Distributed Partitioned Big-Data Optimization via Asynchronous Dual Decomposition

This is the post peer-review accepted manuscript of:

I. Notarnicola, R. Carli and G. Notarstefano, "Distributed Partitioned Big-Data Optimization via Asynchronous Dual Decomposition," in IEEE Transactions on Control of Network Systems, vol. 5, no. 4, pp. 1910-1919, Dec. 2018.

The published version is available online at:

https://doi.org/10.1109/TCNS.2017.2774010

Distributed Partitioned Big-Data Optimization via Asynchronous Dual Decomposition

Ivano Notarnicola[1], *Student Member, IEEE,* Ruggero Carli[2], *Member, IEEE,* and Giuseppe Notarstefano[1], *Member, IEEE*

***Abstract*—In this paper we consider a novel partitioned framework for distributed optimization in peer-to-peer networks. In several important applications the agents of a network have to solve an optimization problem with two key features: (i) the dimension of the decision variable depends on the network size, and (ii) cost function and constraints have a sparsity structure related to the communication graph. For this class of problems a straightforward application of existing consensus methods would show two inefficiencies: poor scalability and redundancy of shared information. We propose an asynchronous distributed algorithm, based on dual decomposition and coordinate methods, to solve partitioned optimization problems. We show that, by exploiting the problem structure, the solution can be partitioned among the nodes, so that each node just stores a local copy of a portion of the decision variable (rather than a copy of the entire decision vector) and solves a small-scale local problem.**

I. Introduction

Distributed optimization has received a widespread attention in the last years due to its key role in multi-agent systems (also known as large-scale systems, sensor networks or peer-to-peer networks). Several solutions have been proposed, but many challenges are still open. In this paper we focus on a main common limitation of the current approaches. That is, in all the currently available algorithms the nodes in the network reach consensus on the entire solution vector. This redundancy of information may be not necessary or even realizable in some problem set-ups. Thus, we exploit a new distributed optimization set-up in which the nodes compute only a portion of the solution and the whole minimizer may be obtained by stacking together the local portions.

We divide the relevant literature for our paper in two parts. That is, we review works on distributed optimization more closely related to the techniques proposed in this paper, and the centralized and parallel literature on big-data optimization.

Early references on distributed optimization are [2], [3]. Convex optimization problems are solved by using a primal distributed subgradient method combined with a consensus scheme. Dual decomposition methods have been proposed in early references in order to develop distributed algorithms in a pure peer-to-peer set-up. In [4] a tutorial on network optimization via dual decomposition can be found. In [5] a synchronous distributed algorithm based on a dual decomposition approach is proposed for a convex optimization problem with a common constraint for all the agents. In [6] equality and inequality constraints are handled in a distributed set-up based on duality. In [7] a distributed algorithm based on an averaging scheme on the dual variables is proposed, to solve convex optimization problems over fixed undirected networks. A slightly different set-up is considered in [8], where a dual decomposition method over time-varying graphs is proposed. In order to induce robustness in the computation and improve convergence in the case of non-strictly convex functions, Alternating Direction Methods of Multipliers (ADMM) have been proposed in the network context [9]. A distributed consensus optimization algorithm based on an inexact ADMM is proposed in [10]. In [11] an asynchronous ADMM-based distributed method is proposed for a separable, constrained optimization problem. A different class of algorithms, working under a general asynchronous and directed communication, is based on the exchange of cutting planes among the network nodes [12] and can be applied also in its dual form to separable convex programs.

A common drawback of the above algorithms is that they are well suited for a set-up in which either the dimension of the decision variable or the number of constraints is constant with respect to the number of nodes in the network. In case both the two features depend on the number of nodes each local computing agent needs to handle a problem whose dimension is not scalable with respect to the network dimension. To cope with big-data optimization problems, deterministic and randomized coordinate methods for both unconstrained and constrained optimization have been proposed, see e.g., [13]–[15]. More general set-ups such as composite and/or separable optimization in a parallel scenario have been addressed for convex problems in [16], [17], whereas nonconvex problems are considered in [18]–[20]. In [21] an edge-based distributed algorithm is proposed to solve linearly coupled optimization problems via a coordinate descent method. A distributed coordinate primal-dual asynchronous algorithm is proposed in [22] to deal with large-scale problems. A dual approach has been combined with a coordinate proximal gradient in [23] to propose an asynchronous distributed algorithm for composite convex optimization.

