol{\lambda}_N^t)$.

Algorithm (3.11) is also known as *parallel dual decomposition.* Notice that we used properties of dual subgradients involving the local primal minimizers to write the dual update (cf. Appendix A.3). Figure 3.2 shows the algorithmic flow of parallel dual decomposition.

Notice that problem (3.9) can also be solved using ADMM (cf. Appendix A.4). The formal updates can be derived as done for the parallel dual decomposition by considering the so-called *augmented* Lagrangian. It can be shown (see [18]) that the resulting algorithm is

$$\mathbf{x}_i^{t+1} = \operatorname*{argmin}_{\mathbf{x}_i \in X_i} \; f_i(\mathbf{x}_i) + \left(\boldsymbol{\lambda}_i^t\right)^\top \mathbf{x}_i + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}^t\|^2, \quad \forall\, i \tag{3.12a}$$

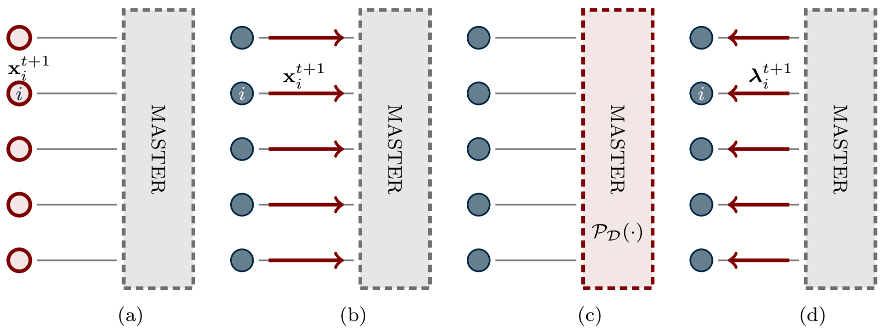


Figure 3.2: Algorithmic evolution of parallel dual decomposition: in (a) each node updates its primal variable according to (3.11a) and sends it to the master node (b). Then, in (c) the master node performs the projection of the dual variables as in (3.11b) and sends them back to the nodes in (d).

$$\mathbf{z}^{t+1} = \frac{1}{\rho}\sum_{i=1}^{N} \boldsymbol{\lambda}_i^t + \sum_{i=1}^{N} \mathbf{x}_i^{t+1} \tag{3.12b}$$

$$\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t + \rho\,(\mathbf{x}_i^{t+1} - \mathbf{z}^{t+1}), \qquad \forall\, i, \tag{3.12c}$$

where ρ is the positive penalty parameter of the augmented Lagrangian. It is worth noting that algorithm (3.12) enjoys a parallel structure similarly to the dual decomposition case.

3.1.3 Graph Duality

A powerful method to decouple a cost-coupled problem (3.1) into a convenient structure, amenable to distributed computation, is to introduce suitable graph-induced constraints, that result into an appropriate dual problem. We term this methodology *graph duality* to stress that it combines the classical duality theory with the network structure. Indeed, the resulting dual problem heavily depends on the specific network as will be detailed next. The method that we now formalize is the general form of the approach used in Section 3.1.1

Let a fixed, undirected and connected graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$ be given, then we define the $\mathcal{G}$-dual of (3.1) as follows. Introduce N copies, say $\mathbf{x}_1, \dots, \mathbf{x}_N$, of the decision variable $\mathbf{x}$ and coherence constraints of the copies matching the graph structure, i.e., $\mathbf{x}_i = \mathbf{x}_j$ for all $(i, j) \in \mathcal{E}$.

Then, problem (3.1) becomes

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to} \quad & \mathbf{x}_i \in X_i, \qquad i \in \{1,\dots,N\} \\ & \mathbf{x}_i = \mathbf{x}_j, \qquad (i,j) \in \mathcal{E}. \end{aligned} \tag{3.13}$$

Being the graph $\mathcal{G}$ connected, the equivalence of problems (3.1) and (3.13) is guaranteed.

Let $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^S$ be the multiplier associated to the constraint $\mathbf{x}_i = \mathbf{x}_j$, then the Lagrangian of (3.13) is

$$\mathcal{L}(\mathbf{x}_1,\dots,\mathbf{x}_N,\boldsymbol{\Lambda}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i) + \sum_{i=1}^{N} \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ij}^\top(\mathbf{x}_i - \mathbf{x}_j), \tag{3.14}$$

where the variable $\boldsymbol{\Lambda}$ stacks all the $|\mathcal{E}|$ multipliers $\boldsymbol{\lambda}_{ij}$.

Notice that, being the communication graph undirected, for each term $\boldsymbol{\lambda}_{ij}^\top(\mathbf{x}_i - \mathbf{x}_j)$ in (3.14) there is also a symmetric counterpart $\boldsymbol{\lambda}_{ji}^\top(\mathbf{x}_j - \mathbf{x}_i)$. Thus, the Lagrangian (3.14) can be rearranged so as to isolate the primal variables $\mathbf{x}_i$, $i \in \{1,\dots,N\}$, as

$$\mathcal{L}(\mathbf{x}_1,\dots,\mathbf{x}_N,\boldsymbol{\Lambda}) = \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}) \Big).$$

At this point, the dual function of (3.13) is obtained by minimizing the Lagrangian $\mathcal{L}$ with respect to the primal variables, leading to a separable function. Finally, the $\mathcal{G}$-dual of (3.1) is the (standard) dual of (3.13), which is given by

$$\max_{\boldsymbol{\Lambda}} \, q(\boldsymbol{\Lambda}) = \max_{\boldsymbol{\Lambda}} \sum_{i=1}^{N} q_i(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{(i,j)\in\mathcal{E}}), \tag{3.15}$$

where the i-th term q_i of the dual function q is defined as

$$q_i(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{(i,j)\in\mathcal{E}}) = \min_{\mathbf{x}_i \in X_i} f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji}),$$

for all $i \in \{1,\dots,N\}$. We notice that problem (3.15) exhibits interesting features for a distributed computation framework. First, it is an unconstrained optimization problem with cost function expressed, similarly to the starting problem, as the sum of local terms q_i. However, differently from the original problem (3.13), in the $\mathcal{G}$-dual (3.15) the i-th cost function depends only on the variables of agent i and of its neighbors, rather than on the entire stack of decision vectors. In Section 3.2, we will derive a distributed algorithm that exploits the special structure of problem (3.15), known in the literature as partitioned optimization (cf. Remark 1.2).

3.2 Distributed Dual Decomposition for Cost-Coupled Problems

In this section, we review an algorithm, known as distributed dual decomposition, that relies on duality to solve cost-coupled problems in a distributed way. Decomposition techniques based on duality have been introduced in [12, 109, 154]. Typically, they are used to obtain parallel algorithms to speed-up the computation. However, the distributed extension of those techniques are only partially discussed in the mentioned references, while in the following we provide a comprehensive and constructive analysis for this scenario.

We consider N agents in a network that want to cooperatively solve a cost-coupled problem (3.1) (cf. Section 1.2.1) that satisfies the following regularity properties.

Assumption 3.1. For all $i \in \{1, \ldots, N\}$, each f_i is a convex function and each X_i is a compact, convex set. Moreover, there exists a vector $\mathbf{x}$ such that $\mathbf{x} \in \operatorname{relint} X_i$[2], for all $i \in \{1, \ldots, N\}$. □

The latter part of Assumption 3.1 is known in the literature as Slater's constraint qualification, and is a sufficient condition to ensure that strong duality holds.

Agent i maintains a primal solution estimate $\mathbf{x}_i^t$, and dual solution estimates $\boldsymbol{\lambda}_{ij}^t, j \in \mathcal{N}_i$. The distributed dual decomposition algorithm is based on a subgradient method applied to the $\mathcal{G}$-dual of (3.1) (see Section 3.1.3), i.e.,

$$\max_{\boldsymbol{\Lambda}} \sum_{i=1}^{N} q_i(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}). \tag{3.16}$$

[2] Given a set $X \subset \mathbb{R}^d$, we denote by $\operatorname{relint} X$ its relative interior.

A subgradient of the dual function $q(\mathbf{\Lambda})$ at a given $\bar{\mathbf{\Lambda}}$ (stacking all the $\bar{\boldsymbol{\lambda}}_{ij}$) can be computed in a distributed way as follows. The component of $\widetilde{\nabla} q$ corresponding to the variable $\boldsymbol{\lambda}_{ij}$ is equal to (cf. Appendix (A.3))

$$\widetilde{\nabla}_{\boldsymbol{\lambda}_{ij}} q(\bar{\mathbf{\Lambda}}) = \bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j,$$

where $\bar{\mathbf{x}}_i$ is computed as

$$\bar{\mathbf{x}}_i \in \operatorname*{argmin}_{\mathbf{x}_i \in X_i} f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \sum_{j \in \mathcal{N}_i} (\bar{\boldsymbol{\lambda}}_{ij} - \bar{\boldsymbol{\lambda}}_{ji}),$$

and, consistently, for $\bar{\mathbf{x}}_j$. Due to the sparse computation of dual subgradients, a subgradient method applied to the $\mathcal{G}$-dual of (3.1) turns out to be a distributed algorithm. Formally, each agent i initializes $\boldsymbol{\lambda}_{ij}^t$ for $j \in \mathcal{N}_i$ to any vector in $\mathbb{R}^d$. At each iteration t, each agent i collects from its neighbors $j \in \mathcal{N}_i$ the updated dual variables $\boldsymbol{\lambda}_{ji}^t$ and performs a primal minimization

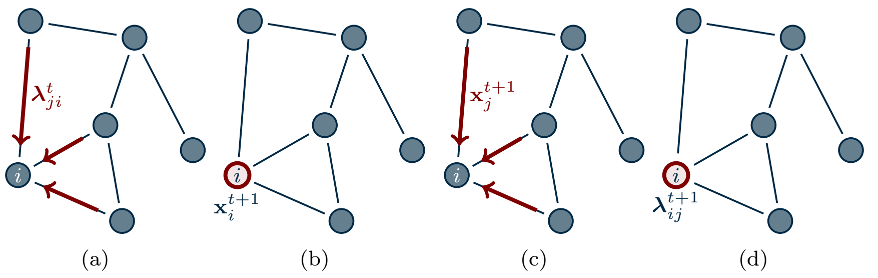
$$\mathbf{x}_i^{t+1} \in \operatorname*{argmin}_{\mathbf{x}_i \in X_i} \; f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t).$$

Then, agents exchange their updated primal solution estimates and perform a subgradient method step on the dual variables according to

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t + \gamma^t \,(\mathbf{x}_i^{t+1} - \mathbf{x}_j^{t+1}), \qquad j \in \mathcal{N}_i,$$

where γ^t is the step-size sequence.

Figure 3.3 shows the algorithmic flow of the distributed dual decomposition while the following table (Algorithm 3) summarizes the algorithm from the perspective of each agent i.



$\boldsymbol{\lambda}_{ji}^t$
i
$\mathbf{x}_i^{t+1}$
$\mathbf{x}_j^{t+1}$
$\boldsymbol{\lambda}_{ij}^{t+1}$
(a)
(b)
(c)
(d)

Figure 3.3: Algorithmic evolution of distributed dual decomposition: in (a) each node receives the dual variables from its neighbors. In (b), the local primal variable is updated according to (3.17). Then, in (c) the primal variables are broadcast to neighbors to allow in (d) for the dual updates (3.18).

Algorithm 3 Distributed Dual Decomposition

```
**Initialization**: $\boldsymbol{\lambda}_{ij}^0$ for all $j \in \mathcal{N}_i$
**Evolution**: for $t = 0, 1, \ldots$
  **Gather** $\boldsymbol{\lambda}_{ji}^t$ from neighbors $j \in \mathcal{N}_i$
  **Compute**
      $\mathbf{x}_i^{t+1} \in \operatorname*{argmin}_{\mathbf{x}_i \in X_i} \; f_i(\mathbf{x}_i) + \mathbf{x}_i^\top \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$   (3.17)
  **Gather** $\mathbf{x}_j^{t+1}$ from neighbors $j \in \mathcal{N}_i$
  **Update** for all $j \in \mathcal{N}_i$
      $\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t + \gamma^t \, (\mathbf{x}_i^{t+1} - \mathbf{x}_j^{t+1})$   (3.18)
```

Next, we provide the convergence result for Algorithm 3.

Theorem 3.2. Let Assumption 3.1 hold. Moreover, let the communication graph be undirected and connected and let the step-size γ^t satisfy Assumption 2.2. Then, the dual variable sequence $\{\boldsymbol{\Lambda}^t\}_{t \geq 0}$ generated by Algorithm 3 satisfies

$$\lim_{t \to \infty} q(\boldsymbol{\Lambda}^t) = f^\star,$$

where $f^\star$ is the optimal cost of problem (3.1).

Proof (Sketch). The proof heavily relies on the constructive derivation we carried out in this section. We have proven that the distributed dual decomposition algorithm is a subgradient method iteration on the $\mathcal{G}$-dual (3.16). Since the primal cost functions f_i are convex and the local sets X_i are compact, it is possible to show that the dual function q has bounded subgradients. Thus, by Proposition A.2, and since the dual function q is concave, every limit point of $\{\mathbf{\Lambda}^t\}_{t\geq 0}$ is an optimal solution of problem (3.16). Therefore, by continuity of q and by strong duality, it holds

$$\lim_{t\to\infty} q(\mathbf{\Lambda}^t) = f^\star.$$

$\square$

Notice that nothing can be said about the convergence of the primal sequence $\{\mathbf{x}_i^t\}_{t\geq 0}$ generated by Algorithm 3. In fact, due to the lack of strict convexity of the cost functions, there is no guarantee of feasibilty of the solutions retrieved by the Lagrangian minimization. This problem has been addressed by introducing averaging mechanisms, i.e., let the sequence $\{\widehat{\mathbf{x}}_i^t\}_{t\geq 0}$ be defined as $\widehat{\mathbf{x}}_i^t = 1/t\sum_{\tau=0}^t \mathbf{x}_i^\tau$, for all t. Then, it holds

$$\begin{aligned} &\lim_{t\to\infty} \sum_{i=1}^N f_i(\widehat{\mathbf{x}}_i^t) = f^\star, \\ &\lim_{t\to\infty} \|\widehat{\mathbf{x}}_i^t - \mathbf{x}^\star\| = 0, \qquad i \in \{1,\ldots,N\}, \end{aligned}$$

where $\mathbf{x}^\star$ and $f^\star$ denote an optimal solution and the optimal cost of problem (3.25), respectively.

Remark 3.1. If each cost function f_i in problem (3.1) is strongly convex then it is possible to improve the result. Specifically, under primal strong convexity the dual function q becomes smooth (i.e., differentiable with Lipschitz continuous gradient) so that a *gradient* method with constant step-size can be applied to solve the dual problem (3.16). Moreover, since strong convexity implies strict convexity, also primal convergence can be established, i.e., $\lim_{t\to\infty}\|\mathbf{x}_i^t - \mathbf{x}^\star\| = 0$ for all i with $\mathbf{x}^\star$ the optimal solution of (3.1). This follows since the Lagrangian minimization admits a unique solution at each iteration t. $\square$

As for the rate of convergence of the dual iterates, the algorithm directly inherits the convergence rate of the standard subgradient method, which is sublinear. If more regular problems are considered (e.g., strongly convex cost functions), then the dual function becomes smooth, therefore the linear convergence rate of gradient method is obtained.

Remark 3.2. Distributed dual decomposition can be also applied to partitioned optimization problems (cf. Remark 1.2). To efficiently exploit the partitioned structure of the problem, one can work on copies of the relevant portions of the global decision vector. This gives rise to tailored distributed dual decomposition algorithms, see, e.g., [22, 98]. The same procedure has been employed for distributed ADMM (cf. the following section) in [39, 135, 4]. □

In the following section we describe a distributed algorithm that can solve convex optimization problems and guarantees asymptotic primal feasibility without resorting to averaging mechanisms.

3.3 Distributed ADMM for Cost-Coupled Problems

In this section we review a distributed algorithm based on the popular Alternating Direction Method of Multipliers (ADMM, cf. Appendix A.4). References for the approach described in this section are, e.g., [77, 110, 81, 122]

We consider a network of N agents that aim to cooperatively solve a cost-coupled problem in the form (3.1). Similarly to distributed dual decomposition, in order to distribute the computation we include sparsity in problem (3.1) by introducing a set of copies of $\mathbf{x}$ and proper coherence constraints matching the sparsity of the communication graph $\mathcal{G}$. That is, problem (3.1) can be equivalently stated as

$$\begin{aligned} \min_{\substack{\mathbf{x}_1,\dots,\mathbf{x}_N \\ \mathbf{z}_1,\dots,\mathbf{z}_N}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to} \quad & \mathbf{x}_i \in X_i, && i \in \{1,\dots,N\}, \\ & \mathbf{x}_i = \mathbf{z}_j, && (i,j) \in \mathcal{E}, \\ & \mathbf{x}_i = \mathbf{z}_i, && i \in \{1,\dots,N\}. \end{aligned} \tag{3.19}$$

This problem reformulation is different from the one used for distributed dual decomposition and is tailored for the ADMM approach which makes use of the augmented Lagrangian. Let us introduce $|\mathcal{E}| + N$ multipliers associated to the coherence constraints. The augmented Lagrangian is

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{X}, \mathbf{Z}, \boldsymbol{\Lambda}) = \sum_{i=1}^{N} \Bigg(f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ij}^\top (\mathbf{x}_i - \mathbf{z}_j) + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \|\mathbf{x}_i - \mathbf{z}_j\|^2 \\ + \boldsymbol{\lambda}_{ii}^\top (\mathbf{x}_i - \mathbf{z}_i) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}_i\|^2 \Bigg),\end{aligned}$$

where $\mathbf{X}$, $\mathbf{Z}$ and $\boldsymbol{\Lambda}$ denote the vectors stacking all the primal variables and all the multipliers, respectively.

The ADMM algorithm described in Appendix A.4 can applied to problem (3.19) using the following identifications. The decision variables $\mathbf{x}$ and $\mathbf{z}$ of (A.11) are $\mathbf{X}$ and $\mathbf{Z}$, respectively. As for the cost functions, we set

$$G_1(\mathbf{X}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i), \qquad G_2(\mathbf{Z}) = 0.$$

As for the constraints, $C_1 = X_1 \times \cdots \times X_N$ while $C_2 \equiv \mathbb{R}^{N \cdot d}$. Finally, the linear constraints can be stated as

$$\underbrace{\begin{bmatrix} I_{N\cdot d} \\ I_{N\cdot d} \end{bmatrix}}_{A} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \underbrace{\begin{bmatrix} \text{Adj} \otimes I_d \\ I_{N\cdot d} \end{bmatrix}}_{B} \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{bmatrix},$$

and c equal to zero, where Adj is the adjacency matrix of $\mathcal{G}$ (without self-loops) while $I_{N\cdot d}$ and I_d are $Nd \times Nd$ and $d \times d$ identity matrices, respectively.

Remark 3.3. An alternative formulation of problem (3.1) has been largely used in the literature and it is known as consensus-ADMM formulation (see, e.g., [161]). Formally, the following equivalent formulation of problem (3.1) is considered,

$$\begin{aligned} \min_{\substack{\mathbf{x}_1,\dots,\mathbf{x}_N \\ \{\mathbf{z}_{ij}\}_{(i,j)\in\mathcal{E}}}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to } \; & \mathbf{x}_i \in X_i, \qquad i \in \{1,\dots,N\} \\ & \mathbf{x}_i = \mathbf{z}_{ij}, \qquad (i,j)\in\mathcal{E} \\ & \mathbf{x}_i = \mathbf{z}_{ji}, \qquad (i,j)\in\mathcal{E}. \end{aligned} \tag{3.20}$$

The resulting ADMM algorithm is derived by following the same steps performed for problem (3.19). However, notice that problem (3.20) has $|\mathcal{E}| + N$ variables and $2\cdot|\mathcal{E}|$ coherence constraints, while problem (3.19) has only $2\cdot N$ variables and $|\mathcal{E}| + N$ coherence constraints. □

ADMM for problem (3.19) turns out to be a fully distributed algorithm. Indeed, the primal **x**-minimization step reads

$$\mathbf{x}_i^{t+1} = \operatorname*{argmin}_{\mathbf{x}_i\in X_i} f_i(\mathbf{x}_i) + \Big(\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ij}^t + \boldsymbol{\lambda}_{ii}^t\Big)^\top \mathbf{x}_i + \frac{\rho}{2}\sum_{j\in\mathcal{N}_i\cup\{i\}} \|\mathbf{x}_i - \mathbf{z}_j^t\|^2. \tag{3.21a}$$

The primal **z**-minimization step is

$$\mathbf{z}_i^{t+1} = \operatorname*{argmin}_{\mathbf{z}_i} -\Big(\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ji}^t + \boldsymbol{\lambda}_{ii}^t\Big)^\top \mathbf{z}_i + \frac{\rho}{2}\sum_{j\in\mathcal{N}_i\cup\{i\}} \|\mathbf{x}_j^{t+1} - \mathbf{z}_i\|^2. \tag{3.21b}$$

Finally, the dual ascent step reads

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t + \rho\,(\mathbf{x}_i^{t+1} - \mathbf{z}_j^{t+1}), \tag{3.21c}$$

$$\boldsymbol{\lambda}_{ii}^{t+1} = \boldsymbol{\lambda}_{ii}^t + \rho\,(\mathbf{x}_i^{t+1} - \mathbf{z}_i^{t+1}), \tag{3.21d}$$

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

It is possible to rephrase the **z**-minimization in (3.21b) by noticing that it is an unconstrained quadratic program. The first order necessary condition of optimality is

$$-\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ji}^t - \boldsymbol{\lambda}_{ii}^t - \rho \sum_{j\in\mathcal{N}_i\cup\{i\}} \mathbf{x}_j^{t+1} - \rho\,(|\mathcal{N}_i| + 1)\mathbf{z}_i^{t+1} = \mathbf{0}.$$

Thus, the explicit solution of (3.21b) is given by

$$\mathbf{z}_i^{t+1} = \frac{\sum_{j\in\mathcal{N}_i\cup\{i\}} \mathbf{x}_j^{t+1} + \mathbf{x}_i^t}{|\mathcal{N}_i| + 1} + \frac{\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{ji}^t + \boldsymbol{\lambda}_{ii}^t}{\rho\,(|\mathcal{N}_i| + 1)}.$$

Figure 3.4 shows the algorithmic flow of distributed ADMM, while in Algorithm 4 we summarize the distributed ADMM algorithm from the perspective of agent i. As for the initialization, each agent i can initialize $\boldsymbol{\lambda}_{ij}^t$ for $j \in \mathcal{N}_i$, $\boldsymbol{\lambda}_{ii}^t$ and $\mathbf{z}_i^t$ to arbitrary vectors in $\mathbb{R}^d$.

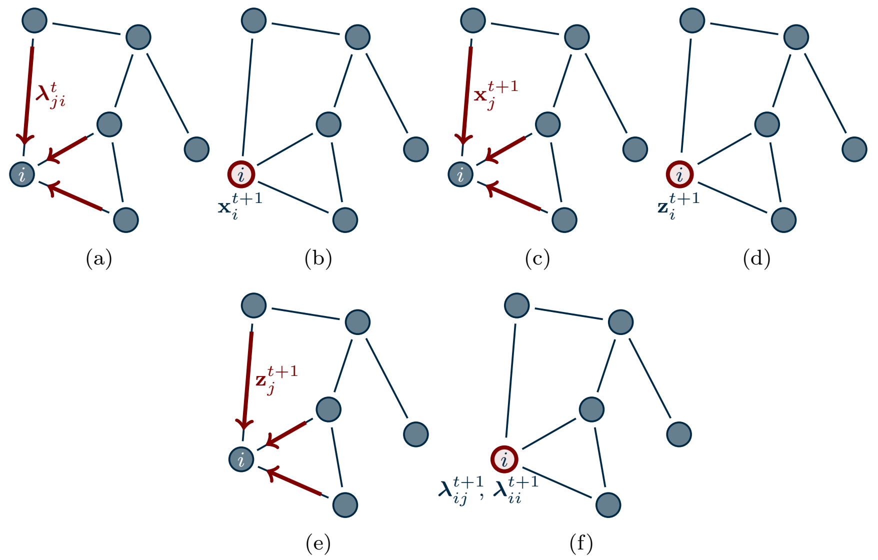


Figure 3.4: Algorithmic evolution of distributed ADMM: in (a) each node receives the dual variables from its neighbors. In (b), the local primal variable $\mathbf{x}$ is updated according to (3.22). Then, in (c) the variables $\mathbf{x}$ are broadcast to neighbors to allow for the update of the variables $\mathbf{z}$, in (d), as in (3.23). Finally, in (e) the primal variables $\mathbf{z}$ are broadcast to neighbors to allow for the dual variables update, in (f), as in (3.24).

Next, we establish convergence of the distributed ADMM algorithm.

Theorem 3.3. Let Assumption 3.1 hold and let the communication graph be undirected and connected. Then, the sequences of local solution estimates $\{\mathbf{x}_i^t\}_{t\geq 0}$, $i \in \{1, \ldots, N\}$, generated by Algorithm 4 are asymptotically consensual to an optimal solution $\mathbf{x}^\star$ of problem (3.1),

Algorithm 4 Distributed ADMM

```
**Initialization**: $\boldsymbol{\lambda}_{ij}^0$ for all $j \in \mathcal{N}_i$, $\boldsymbol{\lambda}_{ii}^0$ and $\mathbf{z}_i^0$
**Evolution**: for $t = 0, 1, \ldots$
  **Gather** $\boldsymbol{\lambda}_{ji}^t$ from neighbors $j \in \mathcal{N}_i$
  **Compute**
    $$\mathbf{x}_i^{t+1} = \operatorname*{argmin}_{\mathbf{x}_i \in X_i} f_i(\mathbf{x}_i) + \Big( \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ij}^t + \boldsymbol{\lambda}_{ii}^t \Big)^\top \mathbf{x}_i + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i \cup \{i\}} \|\mathbf{x}_i - \mathbf{z}_j^t\|^2 \tag{3.22}$$
  **Gather** $\mathbf{x}_j^{t+1}$ from neighbors $j \in \mathcal{N}_i$
  **Compute** $\mathbf{z}_i^{t+1}$ as
    $$\mathbf{z}_i^{t+1} = \frac{\sum_{j \in \mathcal{N}_i \cup \{i\}} \mathbf{x}_j^{t+1}}{|\mathcal{N}_i| + 1} + \frac{\sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ji}^t + \boldsymbol{\lambda}_{ii}^t}{\rho\,(|\mathcal{N}_i| + 1)} \tag{3.23}$$
  **Gather** $\mathbf{z}_j^{t+1}$ from neighbors $j \in \mathcal{N}_i$
  **Update**
    $$\begin{aligned} \boldsymbol{\lambda}_{ij}^{t+1} &= \boldsymbol{\lambda}_{ij}^t + \rho\,(\mathbf{x}_i^{t+1} - \mathbf{z}_j^{t+1}), \qquad j \in \mathcal{N}_i \\ \boldsymbol{\lambda}_{ii}^{t+1} &= \boldsymbol{\lambda}_{ii}^t + \rho\,(\mathbf{x}_i^{t+1} - \mathbf{z}_i^{t+1}) \end{aligned} \tag{3.24}$$
```

i.e.,

$$\lim_{t \to \infty} \|\mathbf{x}_i^t - \mathbf{x}^\star\| = 0.$$

Proof (Sketch). The proof heavily relies on the constructive derivation we carried out in this section. We have shown that Algorithm 4 is an istance of the ADMM algorithm (cf. (A.10) in Appendix A.4) applied to problem (3.19). Thus, by Proposition A.3, it follows that the primal variable sequence $\{(\mathbf{x}_1^t, \ldots, \mathbf{x}_N^t)\}_{t \geq 0}$ converges to an optimal (hence feasible) solution of problem (3.19). Recalling that problem (3.19) is an equivalent formulation of (3.1), the proof follows. □

3.4 Distributed Dual Methods for Constraint-Coupled Problems

In this section, we consider a constraint-coupled optimization problem (cf. Section 1.2.3). We describe how duality can be exploited to develop distributed optimization algorithms for this problem class. Notice that the methods discussed in Section 3.2 and Section 3.3 are designed for a different problem set-up.

3.4.1 Connections between Cost-Coupled and Constraint-Coupled Problems via Duality

In Section 3.1, we have shown that the Fenchel-dual problem (3.10) of a cost-coupled problem is a constraint-coupled problem. Next, we show that there exists a more general symmetry between these two set-ups. In the following, we discuss how duality can be employed to express constraint-coupled problems as cost-coupled ones. Consider a constraint-coupled problem

$$\begin{aligned} \min_{\mathbf{x}_1,\dots,\mathbf{x}_N} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to} \quad & \mathbf{x}_i \in X_i, \qquad i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \le \mathbf{0}, \end{aligned} \tag{3.25}$$

where all the quantities have been introduced in Section 1.2.3.

To derive the dual problem of (3.25), let us introduce a multiplier $\boldsymbol{\mu} \in \mathbb{R}^S$ associated to the coupling constraint $\sum_{i=1}^{N} \mathbf{g}_i(\mathbf{x}_i) \le \mathbf{0}$. Thus, the Lagrangian reads

$$\mathcal{L}(\mathbf{x}_1,\dots,\mathbf{x}_N,\boldsymbol{\mu}) = \sum_{i=1}^{N} \Big(f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top \mathbf{g}_i(\mathbf{x}_i) \Big).$$

The dual of problem (3.25) is

$$\max_{\boldsymbol{\mu}\ge \mathbf{0}} q(\boldsymbol{\mu}) = \max_{\boldsymbol{\mu}\ge \mathbf{0}} \sum_{i=1}^{N} q_i(\boldsymbol{\mu}), \tag{3.26}$$

where the i-th term q_i of the dual function q is defined as

$$q_i(\boldsymbol{\mu}) = \min_{\mathbf{x}_i \in X_i} f_i(\mathbf{x}_i) + \boldsymbol{\mu}^\top \mathbf{g}_i(\mathbf{x}_i). \tag{3.27}$$

It is easy to see that (3.26) is a cost-coupled problem.

We consider N agents in a network modeled as a connected, fixed and undirected graph, which aim to cooperatively solve a constraint-coupled problem (3.25) satisfying the following assumption.

Assumption 3.4. For all $i \in \{1, \ldots, N\}$: each function f_i is convex, each constraint X_i is a non-empty, compact and convex set; each function $\mathbf{g}_i$ is a component-wise convex function. Moreover, there exist $\bar{\mathbf{x}}_1 \in X_1, \ldots, \bar{\mathbf{x}}_N \in X_N$ such that $\sum_{i=1}^N \mathbf{g}_i(\bar{\mathbf{x}}_i) < \mathbf{0}$. □

The latter part of Assumption 3.4 is Slater's constraint qualification and ensures that strong duality holds.

We recall that each agent i aims to compute only its portion $\mathbf{x}_i^\star$ of the entire optimal solution $(\mathbf{x}_1^\star, \ldots, \mathbf{x}_N^\star)$ (cf. Section 1.3). In the following, we introduce two distributed algorithms that solve problem (3.25) by means of problem (3.26).

3.4.2 Distributed Dual Subgradient Algorithm

A (centralized) subgradient method (cf. Appendix A.2) applied to the maximization of the concave problem (3.26) reads

$$\begin{aligned} \boldsymbol{\mu}^{t+1} &= \mathcal{P}_{\boldsymbol{\mu} \geq \mathbf{0}}\Big(\boldsymbol{\mu}^t + \gamma^t \widetilde{\nabla} q(\boldsymbol{\mu}^t)\Big) \\ &= \mathcal{P}_{\boldsymbol{\mu} \geq \mathbf{0}}\Big(\boldsymbol{\mu}^t + \gamma^t \sum_{i=1}^N \widetilde{\nabla} q_i(\boldsymbol{\mu}^t)\Big). \end{aligned} \tag{3.28}$$

Notice that, as discussed in Appendix A.3, a subgradient of q_i at $\boldsymbol{\mu}^t$ can be computed by evaluating the dualized constraints $\mathbf{g}_i$ at the minimizer of the Lagrangian, i.e.,

$$\mathbf{x}_i^{t+1} = \operatorname*{argmin}_{\mathbf{x}_i \in X_i} \; f_i(\mathbf{x}_i) + \big(\boldsymbol{\mu}^t\big)^\top \mathbf{g}_i(\mathbf{x}_i),$$

so that $\widetilde{\nabla} q_i(\boldsymbol{\mu}^t) = \mathbf{g}_i(\mathbf{x}_i^{t+1})$. The method described by (3.28) suggests that the distributed subgradient algorithm (cf. Section 2.1) can be applied to solve problem (3.26).

In the following, we describe the distributed dual subgradient algorithm. Each node i maintains a local dual variable estimate $\boldsymbol{\mu}_i^t$ that is iteratively updated according to a distributed subgradient iteration described by (3.30), and a local primal variable $\mathbf{x}_i^t$, computed by minimizing the i-th term of the Lagrangian as in (3.29). Nodes initialize their local dual variables $\boldsymbol{\mu}_i^t$ to any vector in the positive orthant. Algorithm 5 formally summarizes the distributed dual subgradient algorithm for a constraint-coupled optimization problem (from the perspective of agent i).

Algorithm 5 Distributed Dual Subgradient

```
**Initialization**: $\boldsymbol{\mu}_i^0 \geq \mathbf{0}$
**Evolution**: for $t = 0, 1, ...$
  **Gather** $\boldsymbol{\mu}_j^t$ from neighbors $j \in \mathcal{N}_i$
  **Compute**
      $\mathbf{v}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij}\, \boldsymbol{\mu}_j^t$
      $\mathbf{x}_i^{t+1} \in \operatorname{argmin}_{\mathbf{x}_i \in X_i} \; f_i(\mathbf{x}_i) + \big(\mathbf{v}_i^{t+1}\big)^\top \mathbf{g}_i(\mathbf{x}_i)$      (3.29)
  **Update**
      $\boldsymbol{\mu}_i^{t+1} = \mathcal{P}_{\boldsymbol{\mu} \geq \mathbf{0}}\Big(\mathbf{v}_i^{t+1} + \gamma^t\, \mathbf{g}_i(\mathbf{x}_i^{t+1})\Big)$      (3.30)
```

Being Algorithm 5 a distributed subgradient method (cf. Algorithm 1), the usual convergence properties (discussed in Chapter 2) apply[3]. Consider the same network framework as in Section 2.1 and let Assumption 3.4 hold. We now state the convergence result of the distributed dual subgradient algorithm.

Theorem 3.5. Let Assumption 3.4 hold. Let the communication graph be undirected and connected with weights a_{ij} satisfying Assumption 2.1 and let the step-size γ^t satisfy Assumption 2.2. Then, the sequence of dual variables $\{\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_N^t\}_{t \geq 0}$ generated by Algorithm 5 satisfies

$$\lim_{t \to \infty} \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}^\star\| = 0, \qquad i \in \{1, \ldots, N\},$$

[3]We give the analysis for unconstrained problems, however the algorithm can be extended to a constrained set-up, see, e.g., [96]

where $\boldsymbol{\mu}^\star$ is an optimal solution of problem (3.26), the dual of problem (3.25). Moreover, let the sequence $\{\widehat{\mathbf{x}}_i^t\}_{t\geq 0}$ be defined as $\widehat{\mathbf{x}}_i^t = 1/t \sum_{\tau=0}^t \mathbf{x}_i^\tau$, for all t. Then, it holds

$$\begin{aligned} &\lim_{t\to\infty} \sum_{i=1}^N f_i(\widehat{\mathbf{x}}_i^t) = f^\star, \\ &\lim_{t\to\infty} \|\widehat{\mathbf{x}}_i^t - \mathbf{x}^\star\| = 0, \qquad i \in \{1, \ldots, N\}, \end{aligned}$$

where $\mathbf{x}^\star$ and $f^\star$ denote an optimal solution and the optimal cost of problem (3.25), respectively. $\square$

A proof of the statement is provided in [40] for time-varying networks using a proximal minimization perspective. Notice that Theorem 3.5 does not state any convergence property for the primal variables $\mathbf{x}_i^t$. To this end, as done in Section 3.2, it is useful to employ a local running average (i.e., $\widehat{\mathbf{x}}_i^t$). When the cost function of problem (3.25) is strictly convex, problem (3.25) has a unique optimal solution. In this scenario, convergence of $\mathbf{x}_i^t$ is guaranteed in any case, so that no primal recovery issues arise and no local running average is necessary.

The distributed dual subgradient algorithm enjoys 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 agents do not communicate, and thus disclose, their estimates of the local decision variable, cost or constraints.

3.4.3 Relaxation and Successive Distributed Decomposition

Next, we present a distributed algorithm, named Relaxation and Successive Distributed Decomposition (RSDD), to solve constraint-coupled problems of the form (3.25) that has been proposed and analyzed in [101, 103]. The main leading ideas of the algorithmic development are: (i) to solve the (cost-coupled) dual problem (3.26) by means of distributed dual decomposition, and (ii) to handle infeasibility of local problems, occurring during the algorithmic evolution, via a suitable relaxation. The combination of relaxation and duality steps give rise to a simple and efficient distributed algorithm that overcomes some limitations of the dual distributed subgradient (cf. Section 3.4.2) related to primal recovery.

Algorithm 6 formally states the RSDD distributed algorithm from the perspective of node i.

Algorithm 6 RSDD

```
  **Initialization**: $\boldsymbol{\lambda}_{ij}^0$ for all $j \in \mathcal{N}_i$
  **Evolution**:
    **Gather** $\boldsymbol{\lambda}_{ji}^t$ from neighbors $j \in \mathcal{N}_i$
    **Compute** $((\mathbf{x}_i^{t+1}, \rho_i^{t+1}), \boldsymbol{\mu}_i^{t+1})$ as a primal-dual optimal solution pair of
```

$$\begin{aligned} \min_{\mathbf{x}_i, \rho_i} \;& f_i(\mathbf{x}_i) + M\rho_i \\ \text{subj. to } & \mathbf{x}_i \in X_i, \; \rho_i \geq 0 \\ & \mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \leq \rho_i \mathbf{1} \end{aligned} \tag{3.31}$$

```
    **Gather** $\boldsymbol{\mu}_j^{t+1}$ from neighbors $j \in \mathcal{N}_i$
    **Update** for all $j \in \mathcal{N}_i$
```

$$\boldsymbol{\lambda}_{ij}^{t+1} = \boldsymbol{\lambda}_{ij}^t - \gamma^t(\boldsymbol{\mu}_i^{t+1} - \boldsymbol{\mu}_j^{t+1}) \tag{3.32}$$

Informally, the RSDD algorithm consists of an iterative two-step procedure. Each node i stores a set of variables $((\mathbf{x}_i, \rho_i), \boldsymbol{\mu}_i)$, obtained as a primal-dual optimal solution pair of problem (3.31). The vector $\boldsymbol{\mu}_i$ is the multiplier associated to the local inequality constraint $\mathbf{g}_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t) \leq \rho_i \mathbf{1}$. Notice that problem (3.31) mimics a local version of the original problem (3.25), where the coupling with the other nodes 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. Finally, instead of minimizing only the local function f_i, the (scaled) violation $M\rho_i$, $M > 0$, enters the cost function as well. The auxiliary variables $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$, are updated in a second step according to a linear law which combines neighboring $\boldsymbol{\mu}_i$ as shown in (3.32). Nodes initialize their variables $\boldsymbol{\lambda}_{ij}^t$, $j \in \mathcal{N}_i$ to arbitrary values.

Similarly to the distributed dual subgradient algorithm, the RSDD algorithm also enjoys the same appealing features mentioned in Section 3.4.2, i.e., nicely scaling local computation and information privacy preserving. Moreover, a peculiarity of RSDD is that an estimate of a primal optimal solution component is directly computed by each agent without any averaging mechanism, which results in a faster algorithm.