In this paper we investigate a class of problems of interest in several multi-agent applications in which the decision variable grows as the number of nodes in the network, but

A preliminary short version of this paper has appeared as [1], where only a synchronous dual scheme for the partitioned set-up was considered.

[1]Ivano Notarnicola and Giuseppe Notarstefano are with the Department of Engineering, Università del Salento, Lecce, Italy, name.lastname@unisalento.it. This result 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).

[2]Ruggero Carli is with the Department of Information Engineering, University of Padova, Italy, carlirug@dei.unipd.it.

the cost function and the constraints have a special partitioned structure. We show that such structure is not derived just as a pure academic exercise, but vice-versa appears in several important application scenarios. In particular, we present two of them that have been widely investigated in the literature, namely distributed quadratic estimation and network utility maximization (and its related resource allocation version).

The main contribution of this paper is as follows. For this problem set-up we provide two distributed optimization algorithms, based on dual decomposition, with two main appealing features. First, the algorithms are scalable, in the sense that each node only processes a portion of the decision variable vector. As a result, the information stored and the computation performed by each node does not depend on the network size as long as the node degree is bounded. Second, the asynchronous algorithm works under a communication protocol in which a node wakes-up when triggered by its local timer or by its neighbors, so that no global clock is needed. The distributed algorithms are derived by first writing a suitable equivalent formulation of the original primal optimization problem (which exploits the partitioned structure). Then its dual problem is derived and solved with suitable algorithms. A scaled gradient applied to the dual problem turns out to be a partitioned version of the distributed dual decomposition (synchronous) algorithm. A randomized ascent method applied to the dual problem allows us to write an asynchronous distributed algorithm that converges in objective value with high probability.

As opposed to [4]–[7], even though we also consider a dual decomposition approach, we tailored the methodology for the partitioned set-up, thus explicitly taking into account the partitioned structure of the cost and the constraints. This results in algorithmic formulations that reduce memory and communication burden. The partitioned set-up considered in this paper has been introduced in [24], where a distributed ADMM algorithm is proposed. In [25] a nonconvex maximum likelihood localization partitioned problem is solved via a similar distributed ADMM scheme. In [26] a convex composite optimization problem is considered where the cost has a partitioned part and a fully separable remainder. A parallel coordinate algorithm is proposed with its convergence analysis. In [27] a robust block-Jacobi algorithm for a partitioned quadratic programming under lossy communications is proposed. A related formulation of the partitioned problem is the one considered in [28], where the D-ADMM distributed algorithm proposed in [29] has been applied. Differently from the above references, in this paper we propose a dual decomposition algorithm for optimization problems in which also the constraints exhibit a partitioned structure. Moreover, we develop an asynchronous distributed algorithm, inspired to [23], by combining dual decomposition with coordinate methods.

The paper is organized as follows. In Section II we present the partitioned optimization framework and describe two motivating applications. In Section III we develop a partitioned distributed dual decomposition approach, then we propose and analyze our synchronous and asynchronous distributed algorithms. Finally, in Section IV we run simulations to corroborate the theoretical results.

II. Problem Set-up and Motivating Scenarios

A. Problem Set-up

We consider a network of agents aiming at solving a structured optimization problem in a distributed way. The nodes, $\{1,\dots,n\}$, interact according to a fixed *connected, undirected* graph $\mathcal{G}=(\{1,\dots,n\},\mathcal{E})$. We denote $\mathcal{N}_i$ the set of neighbors of node i in $\mathcal{G}$, that is $\mathcal{N}_i=\{j\in\{1,\dots,n\}\mid (i,j)\in\mathcal{E}\}$. As we will see in the following, the graph $\mathcal{G}$ is related to the structure of the optimization problem.

As for the communication, we will consider a synchronous communication protocol in which nodes communicate over the fixed graph according to a common clock, and an asynchronous protocol in which, although the neighboring agents are determined by the fixed graph $\mathcal{G}$, communication happens asynchronously. We will formally define this last communication protocol in the next sections.

We start by reviewing a common set-up in distributed optimization. That is, we consider the minimization of a separable cost function subject to local constraints,

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

where $f_i:\mathbb{R}^N\to\mathbb{R}$ and $X_i\subseteq\mathbb{R}^N$ for all $i\in\{1,\dots,n\}$. In our set-up the local objective function f_i and the local constraint set X_i are known only by agent i.