Consider the same network framework as in Section 2.1 and let Assumption 3.4 hold. We now present the convergence result of RSDD.

Theorem 3.6. Let Assumption 3.4 hold. Let the communication graph be undirected and connected and let the step-size γ^t satisfy Assumption 2.2. Moreover, letting $\boldsymbol{\mu}^\star$ be an optimal solution of the dual of problem (3.25), assume M be sufficiently large such that $M > \|\boldsymbol{\mu}^\star\|_1$. Consider a sequence $\{\mathbf{x}_i^t, \rho_i^t\}_{t\geq 0}$, $i \in \{1,\dots,N\}$, generated by Algorithm 6. Then, the following holds:

(i) the sequence $\{\sum_{i=1}^N \big(f_i(\mathbf{x}_i^t) + M\rho_i^t\big)\}_{t\geq 0}$ converges to the optimal cost $f^\star$ of (3.25);

(ii) every limit point of $\{\mathbf{x}_i^t\}_{t\geq 0}$, $i \in \{1,\dots,N\}$, is a primal optimal (feasible) solution of (3.25). □

The proof of Theorem 3.6 can be found in [103].

In [21], Algorithm 6 has been interpreted as a distributed primal decomposition method and has been used to solve mixed-integer linear programs by means a suitable coupling constraint restriction. The main challenge is due to the presence of local constraint sets X_i that are mixed-integer polyhedra (i.e., with some of the components constrained to be integer, see also Section 4.3.2).

Remark 3.4. Another important optimization set-up in smart grid applications arises in so-called Demand Side Management (DSM) programs [2]. As an example, a cooperative DSM task has the goal of reducing 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), which gives rise to the following min-max optimization problem

$$\begin{aligned} \min_{\mathbf{x}_1,\ldots,\mathbf{x}_N,P} \;& p \\ \text{subj. to } & \mathbf{x}_i \in X_i, && i \in \{1,\ldots,N\}, \\ & \sum_{i=1}^{N} g_{i,s}(x_{i,s}) \leq p, && s \in \{1,\ldots,S\}, \end{aligned} \tag{3.33}$$

where $p \in \mathbb{R}$ represents the peak value that agents want to shave. A duality-based approach similar to the one leading to the RSDD distributed algorithm has been proposed and analyzed in [99, 100] for solving problem (3.33). □

3.5 Discussion and References

Early popular tutorials on parallel and distributed optimization based on duality are [109, 154, 18]. Distributed algorithms based on the Alternating Direction Method of Multipliers are proposed in [77, 110, 81, 27, 26, 131]. Convergence rates for ADMM-based algorithms are provided in [140, 122, 52, 74]. A distributed algorithm combining a linearization approach with ADMM has been proposed in [70], while quadratic approximations have been explored in [80]. A fast distributed ADMM algorithm for quadratic problems is devised in [68]. A more general ADMM framework is considered in [50], where an explicit converge rate has been provided. An application of the distributed ADMM algorithm to an online optimization scenario (i.e., with time-varying cost function) is analyzed in [69]. An asynchronous version of the distributed ADMM algorithm is proposed in [60].

Primal-dual algorithms for constrained optimization over networks are given in [163, 61]. A primal-dual perturbation approach is explored in the paper [28]. An asynchronous version of such algorithm class is provided in [14]. Augmented Lagrangian algorithms for directed gossip networks are analyzed in [53]. Continuous-time, Lagrangian-based, distributed algorithms are investigated in [43, 58, 30, 31, 76, 155]. A distributed saddle-point algorithm for robust linear programs is proposed in [117]. A saddle-point method for distributed, continuous-time, online optimization is proposed in [65]. An asynchronous, primal-dual, cloudbased algorithm for distributed convex optimization is provided in [47]. An asynchronous algorithm which allows the presence of local nonconvex constraints is presented in [42].

A dual averaging approach for distributed optimization is proposed in [36]. A push-sum version for directed networks is analyzed in [136], while an extension for online optimization is given in [64]. A fully distributed dual gradient algorithm to minimize linearly constrained separable convex problems, with linear convergence rate, is given in [83]. A distributed dual fast gradient algorithm, with sublinear rate, is proposed [84] for linearly constrained separable convex optimization problems. An asynchronous version of the distributed dual decomposition with composite costs is proposed in [102]. An extension to a partitioned set-up is provided in [98]. A time-varying distributed algorithm based on Fenchel duality is provided in [144]. Papers [123, 40] investigate distributed dual subgradient methods for constraint-coupled optimization. In [160] an ADMM approach for the same set-up is proposed in which multiple consensus steps are needed. Dual decomposition techniques applied to control problems are proposed in [34, 44]. In [35] a distributed Jacobi algorithm for convex optimization problems, arising in distributed model predictive control, is presented. A fast dual gradient algorithm for network utility maximization is proposed in [6].

3.6 Numerical Example

In this section, we provide numerical examples of the algorithms presented in this chapter. Since we considered algorithms for both the cost-coupled set-up and the constraint-coupled set-up, we analyze the examples in two separate subsections.

As done in Chapter 2, we consider a network of $N = 10$ agents communicating over a fixed, undirected, connected graph generated according to an Erdős-Rényi random model with parameter $p = 0.2$. For the algorithms embedded with consensus iterations, we assume agents are equipped with a doubly stochastic matrix built according to the

Metropolis-Hastings rule [148], i.e.,

$$a_{ij} = \begin{cases} \frac{1}{\max\{d_i,d_j\}+1}, & \text{if } j \neq i \text{ and } (i,j) \in \mathcal{E}, \\ 1 - \sum_{j\in\mathcal{N}_i} \frac{1}{\max\{d_i,d_j\}+1}, & \text{if } j = i, \\ 0, & \text{otherwise.} \end{cases}$$

3.6.1 Cost-coupled Example

In this subsection, we assume that N agents aim to cooperatively solve the cost-coupled quadratic program

$$\min_{\mathbf{x}\in\mathbb{R}^5} \sum_{i=1}^{N} \left(\mathbf{x}^\top Q_i \mathbf{x} + r_i^\top \mathbf{x}\right), \tag{3.34}$$

where each $Q_i \in \mathbb{R}^{5\times 5}$ is randomly generated such that its eigenvalues are drawn uniformly from $[1, 10]$. We compare distributed ADMM (cf. Algorithm 4), with $\rho = 0.1$ and distributed dual decomposition (cf. Algorithm 3), with diminishing step-size $\gamma^t = (1/t)^{0.7}$.

As for distributed ADMM, in Figure 3.5 we show cost convergence rate, i.e., the evolution of $|\sum_{i=1}^N f_i(\mathbf{x}_i^t) - f^\star|/|f^\star|$. In Figure 3.6, we

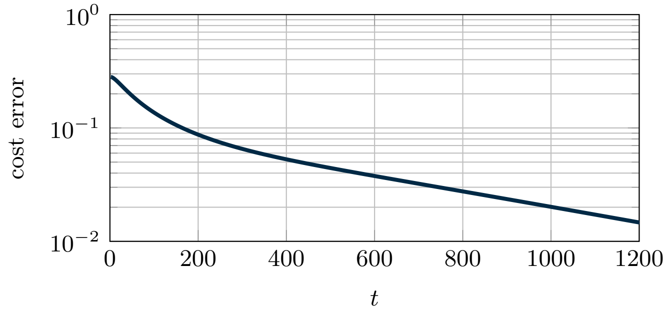


Figure 3.5: Evolution of the cost error for the distributed ADMM algorithm for cost-coupled problems.

show the consensus error of the local solution estimates, i.e., $\|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\|$ for all i, where $\bar{\mathbf{x}}^t = 1/N \cdot \sum_{i=1}^N \mathbf{x}_i^t$.

As regards distributed dual decomposition, in Figure 3.7 we show cost convergence. That is, we plot the evolution of primal and dual cost

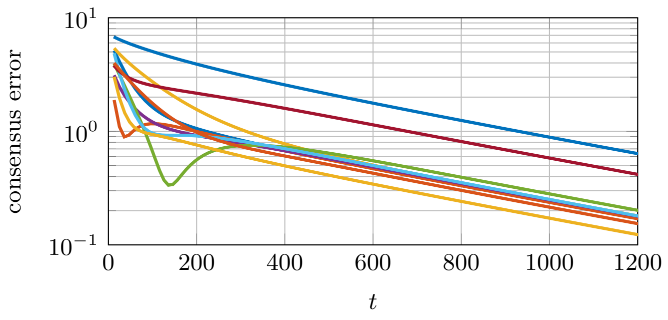


Figure 3.6: Evolution of the consensus error for the distributed ADMM algorithm for cost-coupled problems. Each line refers to an agent in the network.

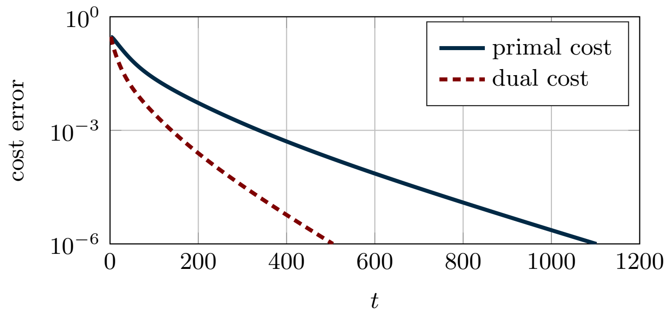


Figure 3.7: Evolution of primal and dual cost errors for the distributed dual decomposition algorithm for cost-coupled problems.

error, i.e., $|\sum_{i=1}^{N} f_i(\mathbf{x}_i^t) - f^\star|/|f^\star|$ and $|q(\mathbf{\Lambda}^t) - f^\star|/|f^\star|$. As expected for a dual method, dual cost converges faster than primal cost.

Finally, in Figure 3.8 we show consensus error of the local solution estimates, i.e., $\|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\|$ for all i, where $\bar{\mathbf{x}}^t = 1/N \cdot \sum_{i=1}^{N} \mathbf{x}_i^t$.

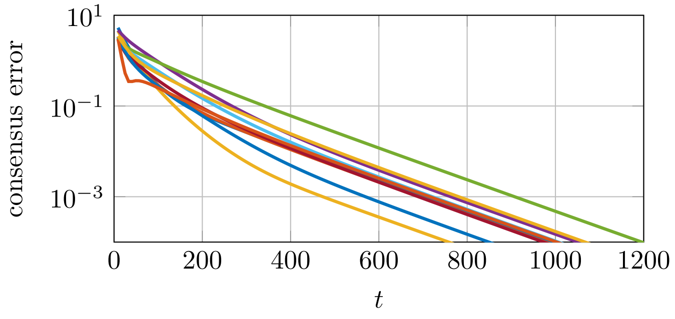


Figure 3.8: Evolution of the consensus error for the distributed dual decomposition for cost-coupled problems. Each line refers to an agent in the network.

3.6.2 Constraint-coupled Example

In this subsection, we consider the Microgrid control problem introduced in Section 1.3.6, where we assume we have 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 $S = 12$ slots. The optimization problem to be solved has the form (cf. Section 1.3.6)

$$\begin{aligned} \min_{\mathbf{x}_1,\ldots,\mathbf{x}_N} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subj. to} \quad & \sum_{i\in\texttt{GEN}} p_{\texttt{gen},i}^{\tau} + \sum_{i\in\texttt{STOR}} p_{\texttt{stor},i}^{\tau} + \sum_{i\in\texttt{CONL}} p_{\texttt{conl},i}^{\tau} + p_{\texttt{tr}}^{\tau} \geq D^{\tau}, \\ & \qquad\qquad \forall\, s \in [0, S], \\ & \mathbf{x}_i \in X_i, \qquad\qquad \forall\, i \in \{1,\ldots,N\}. \end{aligned} \tag{3.35}$$

We compare RSDD (cf. Algorithm 6) and distributed dual subgradient (cf. Algorithm 5). For both algorithms, we use the diminishing step-size $\gamma^t = 0.1 \cdot (1/t)^{0.7}$. For RSDD, we set $M = 10 \cdot \|\boldsymbol{\mu}^\star\|_1$, where

$\boldsymbol{\mu}^\star$ is a dual optimal solution of the problem (3.35) computed by a centralized solver.

In Figure 3.9 we compare the convergence rate of RSDD and of distributed dual subgradient. In particular, for the RSDD algorithm, we plot the difference between the optimal cost $f^\star$ and the sum of local costs $\sum_{i=1}^N f_i(\mathbf{x}_i^t)$, normalized by $f^\star$. For the distributed dual subgradient algorithm, we plot the difference between the optimal cost $f^\star$ and the sum of local costs $\sum_{i=1}^N f_i(\widehat{\mathbf{x}}_i^t)$, normalized by $f^\star$, where $\widehat{\mathbf{x}}_i^t$ denotes the i-th running average of the local Lagrangian minimizers $\mathbf{x}_i^t$.

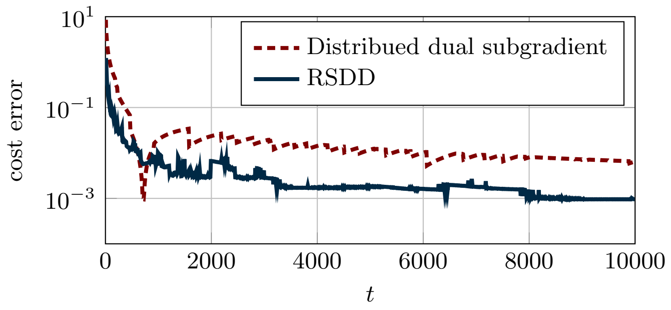


Figure 3.9: Evolution of the cost error $|\sum_{i=1}^N \big(f_i(\mathbf{x}_i^t) + M\rho_i^t\big) - f^\star|/|f^\star|$ that shows convergence to the optimal cost.

For the RSDD algorithm, in Figure 3.10, 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.

Finally, in Figure 3.11 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 constraint of each agent i, namely $\sum_{j\neq i}\mathbf{g}_j(\mathbf{x}_j^t)$. Specifically, for all i, we plot the quantity

$$\max_{s\in\{1,\ldots,S\}}\Big(\sum_{h\neq i}\mathbf{g}_{hs}(\mathbf{x}_h^t) - \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)_s\Big).$$

The picture highlights that $\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$ acts as a “tracker” of the maximum of the contribution in the coupling constraint due to all the other agents $j\neq i$ in the network.

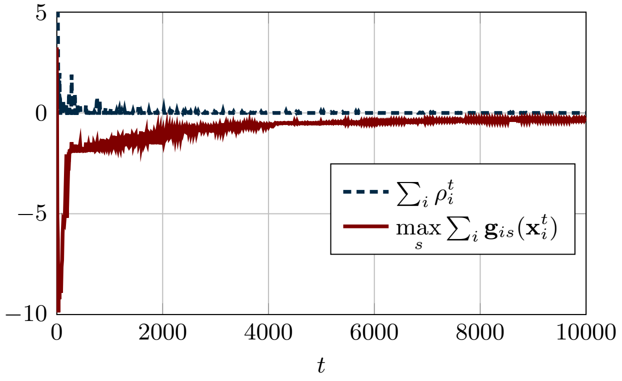


Figure 3.10: Evolution of the maximum violation of coupling constraints showing the feasibility of generated primal sequences (red). Asymptotically vanishing behavior of the sum of local violations (blue).

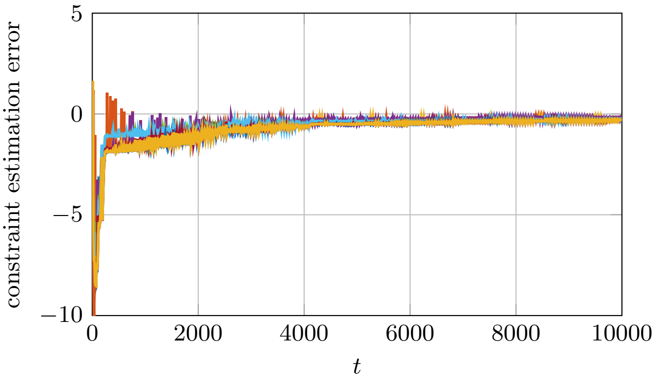


Figure 3.11: Evolution of the error between the unknown part of the coupling constraint and the local term $\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}^t - \boldsymbol{\lambda}_{ji}^t)$, for all i, showing an asymptotic tracking property of the auxiliary variables.

4

Constraint Exchange Methods

In this chapter, we present distributed optimization algorithms based on the exchange of constraints among agents. These algorithms are structurally different from the ones described in Chapters 2 and 3, since the information exchanged by agents (encoding the local solution estimate) amounts to constraints rather than decision variables. We start by introducing the so-called Constraints Consensus algorithm for convex and abstract programs.[1] Following the same approach as in the previous chapter, we present and analyze the algorithm for a simplified optimization set-up, namely linear programs, and then discuss how it in fact applies to general convex and abstract programs. Then, we present other methods based on the constraint exchange approach that generalize Constraints Consensus. Finally, we provide a numerical example to show the main characteristics of the presented methods.

4.1 Constraints Consensus applied to Linear Programs

In this section, we present and analyze a simplified version, applied to Linear Programs (LPs), of the Constraints Consensus algorithm [108].

[1] Abstract programs are a generalization of linear programs, see, e.g., [1, 108].

First, we give some intuition on the algorithm together with its formal description. Then, we provide a convergence analysis, and we briefly mention a variant of the algorithm in which agents exchange "columns" instead of constraints.

4.1.1 Algorithm description

Consider a network of N agents that aim to solve the linear program

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } \;& a_i^\top \mathbf{x} \le b_i, \quad i \in \{1, \dots, N\}, \end{aligned} \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the optimization variable, $c \in \mathbb{R}^d$ is the cost vector, and $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$, $i \in \{1, \dots, N\}$. Notice that problem (4.1) is an instance of the common cost set-up described in Section 1.2.2. For ease of presentation, we suppose that each agent i knows only the constraint $a_i^\top \mathbf{x} \le b_i$, and we say that this is the initial constraint of agent i. Also, we make the standing assumption that the number of agents is greater than the dimension of the variable, i.e., $N > d$.

To convey the idea underlying the Constraints Consensus algorithm, let us elaborate on optimization problems in the form of (4.1). It is known from linear programming theory (cf. Appendix C) that the feasible set of problem (4.1) is polyhedral and that, if $\mathbf{x}^\star$ is an optimal vertex (i.e., an optimal solution attained at a vertex of the feasible set), then there exists a *basis*, consisting of exactly d linearly independent inequality constraints $a_{\ell_1}^\top \mathbf{x} \le b_{\ell_1}, \dots, a_{\ell_d}^\top \mathbf{x} \le b_{\ell_d}$, for some indices $\{\ell_1, \dots, \ell_d\} \subseteq \{1, \dots, N\}$. Such a basis allows for the computation of $\mathbf{x}^\star$ as the (unique) optimal vertex of the linear program

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } \;& a_{\ell_h}^\top \mathbf{x} \le b_{\ell_h}, \quad h \in \{1, \dots, d\}, \end{aligned} \tag{4.2}$$

obtained as a relaxation of problem (4.1) by considering the constraints in the basis only. Roughly speaking, in the Constraints Consensus algorithm, each agent iteratively solves a relaxation of problem (4.1), with constraints given by its initial constraint and constraints collected from neighbors, and computes an optimal solution with its corresponding basis. Then, the basis is broadcast to neighbors and the process is repeated until convergence. A natural question arising at this point is how to handle problems with multiple optimal solutions. For such problems, in order to guarantee convergence of the scheme, it is necessary for agents to select a common solution. A possible approach to guarantee agent agreement is to employ a lexicographic criterion (see Appendix C for a formal description), i.e., agents compute the lexicographically minimal optimal solution, termed *lex-optimal solution*, of the LPs through an appropriate local lexicographic solver. Thus, in the remainder of this section, we will stick to the following definition of basis.

Definition 4.1. Let $\mathbf{x}^\star$ be the lex-optimal solution of a linear program in the form (4.1). A collection of d inequality constraints $a_{\ell_1}^\top \mathbf{x} \leq b_{\ell_1}, \dots, a_{\ell_d}^\top \mathbf{x} \leq b_{\ell_d}$, for some indices $\{\ell_1, \dots, \ell_d\} \subseteq \{1, \dots, N\}$, is called a *basis* of (4.1) if $\mathbf{x}^\star$ is the lex-optimal solution of

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } \;& a_{\ell_h}^\top \mathbf{x} \leq b_{\ell_h}, \qquad h \in \{1, \dots, d\}. \end{aligned} \qquad \square$$

This definition is specifically tailored for linear programs. In fact, it can be obtained as a special version of a more general definition of basis that holds for so-called *abstract programs*, see, e.g., [1, 108]. From now on, we compactly denote a basis as (P, q), where $P \in \mathbb{R}^{d \times d}$ is the matrix obtained by stacking the row vectors $a_{\ell_h}^\top$ and $q \in \mathbb{R}^d$ is the vector obtained by stacking the scalars b_{ℓ_h}, i.e.,

$$P = \begin{bmatrix} a_{\ell_1}^\top \\ \vdots \\ a_{\ell_d}^\top \end{bmatrix}, \quad q = \begin{bmatrix} b_{\ell_1} \\ \vdots \\ b_{\ell_d} \end{bmatrix}.$$

Notice that, even if the lex-optimal solution of a LP is unique, there might be several bases associated to the problem. In Figure 4.1, an example scenario in $\mathbb{R}^2$ is graphically represented.

Next, we describe the Constraints Consensus algorithm applied to problem (4.1). We assume that the agents communicate according to a jointly strongly connected (time-varying) directed graph $\mathcal{G}^t$ (cf. Section 1.1), and we denote by $\mathcal{N}_i^t$ the in-neighbors of agent i at com-

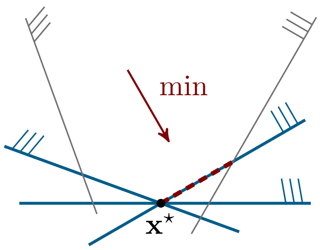


Figure 4.1: Example of an instance of problem (4.1), where the feasible side for each inequality constraint is denoted by three bars. The LP admits several optimal solutions (indicated by a dashed red line), but $\mathbf{x}^\star$ is the lex-optimal solution. Notice that several bases can be chosen (i.e., the constraint on which the optimal solutions lie together with either of the other two constraints in blue).

munication round[2] t. Each agent i maintains a local solution estimate $\mathbf{x}_i^t$ and a local basis (P_i^t, q_i^t). It is initialized to $(a_i^\top, b_i)$ and is incrementally filled as the agent collects information from neighbors during the algorithm evolution. At each communication round t, agent i first gathers the bases from its neighbors, then it constructs a (small) local LP with constraints given by the aggregation of: *(i)* the old basis, *(ii)* the collected bases from neighbors, and *(iii)* its initial constraint. Then, the agent finds a basis for the local LP to update its state. Finally, the updated basis is broadcast to neighbors. Notice that the local LP can be unbounded. Thus, an artificial (sufficiently large) bounding box, denoted as $-M\mathbf{1} \leq \mathbf{x} \leq M\mathbf{1}$, with $M > 0$, is added to ensure that the algorithm is well posed at each communication round, so that the bounding box can becomes part of the local bases. If M is sufficiently large, the lex-optimal solution of problem (4.1) is contained in the bounding box and the bounding box will eventually leave the local bases. Algorithm 7 formally summarizes the Constraints Consensus algorithm applied to linear programs from the perspective of node i.

In Section 4.1.2, we analyze the convergence of Algorithm 7.

Let us now highlight the differences of the constraint exchange approach with respect to the other approaches discussed in this survey.

[2] In a synchronous algorithm the term iteration is more suited. Since the Constraints Consensus algorithm can be implemented also in an asynchronous setting, we prefer to use this terminology.

Algorithm 7 Constraints Consensus applied to LPs

```
**Initialization**: $(P_i^0, q_i^0) = (a_i^\top, b_i)$

**Evolution**: for $t = 0, 1, ...$
  **Gather** $(P_j^t, q_j^t)$ from neighbors $j \in \mathcal{N}_i^t$
  **Compute** $\mathbf{x}_i^{t+1}$ as the lex-optimal solution of
```

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } \;& a_i^\top \mathbf{x} \le b_i \\ & P_i^t \mathbf{x} \le q_i^t \\ & P_j^t \mathbf{x} \le q_j^t, \qquad j \in \mathcal{N}_i^t \\ & -M\mathbf{1} \le \mathbf{x} \le M\mathbf{1} \end{aligned} \tag{4.3}$$

```
  **Update** $(P_i^{t+1}, q_i^{t+1})$ as a basis of (4.3)
```

First, note that in primal methods (cf. Chapter 2), consensus of the agents on a common optimal solution is enforced by means of consensus iterations that steer the local quantities to a common value, whereas in Algorithm 7, consensus follows because eventually the lex-optimal solution of the local problems (4.3) is the same. Second, the communication network assumptions of constraint exchange methods are generally very weak. For instance, Algorithm 7 only requires joint strong connectivity, which allows for an asynchronous implementation of the algorithm (cf. Section 1.1), and allows for unreliable communication links (e.g., subject to packet loss). Also, it is worth mentioning that if the network consists of a large number of agents with relatively small in-degree, the local optimization problem (4.3) solved at each iteration is much smaller than the original problem (4.1), so that the algorithm scales nicely with the network size. This is also corroborated by the fact that the communication is bounded: each exchanged basis always consists of d constraints, except in the early stages of the algorithm in which less than d constraints are available. Finally, note that Algorithm 7 does not require global tuning parameters (e.g., the step-size).

4.1.2 Convergence Analysis

In this subsection, we analyze the convergence of Algorithm 7. The proof reported in this survey is different from the one in [108], which was devised for general abstract programs. Here we present a new proof inspired by the arguments used in [134]. Let us make the following assumption on problem (4.1).

Assumption 4.1. Problem (4.1) is feasible and the lex-optimal solution exists.[3] □

In the following, we prove that Algorithm 7 enjoys finite-time convergence. The line of proof relies on three facts, namely *(i)* (finite-time) convergence of the solution estimates computed by each agent (Lemma 4.2), *(ii)* consensus of the solution estimates at convergence (Lemma 4.3), *(iii)* optimality of the consensual solution estimates.

In the next lemma we prove that the quantities computed by each agent converge in finite time.

Lemma 4.2 (Local convergence)**.** Let Assumption 4.1 hold. Then, for all $i \in \{1, \dots, N\}$,

(i) the cost sequence $\{c^\top \mathbf{x}_i^t\}_{t\geq 0}$ is monotonically non-decreasing and converges in finite time, i.e., there exist $T_i > 0$ and $\bar{J}_i \in \mathbb{R}$ such that

$$c^\top \mathbf{x}_i^t = \bar{J}_i, \qquad \text{for all } t \geq T_i;$$

(ii) the solution estimate sequence $\{\mathbf{x}_i^t\}_{t\geq 0}$ converges in finite time to a vector satisfying the initial constraint of agent i, i.e., there exist $T_i' > 0$ and $\bar{\mathbf{x}}_i$ such that

$$\begin{aligned} \mathbf{x}_i^t &= \bar{\mathbf{x}}_i, \qquad \text{for all } t \geq T_i', \\ a_i^\top \bar{\mathbf{x}}_i &\leq b_i. \end{aligned}$$

Proof. For the sake of analysis, let us denote by $J_i^t \triangleq c^\top \mathbf{x}_i^t$ the cost associated to $\mathbf{x}_i^t$. To prove *(i)*, we consider problem (4.3) at consecutive

[3] For a discussion on the existence of the lex-optimal solution, see Appendix C.

communication rounds, say t and $t+1$. The lex-optimal solution of problem (4.3) is $\mathbf{x}_i^{t+1}$, with cost $J_i^{t+1} = c^\top \mathbf{x}_i^{t+1}$ and (P_i^{t+1}, q_i^{t+1}) an associated basis. Thus, $\mathbf{x}_i^{t+1}$ is the lex-optimal solution of

$$\begin{aligned} \min_{\mathbf{x}} \; & c^\top \mathbf{x} \\ \text{subj. to } & P_i^{t+1}\mathbf{x} \leq q_i^{t+1}, \end{aligned} \tag{4.4}$$

with optimal cost J_i^{t+1}. At the successive communication round $t+1$, the lex-optimal solution of the local problem (4.3) does not violate any constraint of problem (4.4). Thus, it holds $J_i^{t+2} \geq J_i^{t+1}$. Therefore, we conclude that the cost sequence is monotonically non-decreasing, i.e., for all $t \geq 0$,

$$J_i^{t+1} \geq J_i^t, \qquad i \in \{1, \dots, N\}.$$

Also, because of the bounding box, the feasible set of problem (4.3) is bounded, so that $\{J_i^t\}_{t\geq 0}$ converges. Finally, since there is a finite number of constraints in the network, J_i^t can only assume a finite number of values (corresponding to all the possible combinations of constraints). Thus, $\{J_i^t\}_{t\geq 0}$ converges in finite time, i.e., there exist $T_i > 0$ and $\bar{J}_i \in \mathbb{R}$ such that

$$c^\top \mathbf{x}_i^t = J_i^t = \bar{J}_i, \qquad \text{for all } t \geq T_i.$$

To prove *(ii)*, let us consider the sequence of the first component of $\mathbf{x}_i^t$ for $t \geq T_i$, i.e., $\{\mathbf{x}_{i,1}^t\}_{t\geq T_i}$. First, notice that the cost associated to such sequence is identically equal to $\bar{J}_i$, i.e., $c^\top \mathbf{x}_i^t = \bar{J}_i$ for all $t \geq T_i$. In the following, we apply ideas similar to *(i)*, namely we consider problem (4.3) at consecutive communication rounds, say t and $t+1$, with $t \geq T_i$. The lex-optimal solution of problem (4.3) is $\mathbf{x}_i^{t+1}$, with first component $\mathbf{x}_{i,1}^{t+1}$ and (P_i^{t+1}, q_i^{t+1}) an associated basis. Thus, $\mathbf{x}_i^{t+1}$ is the lex-optimal solution of

$$\begin{aligned} \min_{\mathbf{x}} \; & c^\top \mathbf{x} \\ \text{subj. to } & P_i^{t+1}\mathbf{x} \leq q_i^{t+1}. \end{aligned} \tag{4.5}$$

At the successive communication round $t+1$, the optimal cost stays equal to $\bar{J}_i$ and the lex-optimal solution of the local problem (4.3) does not violate any constraint of problem (4.5). Thus, since the local lexicographic solver selects the optimal solution with minimal first component, it follows that $\mathbf{x}_{i,1}^{t+2} \geq \mathbf{x}_{i,1}^{t+1}$. Therefore, we conclude that the sequence $\{\mathbf{x}_{i,1}^t\}_{t \geq T_i}$ is monotonically non-decreasing, i.e., for all $t \geq T_i$,

$$\mathbf{x}_{i,1}^{t+1} \geq \mathbf{x}_{i,1}^t, \qquad i \in \{1, \ldots, N\}.$$

Also, because of the bounding box, the feasible set of problem (4.3) is bounded, so that $\{\mathbf{x}_{i,1}^t\}_{t \geq 0}$ converges. Finally, since there is a finite number of constraints in the network, $\mathbf{x}_{i,1}^t$ can only assume a finite number of values (corresponding to all the possible combinations of constraints). Thus, $\{\mathbf{x}_{i,1}^t\}_{t \geq 0}$ converges in finite time, i.e., there exist $T_i' > 0$ and $\bar{\mathbf{x}}_{i,1} \in \mathbb{R}$ such that

$$\mathbf{x}_{i,1}^t = \bar{\mathbf{x}}_{i,1}, \qquad \text{for all } t \geq T_i'.$$

By repeating the same arguments for each of the subsequent components of $\mathbf{x}_i^t$ for $t \geq T_i'$, we are able to conclude that $\{\mathbf{x}_i^t\}_{t \geq 0}$ converges in finite time to some $\bar{\mathbf{x}}_i$, which by construction satisfies $a_i^\top \bar{\mathbf{x}}_i \leq b_i$. □

In the following lemma, we prove that the solution estimates to which agents converge are consensual.

Lemma 4.3 (Consensus)**.** Let the communication graph be jointly strongly connected. Moreover, assume that the sequences computed by agents have converged, i.e., there exists $T_0 > 0$ such that for all $i \in \{1, \ldots, N\}$ it holds

$$c^\top \mathbf{x}_i^t = \bar{J}_i \ \text{ and } \ \mathbf{x}_i^t = \bar{\mathbf{x}}_i, \qquad \text{for all } t \geq T_0,$$

for some $\bar{J}_i \in \mathbb{R}$ and $\bar{\mathbf{x}}_i \in \mathbb{R}^d$. Then, it holds

$$\bar{J}_i = \bar{J}_j \ \text{ and } \ \bar{\mathbf{x}}_i = \bar{\mathbf{x}}_j, \qquad \text{for all } i, j \in \{1, \ldots, N\}.$$

Proof. For the sake of analysis, let us denote by $J_i^t \triangleq c^\top \mathbf{x}_i^t$ the cost associated to $\mathbf{x}_i^t$. By contradiction, assume that there exist two different agents i and j such that $\bar{J}_i \neq \bar{J}_j$. Without loss of generality, let $\bar{J}_j > \bar{J}_i$.

By finite-time convergence of the cost sequences, there exists $T_0 > 0$ such that $J_j^t = \bar{J}_j > \bar{J}_i = J_i^t$ for all $t \geq T_0$. Moreover, since the communication graph is jointly strongly connected, for all $t \geq T_0$ and each pair of agents (i, j), there exists a sequence of time instants

$\tau_1, \ldots, \tau_k$, with $t \leq \tau_1 < \ldots < \tau_k$, and a sequence of nodes $\nu_1, \ldots, \nu_{k-1}$, such that the directed edges $(j, \nu_1), (\nu_1, \nu_2), \ldots, (\nu_{k-1}, i)$ belong to the digraph at times $\tau_1, \ldots, \tau_k$ (cf. [108]).

At communication round τ_1, agent ν_1 computes $\mathbf{x}_{\nu_1}^{\tau_1+1}$ by minimizing $c^\top \mathbf{x}$ over a subset of the basis associated to $\mathbf{x}_j^{\tau_1}$ (by construction), so that $J_{\nu_1}^{\tau_1+1} \geq J_j^{\tau_1}$. Similarly, at communication round τ_2, agent ν_2 computes $\mathbf{x}_{\nu_2}^{\tau_2+1}$ by minimizing $c^\top \mathbf{x}$ over a subset of the basis associated to $\mathbf{x}_{\nu_1}^{\tau_2}$. Thus, it holds

$$J_{\nu_2}^{\tau_2+1} \geq J_{\nu_1}^{\tau_2}.$$

Since the cost sequences have converged, it follows that $\bar{J}_{\nu_1} = J_{\nu_1}^{\tau_2} = J_{\nu_1}^{\tau_1+1}$. Thus, it holds

$$J_{\nu_2}^{\tau_2+1} \geq J_j^{\tau_1}.$$

The argument can be iterated to conclude that $J_i^{\tau_k+1} \geq J_j^{\tau_1}$. Therefore, for all $t > T_0$ there exists $\theta_{ij} > 0$ such that

$$\bar{J}_i = J_i^{t+\theta_{ij}} \geq J_j^t = \bar{J}_j,$$

contradicting the assumption $\bar{J}_j > \bar{J}_i$. Thus, $\bar{J}_1 = \ldots = \bar{J}_N$, which concludes the first part of the proof. To prove consensus of the solutions, we note that for all $t \geq T_0$, $c^\top \mathbf{x}_1^t = \ldots = c^\top \mathbf{x}_N^t$. Then, it is possible to apply arguments similar to the first part to each component of the solution vector (in lexicographic order, see proof of Lemma 4.2 *(ii)*). □

With Lemma 4.2 and Lemma 4.3 at reach, we are now ready to prove the convergence of Algorithm 7.

Theorem 4.4. Let Assumption 4.1 hold and let the communication graph be jointly strongly connected. Moreover, let $\mathbf{x}^\star$ be the lex-optimal solution of problem (4.1) and assume $M > 0$ is sufficiently large. Consider the sequences $\{\mathbf{x}_i^t\}_{t\geq 0}, i \in \{1, \ldots, N\}$, generated by Algorithm 7. Then, for all $i \in \{1, \ldots, N\}$, the following holds:

1. the cost sequence $\{c^\top \mathbf{x}_i^t\}_{t\geq 0}$ converges in finite time to the optimal cost $J^\star$ of (4.1);

2. the solution sequence $\{\mathbf{x}_i^t\}_{t\geq 0}$ converges in finite time to $\mathbf{x}^\star$.

Proof. For the sake of analysis, let us denote by $J_i^t \triangleq c^\top \mathbf{x}_i^t$ the cost associated to $\mathbf{x}_i^t$. By Lemma 4.2, the cost sequences $\{J_i^t\}_{t\geq 0}$ and the solution sequences $\{\mathbf{x}_i^t\}_{t\geq 0}$ converge in finite time to $\bar{J}_i$ and $\bar{\mathbf{x}}_i$ respectively, and by construction it holds

$$a_i^\top \bar{\mathbf{x}}_i \leq b_i, \qquad \text{for all } i \in \{1, \dots, N\}.$$

By Lemma 4.3, there exist a common scalar $\bar{J} \in \mathbb{R}$ and a common vector $\bar{\mathbf{x}}$ such that $\bar{J}_i = \bar{J}$ and $\bar{\mathbf{x}}_i = \bar{\mathbf{x}}$ for all $i \in \{1, \dots, N\}$. Therefore, $\bar{\mathbf{x}}$ is feasible for problem (4.1), since $a_i^\top \bar{\mathbf{x}} \leq b_i$ for all i. To prove that $\bar{J} = J^\star$, we first note that $\bar{J} \leq J^\star$, since each agent builds up the local LP as a relaxation (i.e., with a lower number of constraints) of the original problem (4.1), and the bounding box is sufficiently large (thus, we can assume that $M > \|\mathbf{x}^\star\|_\infty$). On the other hand, since $\bar{\mathbf{x}}$ is feasible for problem (4.1), then $J^\star \leq c^\top \bar{\mathbf{x}} = \bar{J}$, thus implying $\bar{J} = J^\star$.

Since we have shown that $\bar{\mathbf{x}}$ is feasible and cost-optimal, so that $\bar{\mathbf{x}}$ is an optimal solution of (4.1), we only have to show that it is the lexicographic minimum among all the minima (i.e., $\bar{\mathbf{x}} = \mathbf{x}^\star$). By contradiction, suppose it is not. Then, $\mathbf{x}^\star \overset{L}{<} \bar{\mathbf{x}}$, where the symbol $\overset{L}{<}$ means that $\mathbf{x}^\star$ is lexicographically smaller than $\bar{\mathbf{x}}$ (cf. Appendix C). Now, since $\bar{\mathbf{x}}$ is computed by each agent as the lex-optimal solution to the local problem, there exists a basis $(\bar{P}, \bar{q})$, made up of constraints of problem (4.1), such that $\bar{\mathbf{x}}$ is the lex-optimal solution to

$$\begin{aligned} \min_{\mathbf{x}} \ & c^\top \mathbf{x} \\ \text{subj. to } & \bar{p}_h^\top \mathbf{x} \leq \bar{q}_h, \qquad h \in \{1, \dots, d\}, \end{aligned} \tag{4.6}$$

where $\bar{p}_h^\top \in \mathbb{R}^{1\times d}$ denotes the h-th row of $\bar{P}$ and $\bar{q}_h \in \mathbb{R}$ denotes the h-th entry of $\bar{q}$. But this means that $\mathbf{x}^\star$ must be infeasible for problem (4.6), otherwise the lex-optimal solution of (4.6) would be $\mathbf{x}^\star$ instead of $\bar{\mathbf{x}}$. Therefore, one of the constraints in (4.6) is violated by $\mathbf{x}^\star$, i.e., there exists $h \in \{1, \dots, d\}$ such that $P_h^\top \mathbf{x}^\star > q_h$. But since the constraints in (4.6) are drawn from problem (4.1), this contradicts the fact that $\mathbf{x}^\star$ is feasible for the original LP (4.1). Thus, $\bar{\mathbf{x}} = \mathbf{x}^\star$ and the proof follows. □

A few remarks on the Constraints Consensus algorithm are in order. In the algorithm analysis we did not prove that the local bases are consensual at convergence. Indeed, agents may compute different bases associated to the lex-optimal solution. A sufficient condition for consensus of bases is the so-called *non-degeneracy* of problem (4.1) (see also [108]). Finally, a remarkable property of the algorithm is that a fully distributed halting condition can be obtained. Indeed, if the communication graph is fixed, each agent can halt the execution of the algorithm as soon as the locally computed solution stays constant for $2\,\mathrm{diam}(\mathcal{G})+1$ communication rounds [108, Theorem IV.4]. If the communication graph is time-varying and T-strongly connected (cf. Section 1.1), it can be seen that each agent can halt the execution of the algorithm as soon as the locally computed solution stays constant for $2NT+1$ communication rounds.