In this paper we want to consider problems as in (1) with a specific feature, that is a *partitioned structure*, that we next describe. Let the vector x be partitioned as

$$x=[x_1^\top,\dots,x_n^\top]^\top$$

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

$$\begin{aligned}\min_{x}\ &\sum_{i=1}^n f_i(x_i,\{x_j\}_{j\in\mathcal{N}_i})\\ \text{subj. to } (x_i,\{x_j\}_{j\in\mathcal{N}_i})\in X_i,\qquad & i\in\{1,\dots,n\},\end{aligned}\tag{2}$$

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

We stress that the constraint sets X_i can involve all (neighboring) variables $(x_i,\{x_j\}_{j\in\mathcal{N}_i})$ of agent i and not just x_i. This apparently minor feature in fact adds much more generality to the problem and introduces important significant challenges.

The following assumptions will be used in the paper.

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

Assumption II.2. *The constraint sets $X_i \subseteq \mathbb{R}^{\sum_{j\in\mathcal{N}_i\cup\{i\}} m_j}$, $i \in \{1,\dots,n\}$ are nonempty convex and compact.* □

Assumption II.3 (Constraint qualification)**.** *The intersection of the relative interior of the sets X_i, $i \in \{1,\dots,n\}$, is nonempty.* □

Under Assumptions II.1 and II.2 problem (2) is feasible and admits a unique optimal solution $f^\star$ attained at some $x^\star \in \mathbb{R}^N$. Assumption II.3 is a standard requirement to guarantee that a dual approach will enjoy the strong duality property.

B. Motivating Examples

Next we provide two application scenarios in which the partitioned structure of the optimization problem arises naturally.

1) Distributed estimation in power networks: To describe this example we follow the treatment in [30].

For a power network, the state at a certain instant of time consists of the voltage angles and magnitudes at all the system buses. The (static) state estimation problem refers to the procedure of estimating the state of a power network given a set of measurements of the network variables, such as, for instance, voltages, currents, and power flows along the transmission lines. To be more precise, let $x \in \mathbb{R}^N$ and $z \in \mathbb{R}^P$ be, respectively, the state and measurements vector. Then, the vectors x and z are related by the relation

$$z = h(x) + \eta, \tag{3}$$

where $h(\cdot)$ is a nonlinear measurement function, and where η is the noise measurement, which is traditionally assumed to be a zero mean random vector satisfying $\mathbb{E}[\eta\eta^\top] = \Sigma \succ 0$. An optimal estimate of the network state coincides with the most likely vector $x^\star$ that solves equation (3). This static state estimation problem can be simplified by adopting the approximated estimation model presented in [31], which follows from the linearization around the origin of equation (3). Specifically,

$$z = Hx + v,$$

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

$$\operatorname*{argmin}_x\ (z - Hx)^\top \Sigma^{-1}(z - Hx). \tag{4}$$

Assume $\ker(H) = 0$, then the optimal solution to the above problem is given by

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

For simplicity let us assume that $\Sigma = I$. For a large power network, the centralized computation of x_{wls} might be too onerous. A possible solution to address this complexity problem is to distribute the computation of x_{wls} among geographically deployed control centers (monitors), say n in a way that each monitor is responsible only for a subpart of the whole network. Precisely let the matrices H and Σ and the vector z be partitioned as $[H_{ij}]_{i,j=1}^n$, $x = \left[x_1^\top,\dots,x_n^\top\right]^\top$ and $z = \left[z_1^\top,\dots,z_n^\top\right]^\top$, where $H_{ij} \in \mathbb{R}^{p_i\times m_j}$, $z_i \in \mathbb{R}^{p_i}$, $x_i \in \mathbb{R}^{m_i}$ and $\sum_{i=1}^n m_i = N$, $\sum_{i=1}^n p_i = P$. Observe that, because of the interconnection structure of a power network, the measurement matrix H is usually sparse, i.e., many $H_{ij} = 0$. Assume monitor i knows z_i and H_{ij}, $j \in \{1,\dots,n\}$ and it is interested only in estimating the sub-state x_i. Moreover let $\mathcal{N}_i = \{j \in \{1,\dots,n\} \mid H_{ij} \neq 0\}$. Observe that in general if $H_{ij} \neq 0$ then also $H_{ji} \neq 0$. Then by defining

$$f_i\big(x_i, \{x_j\}_{j\in\mathcal{N}_i}\big) = \Big(z_i - \sum_{j\in\mathcal{N}_i} H_{ij}x_j\Big)^\top \Big(z_i - \sum_{j\in\mathcal{N}_i} H_{ij}x_j\Big),$$

problem (4) can be equivalently rewritten as

$$\operatorname*{argmin}_x \sum_{i=1}^n f_i\big(x_i, \{x_j\}_{j\in\mathcal{N}_i}\big)$$

which is of the form (2).