4.1.3 Distributed Simplex

In this section, we briefly mention a variant of the Constraints Consensus algorithm applied to LPs, namely the Distributed Simplex algorithm [20]. We consider a network of N agents that aim to cooperatively solve linear programs in the so-called standard form, i.e.,

$$\begin{aligned} \min_{\mathbf{x}} \;\; & c^\top \mathbf{x} \\ \text{subj. to} \;\; & A\mathbf{x} = b, \\ & \mathbf{x} \geq 0, \end{aligned} \tag{4.7}$$

where $A \in \mathbb{R}^{d\times N}$, $b \in \mathbb{R}^d$ and $c \in \mathbb{R}^N$ are the problem data and $\mathbf{x} \in \mathbb{R}^N$ is the decision vector. A *column* of problem (4.7) is defined as the vector

$$h_i \triangleq \begin{bmatrix} c_i \\ a_i \end{bmatrix} \in \mathbb{R}^{1+d},$$

where $c_i \in \mathbb{R}$ is the i-th entry of the vector c and $a_i \in \mathbb{R}^N$ is the i-th column of the matrix A.

From a centralized perspective, in the classical simplex method, a set of columns (which for problems in standard form are treated as a basis), is iteratively updated until an optimal solution of problem (4.7) is found. At each iteration, a *leaving* column exits the basis and is replaced by an *entering* column. The Distributed Simplex algorithm extends the (centralized) simplex method. Agents are assumed to initially know only a subset of the problem columns. Informally, at every communication round, each agent builds up a (small) local LP with a subset of the problem columns (namely, the old basis and the bases collected from neighbors). Then, the local LP is solved, a basis associated to the optimal solution is found and is sent to neighbors. It can be shown that the evolution of the Distributed Simplex algorithm applied to problem (4.7) is tightly linked to the evolution of the Constraints Consensus algorithm applied to the dual of problem (4.7) (see [20, Proposition 5.3]).

4.2 Constraints Consensus for Convex and Abstract Programs

In this section, we describe the Constraints Consensus algorithm for more general set-ups than problem (4.1). Formally, assume N agents aim to cooperatively solve the convex program

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } & \mathbf{x} \in \bigcap_{i=1}^{N} X_i, \end{aligned} \tag{4.8}$$

where $c \in \mathbb{R}^d$ is the cost vector and the sets X_i are subsets of $\mathbb{R}^d$, for all $i \in \{1, \ldots, N\}$. Problem (4.8) is in the common-cost form (cf. Section 1.2.2), and we suppose that, for all i, the set X_i is known by agent i only and that the cost vector c is globally known. Notice that the linear cost function in problem (4.8) results in no loss of generality, as discussed in Remark 4.1. We make the following assumption.

Assumption 4.5. Problem (4.8) is feasible and the sets X_i are convex and compact, for all $i \in \{1, \ldots, N\}$. □

The Constraints Consensus algorithm applied to problem (4.8) can be formalized by extending the concept of basis (cf. Definition 4.1) so as to consider the (possible) nonlinear nature of the local constraints X_i. Formally, let $\mathbf{x}^\star$ be the lex-optimal solution of problem (4.8). Then, the collection of δ constraints $X_{\ell_1}, \ldots, X_{\ell_\delta}$, for some indices $\{\ell_1, \ldots, \ell_\delta\} \subseteq$

$\{1, \dots, N\}$, are a basis of (4.8) if $\mathbf{x}^\star$ is the lex-optimal solution of

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } & \mathbf{x} \in X_{\ell_h}, \quad h \in \{1, \dots, \delta\}, \end{aligned}$$

and if the collection of δ constraints is minimal (i.e., removing a constraint from the previous problem implies that the lex-optimal solution changes). We compactly denote the basis as the set $B = \bigcap_{h=1}^{\delta} X_{\ell_h}$. For feasible convex problems in the form (4.8), it holds $\delta \leq d$, whereas for linear programs, it holds $\delta = d$ (cf. Definition 4.1). The maximum δ for a given problem is called the *combinatorial dimension* of the problem. A more comprehensive discussion can be found in [1, 108].

Next, we describe the Constraints Consensus algorithm applied to convex programs in Algorithm 8, from the perspective of node i. Each agent i maintains a local solution estimate $\mathbf{x}_i^t$ and a local basis B_i^t, initialized to X_i. The algorithm looks similar to Algorithm 7, where the main difference is that general convex constraints are considered, instead of linear ones.

Algorithm 8 Constraints Consensus applied to convex problems

```
**Initialization**: $B_i^0 = X_i$

**Evolution**: for $t = 0, 1, ...$
    **Gather** $B_j^t$ from neighbors $j \in \mathcal{N}_i^t$
    **Compute** $\mathbf{x}_i^{t+1}$ as the lex-optimal solution of
```

$$\begin{aligned} \min_{\mathbf{x}} \;& c^\top \mathbf{x} \\ \text{subj. to } & \mathbf{x} \in X_i \\ & \mathbf{x} \in B_i^t \\ & \mathbf{x} \in B_j^t, \quad j \in \mathcal{N}_i^t \end{aligned} \tag{4.9}$$

```
    **Update** $B_i^{t+1}$ as a basis of (4.9)
```

Note that, as in Algorithm 7, we ask processors to use a lexicographic solver to handle possible non-uniqueness of the optimal solution. Algorithm 8 enjoys the same convergence properties of Algorithm 7, formalized next.

Theorem 4.6. Let Assumption 4.5 hold and let the communication graph be jointly strongly connected. Moreover, let $\mathbf{x}^\star$ be the lex-optimal solution of problem (4.8). Consider the sequences $\{\mathbf{x}_i^t\}_{t\geq 0}, i \in \{1,\dots,N\}$, generated by Algorithm 8. Then, for all $i \in \{1,\dots,N\}$, the following holds:

1. the cost sequence $\{c^\top \mathbf{x}_i^t\}_{t\geq 0}$ converges in finite time to the optimal cost $J^\star$ of (4.8);

2. the solution sequence $\{\mathbf{x}_i^t\}_{t\geq 0}$ converges in finite time to $\mathbf{x}^\star$. □

Theorem 4.6 can be proven by using arguments similar to the ones in Theorem 4.4, thus we omit the proof.

We highlight that, in practice, Algorithm 8 can be implemented when the constraint sets X_i are easy to communicate (e.g., when all of them have the same parametric form and they only differ for the parameters). In more difficult set-ups, polyhedral approximations of the local sets X_i can be communicated instead (cf. Section 4.3.1).

Remark 4.1. Algorithm 8 can be properly adapted to handle problems with nonlinear cost in the form

$$\begin{aligned} \min_{\mathbf{x}} \ & f(\mathbf{x}) \\ \text{subj. to } & \mathbf{x} \in \bigcap_{i=1}^{N} X_i, \end{aligned} \tag{4.10}$$

with $f : \mathbb{R}^d \to \mathbb{R}$ a convex cost function. By resorting to the epigraph form of (4.10), which is in the form (4.8), it can be shown that Algorithm 8 can be implemented by simply replacing the linear function in the local problem (4.9) with the nonlinear one and by increasing the maximum number of sets in the bases to $d+1$. □

The Constraints Consensus algorithm can handle more general problems than (4.8). Indeed, in [108], the algorithm has been formulated for general abstract programs (or LP-type problems), which include, as a special case, problems (4.1) and (4.8). We do not give the technical details of abstract programs, but we only mention that they are a generalization of linear programs, which capture numerous geometric optimization problems such as, e.g., computation of the smallest enclosing ball of a set of points. When the combinatorial dimension of the problem is known, the distributed algorithm [108] can be applied directly. Otherwise, if the *Helly number* of the problem is known, one can use the results in [3] to compute the combinatorial dimension of the problem.

4.3 Extensions

In this section, we discuss extensions of the Constraints Consensus algorithm.

4.3.1 Cutting-plane Consensus

Let us consider again the convex program (4.8). The *Cutting-plane Consensus* algorithm [19] is an extension of Algorithm 8, in which outer approximations of the local constraint sets X_i are communicated (instead of the sets X_i themselves). There are several situations in which this approach is desirable, such as, e.g., *(i)* when privacy must be preserved (so that agents do not want to share their own constraint with the other nodes), *(ii)* when it is expensive to send X_i, *(iii)* when there are infinitely many local constraints (e.g., robust, semi-infinite programming).

The Cutting-plane Consensus algorithm is based on a successive refinement of polyhedral approximations of the local sets X_i. In particular, agents repeatedly solve linear programs of the form

$$\begin{aligned} \min_{\mathbf{x}} \ & c^\top \mathbf{x} \\ \text{subj. to } \ & A\mathbf{x} \le b, \end{aligned} \tag{4.11}$$

where the feasible set $\{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} \le b\}$ is a (polyhedral) outer-approximation of $\bigcap_{i=1}^N X_i$. It is constructed by generating and exchanging a particular type of constraints, called *cutting planes*.[4]

The evolution of the Cutting-plane Consensus algorithm can be roughly summarized as follows. Each agent i first solves problem (4.11)

[4]A cutting plane is a half space $h \triangleq \{\mathbf{x} \in \mathbb{R}^d \mid a^\top \mathbf{x} \le b\}$ separating a query point $\mathbf{x}_q \in \mathbb{R}^d$ from a set X, i.e., such that $X \subset h$ and $\mathbf{x}_q \notin h$.

and finds an optimal solution $\mathbf{x}_q$. Then, it checks whether $\mathbf{x}_q$ belongs to its own constraint set X_i. If so, it sends to neighbors a basis associated to $\mathbf{x}_q$ (in terms of the approximated constraints). If not, it generates a new cutting plane, it computes an optimal solution of the new approximate problem and sends a basis to neighbors.

Differently from the Constraints Consensus algorithm, the Cutting-plane Consensus algorithm does not enjoy finite-time convergence, but instead it converges asymptotically. Also, we point out that the tie-break rule used in [19] (in case problem (4.11) has multiple optimal solutions) consists of the minimal 2-norm solution, instead of the lex-optimal solution.

4.3.2 Distributed Mixed-Integer Linear Programming via Cut Generation and Constraint Exchange

Mixed-integer linear programs (MILPs) are linear programs in which some of the variables are constrained to be integer, i.e.,

$$\begin{aligned} \min_{\mathbf{x}} \ & c^\top \mathbf{x} \\ \text{subj. to } \ & a_i^\top \mathbf{x} \le b_i, \qquad i \in \{1, \ldots, N\}, \\ & \mathbf{x} \in \mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R}, \end{aligned} \tag{4.12}$$

where d_Z and d_R are the dimensions of the integer and real variables, $d = d_Z + d_R$, $c \in \mathbb{R}^d$ and $a_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$ for all $i \in \{1, \ldots, N\}$.

It is well known that MILPs are NP-hard problems, which makes problem (4.12) difficult to solve. In [133] and in [134] distributed algorithms are proposed, with finite-time convergence, for the solution of problem (4.12). They are based on a constraint exchange approach as in Constraints Consensus, but appropriate additional constraints (cutting planes, cf. also Section 4.3.1) are generated throughout the algorithm evolution.

Let $P \triangleq \{\mathbf{x} \in \mathbb{R}^d \mid a_i^\top \mathbf{x} \le b_i \text{ for all } i\}$ denote the polyhedron described by the inequality constraints of problem (4.12) and let $P_I \triangleq P \cap (\mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R})$ denote the feasible set of problem (4.12). An important feature of problem (4.12) is that it has the same optimal cost of the linear program

$$\begin{aligned} &\min_{\mathbf{x}} \ c^\top \mathbf{x} \\ &\text{subj. to } \mathbf{x} \in \operatorname{conv}(P_I), \end{aligned} \tag{4.13}$$

where $\operatorname{conv}(P_I)$ is the convex hull of P_I. Moreover, the optimal solution set of problem (4.12) is contained in the optimal solution set of (4.13). In order to solve the original MILP (4.12), the algorithms in [134] produce successive approximations of $\operatorname{conv}(P_I)$ by generating two types of cutting planes: *(i)* mixed-integer Gomory cuts and *(ii)* cost-based cuts. We do not provide the technical details on the algorithms, but we only point out that, as in Constraints Consensus, the algorithms work under asynchronous and unreliable communication and enjoy finite-time convergence.

4.3.3 Other extensions

In this subsection, we briefly mention other extensions of the algorithms presented in this chapter.

Robust optimization is the field of optimization that considers problems in which the problem data is uncertain. Typical approaches to tackle an uncertain problem consider the worst case of the uncertain parameters, giving rise to a *semi-infinite* optimization problem, i.e., with an infinite number of constraints. In [25], a distributed robust optimization algorithm is proposed, which is a *randomized* extension of the Constraints Consensus algorithm, to solve linear programs where the problem data is subject to uncertainty. The algorithm relies on a verification step (based on a random sampling of each agent of its local uncertain constraint set), and on the deterministic solution of a local version of the global semi-infinite problem.

In [107], the authors considered a *big-data* quadratic programming set-up emerging in several learning problems for cyber-physical networks, where the big-data keyword is due to the very high dimension of the optimization variable and of the training samples. For this class of big-data quadratic optimization problems, they proposed a distributed algorithm, obtained as an extension of the Constraints Consensus algorithm, which solves the problem up to an arbitrary tolerance ϵ. The algorithm is based on the notion of core-set used in geometric optimization to approximate the value function of a given set of points with a smaller subset of points. From an optimization point of view, a subset of active constraints is identified, whose number depends on the tolerance ϵ. The resulting approximate solution is such that an ϵ-relaxation of the constraints guarantees no constraint violation.

Submodular optimization is a special class of combinatorial optimization (in which the cost function is actually a *set function*) arising in several machine learning problems, but also in cooperative control of complex systems. In [132], a submodular minimization problem is considered. Agents can evaluate the cost function only for those sets including the agent itself. Then, by relying on a proper linear programming reformulation of the submodular problem (involving a huge number of variables), it is possible to devise a distributed algorithm based on a *column generation* approach, in which columns are generated through a local *greedy* algorithm.

4.4 Numerical Example

In this section, we provide a numerical example of the Constraints Consensus algorithm to highlight its main features.

We consider a network of $N = 30$ agents communicating over a fixed, directed, strongly connected graph generated according to an Erdős-Rényi random model with parameter $p = 0.1$.

We focus on the soft-margin SVM problem introduced in Section 1.3.3, where we consider a two-dimensional space, i.e., $d = 2$, and we recall that each agent i is assigned one training sample $(p_i, \ell_i) \in \mathbb{R}^2 \times \{-1, 1\}$. We suppose that the training samples are randomly picked from two bivariate gaussian distributions with covariance matrix equal to the identity matrix. A number of 15 agents are assigned to the first distribution, which has zero mean and is associated to the label $\ell_i = 1$, while the remaining agents are assigned to the second distribution, associated to the label $\ell_i = -1$ and with mean equal to $[3, 2]^\top$.

The goal for agents is to agree on an optimal solution of problem (1.8), which we recall here

$$\begin{aligned}\min_{w,b,\boldsymbol{\xi}} \ & \frac{1}{2}w^\top w + C\sum_{i=1}^N \xi_i \\ \text{subj. to } \ & \ell_i(w^\top p_i + b) \geq 1 - \xi_i, \qquad i \in \{1, \ldots, N\}, \\ & \boldsymbol{\xi} \geq 0,\end{aligned}$$

where the parameter C is set to 100. In the following we also denote the vector stacking all the optimization variables with $\mathbf{x}$. As discussed in Remark 4.1, in order to solve problem (1.8) with the Constraints Consensus algorithm, we implement the local optimization problems in Algorithm 8 with the cost function $f(\mathbf{x}) = 1/2 w^\top w + C\sum_{i=1}^N \xi_i$ and we allow up to $d+1$ constraints in the bases. To solve the lexmin optimization in (8), we solve a total of $d+1$ problems as follows. First, we obtain the optimal cost $f^\star$ of the problem. Then we add to the problem the constraint $f(x) = f^\star$ (in order to force the optimal cost) and we minimize the first component of the decision variable. We continue this procedure until we obtain the lex-optimal solution. Moreover, artificial box constraints $-M\mathbf{1} \leq w, b, \boldsymbol{\xi} \leq M\mathbf{1}$, with $M = 10$ (which we verified to be sufficiently large for this problem), are added to problem (1.8) in order to satisfy Assumption 4.5.

In our simulation, agents reached consensus on the lex-optimal solution of problem (1.8) in 10 communication rounds, as expected from the finite-time result of Theorem 4.6. In Figure 4.2 we show the convergence rate of Algorithm 8. In particular, we plot the difference between the cost of the solution estimates and the optimal cost $J^\star$ of problem (1.8), i.e., $f(\mathbf{x}_i^t) - J^\star$, for all i. Note that all the lines eventually approach zero.

In Figure 4.3 we show the maximum constraint value associated to the local solution estimates, i.e., for all i we plot the quantity

$$\max_{j\in\{1,\ldots,N\}} \left[1 - \ell_j((w_i^t)^\top p_j + b_i^t)\right].$$

Notice that the algorithm evolves in an outer-approximation fashion, that is, the solution estimates are infeasible for problem (1.8) until the optimal solution is found. This can also be seen by noting in Figure 4.2 that the costs associated to the intermediate solution estimates are lower than the optimal cost of the problem.

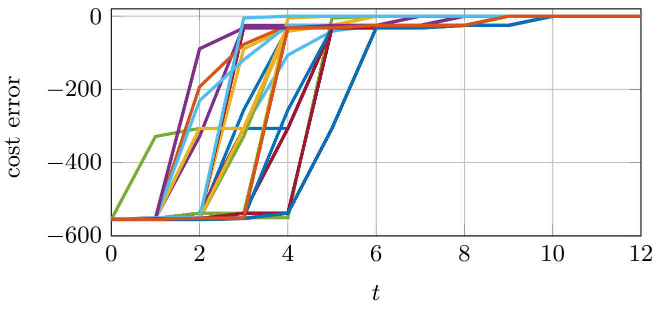


Figure 4.2: Evolution of the cost error $f(\mathbf{x}_i^t) - J^\star$ of local solution estimates $\mathbf{x}_i^t$ for the Constraints Consensus algorithm. Each line refers to an agent in the network.

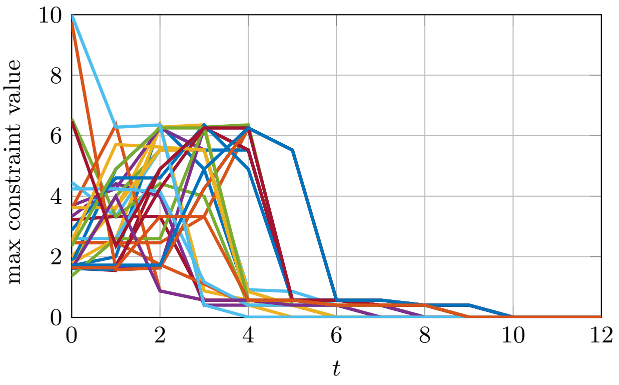


Figure 4.3: Evolution of the maximum value of the constraints for the solution estimate $\mathbf{x}_i^t$ computed by each agent in the Constraints Consensus algorithm. Each line refers to an agent in the network.

In Figure 4.4 we show the distance of the local solution estimates from the lex-optimal solution $\mathbf{x}^\star$ of problem (1.8), i.e., $\|\mathbf{x}_i^t - \mathbf{x}^\star\|$, for all i.

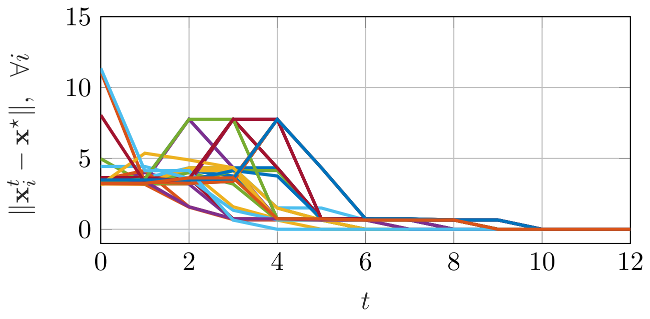


Figure 4.4: Evolution of the distance $\|\mathbf{x}_i^t - \mathbf{x}^\star\|$ of the local solution estimates $\mathbf{x}_i^t$ from the lex-optimal solution $\mathbf{x}^\star$ for the Constraints Consensus algorithm. Each line refers to an agent in the network.

Concluding Remarks

In this survey, we considered a distributed optimization framework arising in modern cyber-physical networks, in which computing units have only a partial knowledge of a global optimization problem and must solve it through local computation and communication without any central coordinator. First, we introduced main optimization set-ups addressed in distributed optimization (i.e., cost-coupled, common-cost, and constraint-coupled), and motivated them with relevant estimation, learning, decision and control applications arising in smart networks. Then, we reviewed three main approaches to design distributed optimization algorithms, namely (primal) consensus-based, duality-based and constraint-exchange methods, and provided a theoretical analysis under simplified communication assumptions and/or problem set-ups. To highlight the behavior of the presented algorithms, the theoretical results are also equipped with numerical examples.

Appendices

A

Centralized Optimization Methods

A.1 Gradient Method

Consider the following unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}), \tag{A.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$. The gradient method is an iterative algorithm given by

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \gamma^t \nabla f(\mathbf{x}^t), \tag{A.2}$$

where $t \geq 0$ denotes the iteration counter and γ^t is the step-size. The following result states the convergence of the gradient method for constant step-size.

Proposition A.1 ([9, Proposition 1.2.3])**.** Assume that f is a $\mathcal{C}^1$ function with Lipschitz continuous gradient ∇f with constant L. Let the step-size be constant, i.e., $\gamma^t = \gamma$, for all $t \geq 0$, and such that $0 < \gamma < 2/L$. Then, every limit point of the sequence $\{\mathbf{x}^t\}_{t\geq 0}$ generated by the gradient method (A.2), is a stationary point of problem (A.1), i.e., there exists a subset of indices $\mathcal{K} \subseteq \mathbb{N}$ such that

$$\lim_{\mathcal{K}\ni t\to\infty} \|\mathbf{x}^t - \bar{\mathbf{x}}\| = 0,$$

where $\bar{\mathbf{x}}$ is a stationary point of (A.1). □

The previous result can be extended in several ways, e.g., with different step-size rules and adapted to constrained problems. We refer the interested reader to [9] and references therein.

A.2 Subgradient Method

Consider the following constrained optimization problem

$$\min_{\mathbf{x}\in X} f(\mathbf{x}), \tag{A.3}$$

with $f : \mathbb{R}^d \to \mathbb{R}$ a convex function and $X \subseteq \mathbb{R}^d$ a closed, convex set.

A vector $\widetilde{\nabla} f(\mathbf{x}) \in \mathbb{R}^d$ is called a subgradient of the convex function f at $\mathbf{x} \in \mathbb{R}^d$ if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \widetilde{\nabla} f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$$

for all $\mathbf{y} \in \mathbb{R}^d$. The (projected) subgradient method is the iterative algorithm given by

$$\mathbf{x}^{t+1} = \mathcal{P}_X\left(\mathbf{x}^t - \gamma^t \widetilde{\nabla} f(\mathbf{x}^t)\right), \tag{A.4}$$

where $t \geq 0$ denotes the iteration counter, γ^t is the step-size, $\widetilde{\nabla} f(\mathbf{x}^t)$ denotes a subgradient of f at $\mathbf{x}^t$, and $\mathcal{P}_X(\,\cdot\,)$ is the Euclidean projection onto X.

Assumption A.1 (Diminishing Step-size)**.** The step-size sequence $\{\gamma^t\}_{t\geq 0}$ is such that $\gamma^t \geq 0$ and satisfies

$$\lim_{t\to\infty} \gamma^t = 0, \qquad \sum_{t=0}^{\infty} \gamma^t = \infty, \qquad \sum_{t=0}^{\infty} (\gamma^t)^2 < \infty. \qquad \square$$

The following proposition formally states the convergence of the subgradient method (A.4).

Proposition A.2 ([10, Proposition 3.2.6])**.** Assume that all the subgradients of f are bounded at each $\mathbf{x} \in X$. Moreover, assume the optimal solution set of problem (A.3) is not empty. Let the step-size γ^t satisfy Assumption A.1. Then, the sequence $\{\mathbf{x}^t\}_{t\geq 0}$ generated by the subgradient method (A.4) converges to an optimal solution $\mathbf{x}^\star$ of problem (A.3), i.e.,

$$\lim_{t\to\infty} \|\mathbf{x}^t - \mathbf{x}^\star\| = 0, \qquad \lim_{t\to\infty} \|f(\mathbf{x}^t) - f^\star\| = 0. \qquad \square$$

A.3 Lagrangian Duality and Dual Subgradient Method

Consider a constrained optimization problem, addressed as primal problem, having the form

$$\begin{aligned}\min_{\mathbf{x}\in X}\ & f(\mathbf{x})\\ \text{subj. to } & \mathbf{g}(\mathbf{x}) \leq \mathbf{0},\end{aligned} \tag{A.5}$$

where $X \subseteq \mathbb{R}^d$ is a convex, 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 $\mathbf{g}_s : \mathbb{R}^d \to \mathbb{R}$, $s \in \{1, \dots, S\}$, is a convex (scalar) function.

The following optimization problem

$$\begin{aligned}\max_{\boldsymbol{\mu}}\ & q(\boldsymbol{\mu})\\ \text{subj. to } & \boldsymbol{\mu} \geq \mathbf{0}\end{aligned} \tag{A.6}$$

is called the dual of problem (A.5), 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., $q(\boldsymbol{\mu}) = \min_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu})$. It can be shown that the domain of q (i.e., the set of $\boldsymbol{\mu}$ such that $q(\boldsymbol{\mu}) > -\infty$) is convex and that q is concave on its domain. A vector $\bar{\boldsymbol{\mu}} \in \mathbb{R}^S$ is said to be a Lagrange multiplier if it holds $\bar{\boldsymbol{\mu}} \geq \mathbf{0}$ and

$$\inf_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \bar{\boldsymbol{\mu}}) = \inf_{\mathbf{x}\in X \,:\, \mathbf{g}(\mathbf{x})\leq \mathbf{0}} f(\mathbf{x}).$$

It can be shown that the following inequality holds [9]

$$\inf_{\mathbf{x}\in X} \sup_{\boldsymbol{\mu}\geq\mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}) \geq \sup_{\boldsymbol{\mu}\geq\mathbf{0}} \inf_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}), \tag{A.7}$$

which is called weak duality. When in (A.7) the equality holds, then we say that strong duality holds and, thus, solving the primal problem (A.5) is equivalent to solving its dual formulation (A.6). In this case the right-hand-side problem in (A.7) is referred to as *saddle-point problem* of (A.5).

Definition A.1. A pair $(\mathbf{x}^\star, \boldsymbol{\mu}^\star)$ is called a primal-dual optimal solution of problem (A.5) if $\mathbf{x}^\star \in X$ and $\boldsymbol{\mu}^\star \geq \mathbf{0}$, 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 $\boldsymbol{\mu} \geq \mathbf{0}$. $\square$

Given the dual function q, an important property is as follows. A subgradient of $-q$ at a given $\bar{\boldsymbol{\mu}}$ can be efficiently computed as $g(\bar{\mathbf{x}})$, where $\bar{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}\in X} f(\mathbf{x}) + \bar{\boldsymbol{\mu}}^\top g(\mathbf{x})$ (see [9, Section 6] for further details). Then, a subgradient method to solve the dual problem (A.6) reads

$$\begin{aligned}\mathbf{x}^{t+1} &= \operatorname*{argmin}_{\mathbf{x}\in X} f(\mathbf{x}) + (\boldsymbol{\mu}^t)^\top g(\mathbf{x}) \\ \boldsymbol{\mu}^{t+1} &= \mathcal{P}_{\boldsymbol{\mu}\geq 0}\Big(\boldsymbol{\mu}^t + \gamma^t g(\mathbf{x}^{t+1})\Big),\end{aligned}$$

where γ^t is a suitable step-size and $\boldsymbol{\mu}^0 \geq 0$ is arbitrary.

A.4 ADMM Algorithm

In this section, we review the Alternating Direction Method of Multipliers (ADMM) following [12, Section 3.4]. Consider the following optimization problem

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^d}\ & G_1(\mathbf{x}) + G_2(A\mathbf{x}) \\ \text{subj. to}\ & \mathbf{x}\in C_1,\ A\mathbf{x}\in C_2,\end{aligned} \tag{A.8}$$

where $G_1 : \mathbb{R}^d \to \mathbb{R}$ and $G_2 : \mathbb{R}^S \to \mathbb{R}$ are convex functions, A is a $S \times d$ matrix, and $C_1 \subseteq \mathbb{R}^d$ and $C_2 \subseteq \mathbb{R}^S$ are nonempty, closed convex sets. We assume that the optimal solution set $X^\star$ of problem (A.8) is nonempty. Furthermore, either C_1 is bounded or else $A^\top A$ is invertible.

Problem (A.8) can be equivalently rewritten as

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^d,\mathbf{z}\in\mathbb{R}^S}\ & G_1(\mathbf{x}) + G_2(\mathbf{z}) \\ \text{subj. to}\ & A\mathbf{x} = \mathbf{z}, \\ & \mathbf{x}\in C_1,\ \mathbf{z}\in C_2.\end{aligned} \tag{A.9}$$

Let $\boldsymbol{\lambda} \in \mathbb{R}^S$ be a multiplier associated to the equality constraint $A\mathbf{x} = \mathbf{z}$ and introduce the *augmented* Lagrangian of problem (A.9)

$$\mathcal{L}_\rho(\mathbf{x},\mathbf{z},\boldsymbol{\lambda}) = G_1(\mathbf{x}) + G_2(\mathbf{z}) + \boldsymbol{\lambda}^\top(A\mathbf{x}-\mathbf{z}) + \frac{\rho}{2}\|A\mathbf{x}-\mathbf{z}\|^2$$

where $\rho > 0$ is a penalty parameter. The ADMM algorithm is an iterative procedure in which at each iteration $t \geq 0$, the following steps are performed

$$\mathbf{x}^{t+1} = \underset{\mathbf{x} \in C_1}{\operatorname{argmin}} \ \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}^t, \boldsymbol{\lambda}^t) \tag{A.10a}$$

$$\mathbf{z}^{t+1} = \underset{\mathbf{z} \in C_2}{\operatorname{argmin}} \ \mathcal{L}_\rho(\mathbf{x}^{t+1}, \mathbf{z}, \boldsymbol{\lambda}^t) \tag{A.10b}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho\,(A\mathbf{x}^{t+1} - \mathbf{z}^{t+1}), \tag{A.10c}$$

where the initialization of the variables $\mathbf{z}^0$ and $\boldsymbol{\lambda}^0$ can be arbitrary.

The ADMM algorithm is very similar to dual ascent and to the Method of Multipliers (MM): it consists of an $\mathbf{x}$-minimization, a $\mathbf{z}$-minimization, and a dual variable update. As in the method of multipliers, the dual variable update uses a step-size equal to the augmented Lagrangian parameter ρ. In the MM, the augmented Lagrangian $\mathcal{L}_\rho$ is minimized jointly with respect to the two primal variables. In ADMM, on the other hand, $\mathbf{x}$ and $\mathbf{z}$ are updated in an alternating or sequential fashion, which accounts for the term *alternating direction*.

Proposition A.3 ([12, Proposition 4.2])**.** Consider a sequence $\{\mathbf{x}^t, \mathbf{z}^t, \boldsymbol{\lambda}^t\}_{t\geq 0}$ generated by the ADMM algorithm (A.10). Then, the generated sequence is bounded and every limit point of $\{\mathbf{x}^t\}_{t\geq 0}$ is an optimal solution of problem (A.8). Furthermore, the sequence $\{\boldsymbol{\lambda}^t\}_{t\geq 0}$ converges to an optimal solution of the dual of problem (A.8). □

In [18] a more general problem set-up for ADMM is considered. Specifically, let us consider a two-variable problem defined as

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^d, \mathbf{z} \in \mathbb{R}^S} \quad & G_1(\mathbf{x}) + G_2(\mathbf{z}) \\ \text{subj. to} \quad & A\mathbf{x} + B\mathbf{z} + c = 0 \\ & \mathbf{x} \in C_1, \ \mathbf{z} \in C_2. \end{aligned} \tag{A.11}$$

with $A \in \mathbb{R}^{p \times d}$, $B \in \mathbb{R}^{p \times S}$ and $c \in \mathbb{R}^{p \times 1}$. Then, the ADMM algorithm applied to problem (A.11) reads as follows

$$\mathbf{x}^{t+1} = \underset{\mathbf{x} \in C_1}{\operatorname{argmin}} \ \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}^t, \boldsymbol{\lambda}^t) \tag{A.12a}$$

$$\mathbf{z}^{t+1} = \operatorname*{argmin}_{\mathbf{z} \in C_2} \mathcal{L}_\rho(\mathbf{x}^{t+1}, \mathbf{z}, \boldsymbol{\lambda}^t) \tag{A.12b}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho\,(A\mathbf{x}^{t+1} + B\mathbf{z}^{t+1} + c), \tag{A.12c}$$

where the augmented Lagrangian is defined as

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = G_1(\mathbf{x}) + G_2(\mathbf{z}) + \boldsymbol{\lambda}^\top (A\mathbf{x} + B\mathbf{z} + c) + \frac{\rho}{2}\|A\mathbf{x} + B\mathbf{z} + c\|^2.$$

B

Consensus Over Networks

Consensus and distributed averaging are fundamental building blocks in distributed optimization.

We introduce the consensus problem for a group of N agents that considers conditions under which, using a certain message-passing protocol, the local variables of each agent converge to the same value. There exist several results related to the convergence of local variables to a common value using various information exchange protocols among agents.

B.1 Average Consensus over Static Networks

One of the most used models for consensus is based on the following discrete-time iteration: to generate an estimate at iteration $t+1$, agent i forms a convex combination of its current estimate $\mathbf{z}_i^t$ with the estimates received from other agents as

$$\mathbf{z}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij} \, \mathbf{z}_j^t, \tag{B.1}$$

where a_{ij} denotes a (positive) weight that agent i assigns to each neighbor j, and we recall that $\mathcal{N}_i$ is the set of neighbors of agent i in the (static) undirected communication graph. The weights a_{ij} are set to zero if i and j are not neighbors in the communication graph $\mathcal{G}$ and are doubly stochastic, i.e., they satisfy $\sum_{j=1}^N a_{ij} = 1$, for all $i \in \{1, \dots, N\}$, and $\sum_{i=1}^N a_{ij} = 1$, for all $j \in \{1, \dots, N\}$.

The consensus algorithm can be written in an aggregate form by stacking all the agents' estimates in a single variable which evolves according to

$$\mathbf{z}^{t+1} = \begin{bmatrix} \mathbf{z}_1^{t+1} \\ \vdots \\ \mathbf{z}_N^{t+1} \end{bmatrix} = A\mathbf{z}^t, \tag{B.2}$$

where A is a matrix whose (i, j)-th entry is a_{ij} for all $i, j \in \{1, \dots, N\}$.

A useful property of doubly stochastic matrices is the following. Given A doubly stochastic, it holds

$$\|A\mathbf{z} - \bar{\mathbf{z}}\| \leq \sigma_A \|\mathbf{z} - \bar{\mathbf{z}}\|,$$

where $\bar{\mathbf{z}} \triangleq \frac{1}{N}\sum_{i=1}^N \mathbf{z}_i$ and σ_A is the spectral radius of $A - \mathbf{1}\mathbf{1}^\top/N$. It can be proven (see [148]) that if the graph is connected and A is doubly stochastic, then $\sigma_A \in (0, 1)$, and specifically $\sigma_A = \max\{|\lambda_2|, |\lambda_N|\}$, where λ_h denotes the h-th largest eigenvalue of A.

Theorem B.1. Let $\mathcal{G}$ be a connected graph and let a_{ij}, $i, j \in \{1, \dots, N\}$ be doubly stochastic weights matching the graph. Then, the sequences $\{\mathbf{z}_i^t\}_{t\geq 0}$, $i \in \{1, \dots, N\}$, generated by (B.1) satisfy

$$\lim_{t\to\infty} \|\mathbf{z}_i^t - \bar{\mathbf{z}}^0\| = 0,$$

for all $i \in \{1, \dots, N\}$, where $\bar{\mathbf{z}}^0 = \frac{1}{N}\sum_{i=1}^N \mathbf{z}_i^0$. □

Several extensions of the basic consensus algorithm (B.1) exist. For instance, one can consider time-varying networks that have some long-term connectivity properties. The consensus algorithm needs to be adapted to accommodate the time-varying network by considering time-varying weights a_{ij}^t. Also, it is possible to design a consensus algorithm that works under delays and is robust to packet losses. See [46] for a recent survey on this topic. Next, we describe another extension in which the consensus algorithm is tailored for directed networks.

B.2 Push-sum Consensus over Directed Networks

In this section we describe how the average consensus algorithm can be adapted to work on directed networks. This algorithm is known as push-sum algorithm and has been introduced in [7].

In directed networks is not always possible to construct a doubly stochastic matrix A, while a column stochastic matrix is always available. We use B to denote a column stochastic matrix, i.e., such that $\mathbf{1}^\top B = \mathbf{1}^\top$. Formally, the push-sum consensus reads

$$\phi_i^{t+1} = \sum_{j \in \mathcal{N}_i} b_{ij}\, \phi_j^t \tag{B.3a}$$

$$\mathbf{s}_i^{t+1} = \sum_{j \in \mathcal{N}_i} b_{ij}\, \mathbf{s}_j^t \tag{B.3b}$$

$$\mathbf{z}_i^{t+1} = \frac{\mathbf{s}_i^{t+1}}{\phi_i^{t+1}}, \tag{B.3c}$$

with the initial values $\phi_i^0 = 1$ for all $i \in \{1, \dots, N\}$.

The convergence of this scheme has been proven in [7], i.e., the sequences $\{\mathbf{z}_i^t\}_{t \geq 0}$, $i \in \{1, \dots, N\}$, generated by (B.3) satisfy

$$\lim_{t \to \infty} \|\mathbf{z}_i^t - \bar{\mathbf{z}}^0\| = 0,$$

for all $i \in \{1, \dots, N\}$, where $\bar{\mathbf{z}}^0 = \frac{1}{N} \sum_{i=1}^N \mathbf{z}_i^0$.

B.3 Dynamic Average Consensus Algorithm

In this section, we present a distributed algorithm to achieve dynamic average consensus that has been proposed in [162]. See also [59] for a very recent tutorial.