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

2) Network utility maximization and resource allocation: We consider the flow optimization problem, or *Network Utility Maximization (NUM)* problem introduced in [32] and studied in [33] in a distributed context. A flow network (which is different from a communication network) consists of a set L of unidirectional links with capacities c_ℓ, $\ell \in L$. The network is shared by a set of n sources. Each source has a strongly concave utility function $U_i(x_i)$ The goal is to calculate source rates that maximize the sum of the utilities $\sum_{i=1}^n U_i(x_i)$ over x_i subject to capacity constraints. Formally, using a notation consistent with [33], let $L(i) \subseteq L$ be the set of links used by source i and $N(\ell) = \{i \in \{1,\dots,n\} \mid \ell \in L(i)\}$ be the set of sources that use link ℓ. Note that $\ell \in L(i)$ if and only if $i \in N(\ell)$. Also, let $I_i = [\kappa_i, K_i]$, with $0 \le \kappa_i < K_i$, be the interval of transmission rates allowed to node i. The network flow optimization problem is given by

$$\begin{aligned} \max_{x_1,\dots,x_n}\ & \sum_{i=1}^n U_i(x_i) \\ \text{subj. to}\ & x_i \in I_i, && i \in \{1,\dots,n\}, \\ & \sum_{j\in N(\ell)} x_j \le c_\ell, && \ell \in \{1,\dots,|L|\}. \end{aligned} \tag{5}$$

Notice that problem (5) is well posed and has compact domain.

In Figure 1 (left) we graphically represent an example of 5 sources (filled circles) that use (dotted arrows) 3 links (gray stripes). In [33] a distributed optimization algorithm is proposed in which both the sources and the links are computation units. Here we consider a set-up in which only the sources are computation units. In particular, the sources have the computation and communication capabilities introduced in the previous subsection. We assume that sources using

the same links can communicate and both know the capacity constraint on those links. Formally, we introduce a graph $\mathcal{G}$ having an edge (i,j) connecting source i to j if and only if there exists $\ell \in L$ such that $\ell \in L(i) \cap L(j)$. In Figure 1 (right) we show the induced communication graph (solid lines) for the considered example.

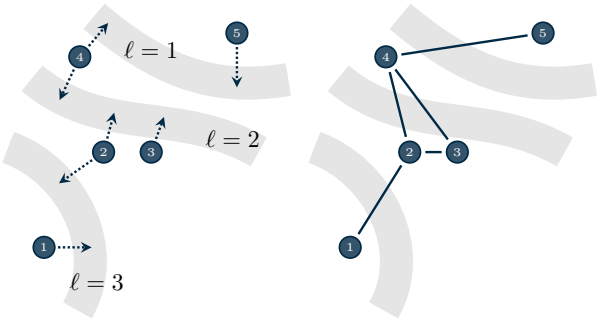

Fig. 1. Network Utility Maximization problem with 5 sources (filled circles) using 3 links (gray stripes).

Thus, optimization problem (5) can be rewritten as

$$\begin{aligned} \max_{x_1,\dots,x_n} \ & \sum_{i=1}^{n} U_i(x_i) \\ \text{subj. to } \ & (x_i, \{x_j\}_{j\in\mathcal{N}_i}) \in \prod_{j\in\mathcal{N}_i\cup\{i\}} I_j,\ i \in \{1,\dots,n\}, \\ & \sum_{j\in\mathcal{N}_i\cup\{i\}} a_{j,\ell}\, x_j \le c_\ell,\ \ \ell \in L(i),\ i \in \{1,\dots,n\}, \end{aligned} \tag{6}$$