We consider a network of N agents in which each agent i is able to measure a local discrete-time signal $\{\mathbf{r}_i^t\}_{t \geq 0}$. The goal is to design a distributed algorithm that enables agents to eventually track the average of their signal $\mathbf{r}_i^t$, $i \in \{1, \dots, N\}$, by means of local communication only.

The dynamic consensus algorithm proposed in [162] consists in a consensus-based procedure in which each agent maintains a local estimate $\mathbf{z}_i^t$ of the average. The local estimate is iteratively updated according to

$$\mathbf{z}_i^{t+1} = \sum_{j \in \mathcal{N}_i} a_{ij}\, \mathbf{z}_j^t + \left(\mathbf{r}_i^{t+1} - \mathbf{r}_i^t\right), \tag{B.4}$$

where a_{ij} are entries of a doubly stochastic matrix.

If the input signals $\mathbf{r}_i^t$ asymptotically converge to a constant value, then the dynamic average consensus algorithm in (B.4) is guaranteed to converge, i.e., for all $i \in \{1, \ldots, N\}$, it holds

$$\lim_{t \to \infty} \|\mathbf{z}_i^t - \bar{\mathbf{r}}^t\| = 0,$$

where $\bar{\mathbf{r}}^t = \frac{1}{N} \sum_{i=1}^N \mathbf{r}_i^t$ for all $t \geq 0$.

The interested reader can find a rigorous treatment and a more comprehensive discussion on this class of algorithms in [162, 59].

C

Linear Programming

A Linear Program (LP) is an optimization problem with linear cost function and linear constraints:

$$\begin{aligned} &\min_{\mathbf{x}} \ c^\top \mathbf{x} \\ &\text{subj. to } a_k^\top \mathbf{x} \leq b_k, \quad k \in \{1, \dots, K\}, \end{aligned} \tag{C.1}$$

where $c \in \mathbb{R}^d$ is the cost vector and $a_k \in \mathbb{R}^d$ and $b_k \in \mathbb{R}$ describe K inequality constraints. In the subsequent discussion, we assume that $d \leq K$. The feasible set $\mathcal{X}$ of problem (C.1) is the set of vectors satisfying all the constraints, i.e.,

$$\mathcal{X} \triangleq \{\mathbf{x} \in \mathbb{R}^d \mid a_k^\top \mathbf{x} \leq b_k \text{ for all } K \in \{1, \dots, K\}\}.$$

Note that $\mathcal{X}$ is a polyhedron, for which the following definition of vertex can be given.

Definition C.1. A vector $\tilde{\mathbf{x}} \in \mathbb{R}^d$ is a vertex of $\mathcal{X}$ if there exists some $c \in \mathbb{R}^d$ such that $c^\top \tilde{\mathbf{x}} < c^\top \mathbf{x}$ for all $\mathbf{x} \in \mathcal{X}$ with $\mathbf{x} \neq \tilde{\mathbf{x}}$. $\square$

If problem (C.1) admits an optimal solution, it can be shown that there exists an optimal vertex, i.e., a vertex which is an optimal solution of the problem (see, e.g., [13, Theorem 2.7]). Let $\mathbf{x}^\star$ be an optimal vertex of problem (C.1). Then, it is a standard result in linear programming theory that there exists an index set $\{\ell_1, \dots, \ell_d\} \subset \{1, \dots, K\}$, with cardinality d, such that $\mathbf{x}^\star$ is the unique optimal vertex of the problem

$$\begin{aligned} &\min_{\mathbf{x}} \ c^\top \mathbf{x} \\ &\text{subj. to } a_{\ell_h}^\top \mathbf{x} \leq b_{\ell_h}, \quad h \in \{1, \dots, d\}, \end{aligned}$$

which is a relaxed version of problem (C.1) in which only d constraints are considered. In addition, the vectors $a_{\ell_h}, h \in \{1, \dots, d\}$ are linearly independent, so that they form a basis of $\mathbb{R}^d$. By analogy, the constraints $a_{\ell_h}^\top \mathbf{x} \leq b_{\ell_h}, h \in \{1, \dots, d\}$ are called a *basis* of the point $\mathbf{x}^\star$. Due to the optimality of $\mathbf{x}^\star$, we call it also a basis of problem (C.1). To compactly denote such basis, we introduce a matrix $P \in \mathbb{R}^{d \times d}$, obtained by stacking the row vectors $a_{\ell_h}^\top$, and a vector $q \in \mathbb{R}^d$, obtained by stacking the scalars b_{ℓ_h}, i.e.,

$$P = \begin{bmatrix} a_{\ell_1}^\top \\ \vdots \\ a_{\ell_d}^\top \end{bmatrix}, \qquad q = \begin{bmatrix} b_{\ell_1} \\ \vdots \\ b_{\ell_d} \end{bmatrix}.$$

Then, $\mathbf{x}^\star = P^{-1} q$, and we say that the tuple (P, q) is a basis of (C.1).

If problem (C.1) has multiple optimal solutions, we say that the LP is *dual degenerate*. In presence of dual degeneracy, it is not trivial to guarantee convergence of distributed algorithms to the same optimal solution. In order to overcome this issue, it is possible to rely on the lexicographic ordering of vectors. We now give some definitions.

Definition C.2. A vector $\mathbf{v} \in \mathbb{R}^n$ is said to be *lexicographically positive* (or *lex-positive*) if $\mathbf{v} \neq \mathbf{0}$ and the first non-zero component of $\mathbf{v}$ is positive. In symbols:

$$\mathbf{u} \overset{L}{>} \mathbf{0}.$$

A vector $\mathbf{u} \in \mathbb{R}^n$ is said to be *lexicographically larger* (resp., *smaller*) than another vector $\mathbf{v} \in \mathbb{R}^n$ if $\mathbf{u} - \mathbf{v}$ is lex-positive (resp. $\mathbf{v} - \mathbf{u}$ is lex-positive), or, equivalently, if $\mathbf{u} \neq \mathbf{v}$ and the first nonzero component of $\mathbf{u} - \mathbf{v}$ is positive (resp., negative). In symbols:

$$\mathbf{u} \overset{L}{>} \mathbf{v} \quad \text{or} \quad \mathbf{u} \overset{L}{<} \mathbf{v}.$$

Given a set of vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_r\}$, the lexicographic minimum is the element $\mathbf{v}_i$ such that $\mathbf{v}_j \overset{L}{>} \mathbf{v}_i$ for all $j \neq i$. In symbols:

$$\mathbf{v}_i = \operatorname{lexmin}\{\mathbf{v}_1, \dots, \mathbf{v}_r\}. \qquad \square$$

Now, consider the optimal solution set of problem (C.1), i.e., $\mathcal{X}^\star \triangleq \{\mathbf{x} \in \mathcal{X} \mid c^\top \mathbf{x} \leq c^\top \mathbf{x}' \text{ for all } \mathbf{x}' \in \mathcal{X}\} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the feasible set of problem (C.1). Among all the optimal solutions in $\mathcal{X}^\star$, it is possible to compute the lexicographically minimal one, i.e., lexmin$(\mathcal{S}^\star)$. It turns out that finding lexmin$(\mathcal{S}^\star)$ is equivalent to finding the (unique) optimal solution to a modified (non dual-degenerate) version of the original problem (C.1), where the cost vector c is perturbed to $c' = c + \Delta$, with Δ a lexicographic perturbation vector:

$$\Delta^\top = [\Delta_0 \;\; \Delta_0^2 \;\; \dots \;\; \Delta_0^d],$$

for a sufficiently small $\Delta_0 > 0$ (see [56]). Therefore, the lex-optimal solution of problem (C.1) is the *unique* optimal solution of the problem with perturbed cost

$$\begin{aligned} \min_{\mathbf{x}} \;\; & (c + \Delta)^\top \mathbf{x} \\ \text{subj. to } \; & a_k^\top \mathbf{x} \leq b_k, \qquad k \in \{1, \dots, K\}. \end{aligned} \tag{C.2}$$

Thus, the lex-optimal solution of problem (C.1) exists if and only if problem (C.2) admits an optimal solution. Moreover, the optimal solution of (C.2) is attained at a vertex of (C.1), therefore it is an optimal vertex of problem (C.1).

Acknowledgements

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

References

[1] Agarwal, P. K. and S. Sen (2001). “Randomized algorithms for geometric optimization problems”. *Combinatorial Optimization Dordrecht*. 9(1): 151–187.

[2] Alizadeh, M., X. Li, Z. Wang, A. Scaglione, and R. Melton (2012). “Demand-side management in the smart grid: Information processing for the power switch”. *IEEE Signal Processing Magazine*. 29(5): 55–67.

[3] Amenta, N. (1994). “Helly-type theorems and generalized linear programming”. *Discrete & Computational Geometry*. 12(3): 241–261.

[4] Bastianello, N., R. Carli, L. Schenato, and M. Todescato (2018). “A Partition-Based Implementation of the Relaxed ADMM for Distributed Convex Optimization over Lossy Networks”. In: *IEEE Conference on Decision and Control (CDC)*. 3379–3384.

[5] Beck, A. (2017). *First-order methods in optimization*. Vol. 25. SIAM.

[6] Beck, A., A. Nedic, A. Ozdaglar, and M. Teboulle (2014). “An $O(1/k)$ Gradient Method for Network Resource Allocation Problems”. *IEEE Transactions on Control of Network Systems*. 1(1): 64–73.

[7] Bénézit, F., V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli (2010). "Weighted gossip: Distributed averaging using non-doubly stochastic matrices". In: *IEEE International Symposium on Information Theory (ISIT)*. 1753–1757.

[8] Bertsekas, D. P. (1998). *Network optimization: continuous and discrete models*. Citeseer.

[9] Bertsekas, D. P. (1999). *Nonlinear programming*. Athena scientific.

[10] Bertsekas, D. P. (2015). *Convex Optimization Algorithms*. Athena Scientific.

[11] Bertsekas, D. P. and J. N. Tsitsiklis (2000). "Gradient convergence in gradient methods with errors". *SIAM Journal on Optimization*. 10(3): 627–642.

[12] Bertsekas, D. P. and J. N. Tsitsiklis (1989). *Parallel and distributed computation: numerical methods*. Vol. 23. Prentice Hall Englewood Cliffs, NJ.

[13] Bertsimas, D. and J. N. Tsitsiklis (1997). *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA.

[14] Bianchi, P., W. Hachem, and F. Iutzeler (2016). "A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization". *IEEE Transactions on Automatic Control*. 61(10): 2947–2957.

[15] Bianchi, P. and J. Jakubowicz (2013). "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization". *IEEE Transactions on Automatic Control*. 58(2): 391–405.

[16] Bof, N., R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo (2018). "Multi-Agent Newton-Raphson Optimization Over Lossy Networks". *IEEE Transactions on Automatic Control*.

[17] Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992). "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 144–152.

[18] Boyd, S., N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.* (2011). “Distributed optimization and statistical learning via the alternating direction method of multipliers”. *Foundations and Trends® in Machine learning.* 3(1): 1–122.

[19] Bürger, M., G. Notarstefano, and F. Allgöwer (2014). “A Polyhedral Approximation Framework for Convex and Robust Distributed Optimization”. *IEEE Transactions on Automatic Control.* 59(2): 384–395.

[20] Bürger, M., G. Notarstefano, F. Bullo, and F. Allgöwer (2012). “A distributed simplex algorithm for degenerate linear programs and multi-agent assignments”. *Automatica.* 48(9): 2298–2304.

[21] Camisa, A., I. Notarnicola, and G. Notarstefano (2018). “A Primal Decomposition Method with Suboptimality Bounds for Distributed Mixed-Integer Linear Programming”. In: *IEEE Conference on Decision and Control (CDC).* 3391–3396.

[22] Carli, R. and G. Notarstefano (2013). “Distributed partition-based optimization via dual decomposition”. In: *IEEE Conference on Decision and Control (CDC).* 2979–2984.

[23] Carli, R., G. Notarstefano, L. Schenato, and D. Varagnolo (2015). “Analysis of Newton-Raphson Consensus for multi-agent convex optimization under asynchronous and lossy communications”. In: *IEEE Conference on Decision and Control (CDC).* 418–424.

[24] Cattivelli, F. S. and A. H. Sayed (2010). “Diffusion LMS strategies for distributed estimation”. *IEEE Transactions on Signal Processing.* 58(3): 1035–1048.

[25] Chamanbaz, M., G. Notarstefano, and R. Bouffanais (2017). “Randomized Constraints Consensus for Distributed Robust Linear Programming”. *IFAC-PapersOnLine.* 50(1): 4973–4978.

[26] Chang, T.-H. (2016). “A proximal dual consensus ADMM method for multi-agent constrained optimization”. *IEEE Transactions on Signal Processing.* 64(14): 3719–3734.

[27] Chang, T.-H., M. Hong, and X. Wang (2015). “Multi-agent distributed optimization via inexact consensus ADMM”. *IEEE Transactions on Signal Processing.* 63(2): 482–497.

[28] Chang, T.-H., A. Nedić, and A. Scaglione (2014). "Distributed constrained optimization by consensus-based primal-dual perturbation method". *IEEE Transactions on Automatic Control.* 59(6): 1524–1538.

[29] Chen, J. and A. H. Sayed (2012). "Diffusion adaptation strategies for distributed optimization and learning over networks". *IEEE Transactions on Signal Processing.* 60(8): 4289–4305.

[30] Cherukuri, A. and J. Cortés (2015). "Distributed generator coordination for initialization and anytime optimization in economic dispatch". *IEEE Transactions on Control of Network Systems.* 2(3): 226–237.

[31] Cherukuri, A. and J. Cortés (2016). "Initialization-free distributed coordination for economic dispatch under varying loads and generator commitment". *Automatica.* 74: 183–193.

[32] Di Lorenzo, P. and G. Scutari (2015). "Distributed Nonconvex Optimization over Networks". In: *IEEE Intern. Conf. on Comput. Advances in Multi-Sensor Adaptive Process. (CAMSAP).* 229–232.

[33] Di Lorenzo, P. and G. Scutari (2016). "Next: In-network nonconvex optimization". *IEEE Transactions on Signal and Information Processing over Networks.* 2(2): 120–136.

[34] Dinh, Q. T., I. Necoara, and M. Diehl (2013). "A dual decomposition algorithm for separable nonconvex optimization using the penalty function framework". In: *IEEE Conference on Decision and Control (CDC).* 2372–2377.

[35] Doan, M. D., M. Diehl, T. Keviczky, and B. De Schutter (2017). "A Jacobi decomposition algorithm for distributed convex optimization in distributed model predictive control". *IFAC-PapersOnLine.* 50(1): 4905–4911.

[36] Duchi, J. C., A. Agarwal, and M. J. Wainwright (2012). "Dual averaging for distributed optimization: Convergence analysis and network scaling". *IEEE Transactions on Automatic control.* 57(3): 592–606.

[37] Ebenbauer, C., S. Michalowsky, V. Grushkovskaya, and B. Gharesifard (2017). “Distributed optimization over directed graphs with the help of Lie brackets”. *IFAC-PapersOnLine*. 50(1): 15343–15348.

[38] Eisen, M., A. Mokhtari, and A. Ribeiro (2017). “Decentralized quasi-Newton methods”. *IEEE Transactions on Signal Processing*. 65(10): 2613–2628.

[39] Erseghe, T. (2012). “A Distributed and Scalable Processing Method Based Upon ADMM”. *IEEE Signal Processing Letters*. 19(9): 563–566.

[40] Falsone, A., K. Margellos, S. Garatti, and M. Prandini (2017). “Dual decomposition for multi-agent distributed optimization with coupling constraints”. *Automatica*. 84: 149–158.

[41] Farina, F., A. Garulli, and A. Giannitrapani (2018). “Distributed Interpolatory Algorithms for Set Membership Estimation”. *IEEE Transactions on Automatic Control*.

[42] Farina, F., A. Garulli, A. Giannitrapani, and G. Notarstefano (2019). “A Distributed Asynchronous Method of Multipliers for Constrained Nonconvex Optimization”. *Automatica*. 103: 243–253.

[43] Gharesifard, B. and J. Cortés (2014). “Distributed continuous-time convex optimization on weight-balanced digraphs”. *IEEE Transactions on Automatic Control*. 59(3): 781–786.

[44] Giselsson, P., M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer (2013). “Accelerated gradient methods and dual decomposition in distributed model predictive control”. *Automatica*. 49(3): 829–833.

[45] Giselsson, P. and A. Rantzer (2018). *Large-scale and Distributed Optimization*. Vol. 2227. Springer.

[46] Hadjicostis, C. N., A. D. Domínguez-García, and T. Charalambous (2018). “Distributed Averaging and Balancing in Network Systems: with Applications to Coordination and Control”. *Foundations and Trends® in Systems and Control*. 5(2-3): 99–292.

[47] Hale, M. T., A. Nedić, and M. Egerstedt (2017). “Asynchronous multiagent primal-dual optimization”. *IEEE Transactions on Automatic Control*. 62(9): 4421–4435.

[48] Hatanaka, T., N. Chopra, T. Ishizaki, and N. Li (2018). “Passivity-based distributed optimization with communication delays using PI consensus algorithm”. *IEEE Transactions on Automatic Control*.

[49] Hochhaus, S. and M. T. Hale (2018). “Asynchronous Distributed Optimization with Heterogeneous Regularizations and Normalizations”. In: *IEEE Conference on Decision and Control (CDC)*. 4232–4237.

[50] Iutzeler, F., P. Bianchi, P. Ciblat, and W. Hachem (2016). “Explicit convergence rate of a distributed alternating direction method of multipliers”. *IEEE Transactions on Automatic Control*. 61(4): 892–904.

[51] Jakovetić, D., D. Bajovic, N. Krejic, and N. K. Jerinkic (2016). “Distributed Gradient Methods with Variable Number of Working Nodes.” *IEEE Transactions on Signal Processing*. 64(15): 4080–4095.

[52] Jakovetić, D., J. M. Moura, and J. Xavier (2015). “Linear Convergence Rate of a Class of Distributed Augmented Lagrangian Algorithms”. *IEEE Transactions on Automatic Control*. 60(4): 922–936.

[53] Jakovetić, D., J. Xavier, and J. M. Moura (2011). “Cooperative convex optimization in networked systems: Augmented Lagrangian algorithms with directed gossip communication”. *IEEE Transactions on Signal Processing*. 59(8): 3889–3902.

[54] Jakovetić, D., J. Xavier, and J. M. Moura (2014). “Fast distributed gradient methods”. *IEEE Transactions on Automatic Control*. 59(5): 1131–1146.

[55] Johansson, B., M. Rabi, and M. Johansson (2009). “A randomized incremental subgradient method for distributed optimization in networked systems”. *SIAM Journal on Optimization*. 20(3): 1157–1170.

[56] Jones, C. N., E. C. Kerrigan, and J. M. Maciejowski (2007). “Lexicographic perturbation for multiparametric linear programming with applications to control”. *Automatica*. 43(10): 1808–1816.

[57] Kia, S. S. (2017). “Distributed optimal in-network resource allocation algorithm design via a control theoretic approach”. *Systems & Control Letters*. 107: 49–57.

[58] Kia, S. S., J. Cortés, and S. Martínez (2015). “Distributed Convex Optimization via Continuous-time Coordination Algorithms with Discrete-time Communication”. *Automatica*. 55: 254–264.

[59] Kia, S. S., B. Van Scoy, J. Cortés, R. A. Freeman, K. M. Lynch, and S. Martínez (2019). “Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms”. *IEEE Control Systems Magazine*. 39(3): 40–72.

[60] Kumar, S., R. Jain, and K. Rajawat (2017). “Asynchronous optimization over heterogeneous networks via consensus ADMM”. *IEEE Transactions on Signal and Information Processing over Networks*. 3(1): 114–129.

[61] Latafat, P., L. Stella, and P. Patrinos (2016). “New primal-dual proximal algorithm for distributed optimization”. In: *IEEE Conference on Decision and Control (CDC)*. IEEE. 1959–1964.

[62] Lee, S. and A. Nedić (2013). “Distributed random projection algorithm for convex optimization”. *IEEE Journal of Selected Topics in Signal Processing*. 7(2): 221–229.

[63] Lee, S. and A. Nedić (2016). “Asynchronous gossip-based random projection algorithms over networks”. *IEEE Transactions on Automatic Control*. 61(4): 953–968.

[64] Lee, S., A. Nedić, and M. Raginsky (2017). “Stochastic dual averaging for decentralized online optimization on time-varying communication graphs”. *IEEE Transactions on Automatic Control*. 62(12): 6407–6414.

[65] Lee, S., A. Ribeiro, and M. M. Zavlanos (2016). “Distributed continuous-time online optimization using saddle-point methods”. In: *IEEE Conf. on Decision and Control (CDC)*. 4314–4319.

[66] Li, N. and J. R. Marden (2013). "Designing games for distributed optimization". *IEEE Journal of Selected Topics in Signal Processing*. 7(2): 230–242.

[67] Lin, P., W. Ren, and Y. Song (2016). "Distributed multi-agent optimization subject to nonidentical constraints and communication delays". *Automatica*. 65: 120–131.

[68] Ling, Q., Y. Liu, W. Shi, and Z. Tian (2016). "Weighted ADMM for fast decentralized network optimization". *IEEE Transactions on Signal Processing*. 64(22): 5930–5942.

[69] Ling, Q. and A. Ribeiro (2014). "Decentralized Dynamic Optimization Through the Alternating Direction Method of Multipliers". *IEEE Transactions on Signal Processing*. 5(62): 1185–1197.

[70] Ling, Q., W. Shi, G. Wu, and A. Ribeiro (2015). "DLM: Decentralized linearized alternating direction method of multipliers". *IEEE Transactions on Signal Processing*. 63(15): 4051–4064.

[71] Liu, S., Z. Qiu, and L. Xie (2017). "Convergence rate analysis of distributed optimization with projected subgradient algorithm". *Automatica*. 83: 162–169.

[72] Lobel, I. and A. Ozdaglar (2011). "Distributed subgradient methods for convex optimization over random networks". *IEEE Transactions on Automatic Control*. 56(6): 1291–1306.

[73] Lu, J. and C. Y. Tang (2012). "Zero-gradient-sum algorithms for distributed convex optimization: The continuous-time case". *IEEE Transactions on Automatic Control*. 57(9): 2348–2354.

[74] Makhdoumi, A. and A. Ozdaglar (2017). "Convergence rate of distributed ADMM over networks". *IEEE Transactions on Automatic Control*. 62(10): 5082–5095.

[75] Margellos, K., A. Falsone, S. Garatti, and M. Prandini (2018). "Distributed constrained optimization and consensus in uncertain networks via proximal minimization". *IEEE Transactions on Automatic Control*. 63(5): 1372–1387.

[76] Mateos-Núnez, D. and J. Cortés (2017). "Distributed saddle-point subgradient algorithms with Laplacian averaging". *IEEE Transactions on Automatic Control*. 62(6): 2720–2735.

[77] Mateos, G., J. A. Bazerque, and G. B. Giannakis (2010). “Distributed sparse linear regression”. *IEEE Transactions on Signal Processing*. 58(10): 5262–5276.

[78] Michalowsky, S., B. Gharesifard, and C. Ebenbauer (2018). “On the Lie bracket approximation approach to distributed optimization: Extensions and limitations”. In: *IEEE European Control Conference (ECC)*. 119–124.

[79] Mokhtari, A., Q. Ling, and A. Ribeiro (2017). “Network Newton distributed optimization methods”. *IEEE Transactions on Signal Processing*. 65(1): 146–161.

[80] Mokhtari, A., W. Shi, Q. Ling, and A. Ribeiro (2016). “DQM: Decentralized Quadratically Approximated Alternating Direction Method of Multipliers”. *IEEE Transactions on Signal Processing*. 64(19): 5158–5173.

[81] Mota, J. F., J. M. Xavier, P. M. Aguiar, and M. Püschel (2013). “D-ADMM: A communication-efficient distributed algorithm for separable optimization”. *IEEE Transactions on Signal Processing*. 61(10): 2718–2723.

[82] Necoara, I. (2013). “Random coordinate descent algorithms for multi-agent convex optimization over networks”. *IEEE Transactions on Automatic Control*. 58(8): 2001–2012.

[83] Necoara, I. and V. Nedelcu (2015). “On linear convergence of a distributed dual gradient algorithm for linearly constrained separable convex problems”. *Automatica*. 55: 209–216.

[84] Necoara, I., V. Nedelcu, D. Clipici, and L. Toma (2017). “On fully distributed dual first order methods for convex network optimization”. *IFAC-PapersOnLine*. 50(1): 2788–2793.

[85] Necoara, I., V. Nedelcu, and I. Dumitrache (2011). “Parallel and distributed optimization methods for estimation and control in networks”. *Journal of Process Control*. 21(5): 756–766.

[86] Nedić, A. (2011). “Asynchronous broadcast-based convex optimization over a network”. *IEEE Transactions on Automatic Control*. 56(6): 1337–1351.

[87] Nedić, A. (2015). “Convergence rate of distributed averaging dynamics and optimization in networks”. *Foundations and Trends® in Systems and Control*. 2(1): 1–100.

[88] Nedić, A. and J. Liu (2018). “Distributed Optimization for Control”. *Annual Review of Control, Robotics, and Autonomous Systems*. 1: 77–103.

[89] Nedić, A. and A. Olshevsky (2015). “Distributed optimization over time-varying directed graphs”. *IEEE Transactions on Automatic Control*. 60(3): 601–615.

[90] Nedić, A. and A. Olshevsky (2016). “Stochastic gradient-push for strongly convex functions on time-varying directed graphs”. *IEEE Transactions on Automatic Control*. 61(12): 3936–3947.

[91] Nedić, A., A. Olshevsky, and M. G. Rabbat (2018a). “Network topology and communication-computation tradeoffs in decentralized optimization”. *Proceedings of the IEEE*. 106(5): 953–976.

[92] Nedić, A., A. Olshevsky, and W. Shi (2016). “A geometrically convergent method for distributed optimization over time-varying graphs”. In: *IEEE Conf. on Decision and Control (CDC)*. 1023–1029.

[93] Nedić, A., A. Olshevsky, and W. Shi (2017a). “Achieving geometric convergence for distributed optimization over time-varying graphs”. *SIAM Journal on Optimization*. 27(4): 2597–2633.

[94] Nedić, A., A. Olshevsky, W. Shi, and C. A. Uribe (2017b). “Geometrically convergent distributed optimization with uncoordinated step-sizes”. In: *American Control Conference (ACC)*. IEEE. 3950–3955.

[95] Nedić, A. and A. Ozdaglar (2009). “Distributed subgradient methods for multi-agent optimization”. *IEEE Transactions on Automatic Control*. 54(1): 48–61.

[96] Nedić, A., A. Ozdaglar, and P. A. Parrilo (2010). “Constrained Consensus and Optimization in Multi-Agent Networks”. *IEEE Transactions on Automatic Control*. 55(4): 922–938.

[97] Nedić, A., J.-S. Pang, G. Scutari, and Y. Sun (2018b). *Multi-agent Optimization*. Vol. 2224. Springer.

[98] Notarnicola, I., R. Carli, and G. Notarstefano (2018a). "Distributed Partitioned Big-Data Optimization via Asynchronous Dual Decomposition". *IEEE Transactions on Control of Network Systems*. 5(4): 1910–1919.

[99] Notarnicola, I., M. Franceschelli, and G. Notarstefano (2016). "A duality-based approach for distributed min-max optimization with application to demand side management". In: *IEEE Conference on Decision and Control (CDC)*. 1877–1882.

[100] Notarnicola, I., M. Franceschelli, and G. Notarstefano (2019). "A duality-based approach for distributed min-max optimization". *IEEE Transactions on Automatic Control*. 64(6): 2559–2566.

[101] Notarnicola, I. and G. Notarstefano (2017a). "A duality-based approach for distributed optimization with coupling constraints". In: *IFAC World Congress*. 14891–14896.

[102] Notarnicola, I. and G. Notarstefano (2017b). "Asynchronous distributed optimization via randomized dual proximal gradient". *IEEE Transactions on Automatic Control*. 62(5): 2095–2106.

[103] Notarnicola, I. and G. Notarstefano (2019). "Constraint-Coupled Distributed Optimization: a Relaxation and Duality Approach". *IEEE Transactions on Control of Network Systems*.

[104] Notarnicola, I., Y. Sun, G. Scutari, and G. Notarstefano (2017a). "Distributed big-data optimization via block communications". In: *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. 1–5.

[105] Notarnicola, I., Y. Sun, G. Scutari, and G. Notarstefano (2017b). "Distributed big-data optimization via block-iterative convexification and averaging". In: *IEEE Conference on Decision and Control (CDC)*. 2281–2288.

[106] Notarnicola, I., Y. Sun, G. Scutari, and G. Notarstefano (2018b). "Distributed big-data optimization via block-iterative gradient tracking". *arXiv preprint arXiv:1808.07252*.

[107] Notarstefano, G. (2015). "A core-set approach for distributed quadratic programming in big-data classification". In: *IEEE Conference on Decision and Control (CDC)*. 1372–1377.

[108] Notarstefano, G. and F. Bullo (2011). “Distributed Abstract Optimization via Constraints Consensus: Theory and Applications”. *IEEE Transactions on Automatic Control*. 56(10): 2247–2261.

[109] Palomar, D. P. and M. Chiang (2006). “A tutorial on decomposition methods for network utility maximization”. *IEEE Journal on Selected Areas in Communications*. 24(8): 1439–1451.

[110] Paul, H., J. Fliege, and A. Dekorsy (2013). “In-network-processing: Distributed consensus-based linear estimation”. *IEEE Communications Letters*. 17(1): 59–62.

[111] Qu, G. and N. Li (2016). “Harnessing smoothness to accelerate distributed optimization”. In: *IEEE Conf. on Decision and Control (CDC)*. 159–166.

[112] Qu, G. and N. Li (2017a). “Accelerated distributed Nesterov Gradient Descent for convex and smooth functions”. In: *IEEE Conf. on Decision and Control (CDC)*. 2260–2267.

[113] Qu, G. and N. Li (2017b). “Harnessing smoothness to accelerate distributed optimization”. *IEEE Transactions on Control of Network Systems*. 5(3): 1245–1260.

[114] Ram, S. S., A. Nedić, and V. V. Veeravalli (2010). “Distributed stochastic subgradient projection algorithms for convex optimization”. *Journal of optimization theory and applications*. 147(3): 516–545.

[115] Rawat, A. and N. Elia (2018). “Distributed Method of Multiplier for Coupled Lagrangian Problems: A Control Approach”. In: *American Control Conference (ACC)*. 6475–6480.

[116] Rawlings, J. B. and D. Q. Mayne (2009). “Model predictive control: Theory and design”.

[117] Richert, D. and J. Cortés (2015). “Robust distributed linear programming”. *IEEE Transactions on Automatic Control*. 60(10): 2567–2582.

[118] Schizas, I. D., A. Ribeiro, and G. B. Giannakis (2008). “Consensus in ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals”. *IEEE Transactions on Signal Processing*. 56(1): 350–364.

[119] Scutari, G. and Y. Sun (2019). "Distributed nonconvex constrained optimization over time-varying digraphs". *Mathematical Programming*. 176(1-2): 497–544.

[120] Shi, W., Q. Ling, G. Wu, and W. Yin (2015a). "A proximal gradient algorithm for decentralized composite optimization". *IEEE Transactions on Signal Processing*. 63(22): 6013–6023.

[121] Shi, W., Q. Ling, G. Wu, and W. Yin (2015b). "EXTRA: An Exact First-order Algorithm for Decentralized Consensus Optimization". *SIAM Journal on Optimization*. 25(2): 944–966.

[122] Shi, W., Q. Ling, K. Yuan, G. Wu, and W. Yin (2014). "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization". *IEEE Transactions on Signal Processing*. 62(7): 1750–1761.

[123] Simonetto, A. and H. Jamali-Rad (2016). "Primal recovery from consensus-based dual decomposition for distributed convex optimization". *Journal of Optimization Theory and Applications*. 168(1): 172–197.

[124] Simonetto, A., A. Koppel, A. Mokhtari, G. Leus, and A. Ribeiro (2017). "Decentralized prediction-correction methods for networked time-varying convex optimization". *IEEE Transactions on Automatic Control*. 62(11): 5724–5738.

[125] Srivastava, K. and A. Nedic (2011). "Distributed asynchronous constrained stochastic optimization". *IEEE Journal of Selected Topics in Signal Processing*. 5(4): 772–790.

[126] Stankovic, M. S., K. H. Johansson, and D. M. Stipanovic (2011). "Distributed seeking of Nash equilibria with applications to mobile sensor networks". *IEEE Transactions on Automatic Control*. 57(4): 904–919.

[127] Sun, Y. and G. Scutari (2017). "Distributed nonconvex optimization for sparse representation". In: *IEEE Intern. Conf. on Speech and Signal Process. (ICASSP)*. 4044–4048.

[128] Sun, Y., G. Scutari, and D. P. Palomar (2016). "Distributed Nonconvex Multiagent Optimization over Time-varying Networks". In: *IEEE Asilomar Conf. on Signals, Systems, and Computers*. 788–794.

[129] Sundararajan, A., B. Hu, and L. Lessard (2017). “Robust convergence analysis of distributed optimization algorithms”. In: *Allerton Conference on Communication, Control, and Computing (Allerton).* IEEE. 1206–1212.

[130] Tatarenko, T. and B. Touri (2017). “Non-convex distributed optimization”. *IEEE Transactions on Automatic Control.* 62(8): 3744–3757.

[131] Teixeira, A., E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson (2016). “The ADMM algorithm for distributed quadratic problems: Parameter selection and constraint preconditioning”. *IEEE Transactions on Signal Processing.* 64(2): 290–305.

[132] Testa, A., I. Notarnicola, and G. Notarstefano (2018). “Distributed Submodular Minimization over Networks: a Greedy Column Generation Approach”. In: *IEEE Conference on Decision and Control (CDC).* 4945–4950.

[133] Testa, A., A. Rucco, and G. Notarstefano (2017). “A finite-time cutting plane algorithm for distributed mixed integer linear programming”. In: *IEEE Conference on Decision and Control (CDC).* 3847–3852.

[134] Testa, A., A. Rucco, and G. Notarstefano (2019). “Distributed Mixed-Integer Linear Programming via Cut Generation and Constraint Exchange”. *IEEE Transactions on Automatic Control.*

[135] Todescato, M., G. Cavraro, R. Carli, and L. Schenato (2015). “A Robust Block-Jacobi Algorithm for Quadratic Programming under Lossy Communications”. *IFAC-PapersOnLine.* 48(22): 126–131.

[136] Tsianos, K. I., S. Lawlor, and M. G. Rabbat (2012). “Push-sum distributed dual averaging for convex optimization”. In: *IEEE Conference on Decision and Control (CDC).* 5453–5458.

[137] Varagnolo, D., F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato (2016). “Newton-Raphson consensus for distributed convex optimization”. *IEEE Transactions on Automatic Control.* 61(4): 994–1009.

[138] Wai, H.-T., J. Lafond, A. Scaglione, and E. Moulines (2017). "Decentralized Frank–Wolfe Algorithm for Convex and Nonconvex Problems". *IEEE Transactions on Automatic Control.* 62(11): 5522–5537.

[139] Wang, J. and N. Elia (2011). "A control perspective for centralized and distributed convex optimization". In: *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC).* 3800–3805.

[140] Wei, E. and A. Ozdaglar (2013). "On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers". In: *IEEE Global conference on signal and information processing (GlobalSIP).* 551–554.

[141] Wei, E., A. Ozdaglar, and A. Jadbabaie (2013a). "A distributed Newton method for network utility maximization–Part I: Algorithm". *IEEE Transactions on Automatic Control.* 58(9): 2162–2175.

[142] Wei, E., A. Ozdaglar, and A. Jadbabaie (2013b). "A distributed Newton method for network utility maximization–Part II: Convergence". *IEEE Transactions on Automatic Control.* 58(9): 2176–2188.

[143] Wu, T., K. Yuan, Q. Ling, W. Yin, and A. H. Sayed (2017). "Decentralized consensus optimization with asynchrony and delays". *IEEE Transactions on Signal and Information Processing over Networks.* 4(2): 293–307.

[144] Wu, X. and J. Lu (2019). "Fenchel dual gradient methods for distributed convex optimization over time-varying networks". *IEEE Transactions on Automatic Control.*

[145] Xi, C. and U. A. Khan (2017). "DEXTRA: A fast algorithm for optimization over directed graphs". *IEEE Transactions on Automatic Control.* 62(10): 4980–4993.

[146] Xi, C., V. S. Mai, R. Xin, E. H. Abed, and U. A. Khan (2018a). "Linear convergence in optimization over directed graphs with row-stochastic matrices". *IEEE Transactions on Automatic Control.* 63(10): 3558–3565.

[147] Xi, C., R. Xin, and U. A. Khan (2018b). "ADD-OPT: Accelerated distributed directed optimization". *IEEE Transactions on Automatic Control*. 63(5): 1329–1339.

[148] Xiao, L. and S. Boyd (2004). "Fast linear iterations for distributed averaging". *Systems & Control Letters*. 53(1): 65–78.

[149] Xie, P., K. You, R. Tempo, S. Song, and C. Wu (2018). "Distributed Convex Optimization with Inequality Constraints over Time-varying Unbalanced Digraphs". *IEEE Transactions on Automatic Control*. 63(12): 4331–4337.

[150] Xin, R. and U. A. Khan (2018). "A linear algorithm for optimization over directed graphs with geometric convergence". *IEEE Control Systems Letters*. 2(3): 325–330.

[151] Xu, J., S. Zhu, Y. C. Soh, and L. Xie (2015). "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes". In: *IEEE Conf. on Decision and Control (CDC)*. 2055–2060.

[152] Xu, J., S. Zhu, Y. C. Soh, and L. Xie (2018a). "A Bregman splitting scheme for distributed optimization over networks". *IEEE Transactions on Automatic Control*. 63(11): 3809–3824.

[153] Xu, J., S. Zhu, Y. C. Soh, and L. Xie (2018b). "Convergence of asynchronous distributed gradient methods over stochastic networks". *IEEE Transactions on Automatic Control*. 63(2): 434–448.

[154] Yang, B. and M. Johansson (2010). "Distributed optimization and games: A tutorial overview". In: *Networked Control Systems*. Springer. 109–148.

[155] Yang, S., Q. Liu, and J. Wang (2017). "A multi-agent system with a proportional-integral protocol for distributed constrained optimization". *IEEE Transactions on Automatic Control*. 62(7): 3461–3467.

[156] Yuan, K., Q. Ling, and W. Yin (2016). "On the convergence of decentralized gradient descent". *SIAM Journal on Optimization*. 26(3): 1835–1854.

[157] Zanella, F., D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato (2011). "Newton-Raphson consensus for distributed convex optimization". In: *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. 5917–5922.

[158] Zanella, F., D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato (2012). "Asynchronous Newton-Raphson Consensus for Distributed Convex Optimization". In: *IFAC Workshop on Distributed Estimation and Control in Networked Systems.*

[159] Zeng, J. and W. Yin (2018). "On nonconvex decentralized gradient descent". *IEEE Transactions on signal processing*. 66(11): 2834–2848.

[160] Zhang, Y. and M. M. Zavlanos (2018). "A Consensus-Based Distributed Augmented Lagrangian Method". In: *IEEE Conference on Decision and Control (CDC)*. 1763–1768.

[161] Zhu, H., G. B. Giannakis, and A. Cano (2009). "Distributed in-network channel decoding". *IEEE Transactions on Signal Processing*. 57(10): 3970–3983.

[162] Zhu, M. and S. Martínez (2010). "Discrete-time dynamic average consensus". *Automatica*. 46(2): 322–329.

[163] Zhu, M. and S. Martínez (2012). "On distributed convex optimization under inequality and equality constraints". *IEEE Transactions on Automatic Control*. 57(1): 151–164.