where $\mathcal{N}_i$ is the neighbor set of i in $\mathcal{G}$, $a_{j,\ell}$ is 1 if agent j can use link ℓ and 0 otherwise. Notice that in problem (6) if sources i and j share a link ℓ, then they both have the capacity constraint of link ℓ. Moreover, in order to have compactness of the local constraint set X_i, transmission rate constraints of neighboring nodes are also taken into account. Finally, in order to fulfill the strong convexity assumption on the local costs, two strategies can be used. First, one can assume that each agent knows the utility functions of neighboring agents, so that it sets $f_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) = \sum_{j\in\mathcal{N}_i\cup\{i\}} U_j(x_j)$. Alternatively, one can consider an additional separable (small) regularization term in the NUM problem formulation in (5), e.g., $\epsilon \sum_{i=1}^{n} x_i^2$ with $\epsilon > 0$. In this case each agent sets its local cost function to $f_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) = U_i(x_i) + \sum_{j\in\mathcal{N}_i} \epsilon_{ij} x_j^2$, where $\epsilon_{ij} > 0$ are suitable fractions of ϵ. Except for the maximization versus minimization, this problem is partitioned, that is it has the same structure as (2).

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

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

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

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

It is worth noting that dual decomposition is often used in network utility maximization and resource allocation problems. See for example [34] for a tutorial on dual decomposition methods in network utility maximization. Usually, in this context, the capacity constraints are dualized to obtain a master-subproblem or a distributed algorithm. However, in these early references, as e.g., in [33], the dual decomposition gives rise to algorithms that are not suited for a pure peer-to-peer network as the one we consider. In our partitioned approach the dual decomposition is used to enforce the coherence constraints, whereas the capacity constraints are taken into account in the primal local minimization. These aspects will be more clear in the next section in which we derive the partitioned dual decomposition approach.

III. Partitioned Dual Decomposition for Distributed Optimization

In order to introduce our distributed algorithms, we derive a partitioned dual decomposition scheme by introducing suitable copies of the decision variables.

As a preliminary step, we briefly recall the standard dual decomposition approach for distributed optimization. In order to solve problem (1) in a distributed way, a common approach consists of writing it in the equivalent form

$$\begin{aligned} \min_{x^{(1)},\dots,x^{(n)}} \ & \sum_{i=1}^{n} f_i(x^{(i)}) \\ \text{subj. to } \ & x^{(i)} \in X_i, \qquad i \in \{1,\dots,n\}, \\ & x^{(i)} = x^{(j)}, \qquad (i,j) \in \mathcal{E}, \end{aligned} \tag{7}$$

where each $x^{(i)}$ can be seen as a copy of x subject to the additional constraint that all the copies must be equal. Clearly, the connected nature of the network ensures equality between all $x^{(i)}$ and, in turn, the equivalence between (7) and (1).

When considering a partitioned problem as in (2), because of the structure of f_i and X_i, $i \in \{1,\dots,n\}$, the formulation (1) is considerably redundant. The idea is to exploit the partitioned structure to modify (7) in order to limit the range of equivalences among the auxiliary variables, and, in turn, their diffusion over the network.

A. Partitioned Dual Decomposition Set-up

Once we create copies of the vector $x \in \mathbb{R}^N$, we enforce each state $x_i \in \mathbb{R}^{m_i}$ to be identical only for the neighboring

nodes $j \in \mathcal{N}_i \cup \{i\}$ which use this information. Formally, we reformulate problem (2) as

$$\begin{aligned} \min \ & \sum_{i=1}^{n} f_i\big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ \text{subj. to } & \big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \in X_i \qquad i \in \{1,\dots,n\}, \\ & x_i^{(i)} = x_i^{(j)}, \qquad\quad j \in \mathcal{N}_i,\ i \in \{1,\dots,n\}, \\ & x_j^{(i)} = x_j^{(j)}, \qquad\quad j \in \mathcal{N}_i,\ i \in \{1,\dots,n\}, \end{aligned} \tag{8}$$

where $x_i^{(j)}$ denotes the copy of state x_i stored in memory of node j. Notice that connected nature of the graph $\mathcal{G}$ ensures equivalence between (2) and (8).

As an example, in Figure 2 we visualize the partitioned set-up for a path graph of $n = 4$ nodes. Along i-th column, we show the coupling due to the local cost f_i and the local constraint X_i, which involves only the states handled by node i, i.e., $x_i^{(i)}$ and $x_j^{(i)}$ with $j \in \mathcal{N}_i$. Along the i-th row, we show the coupling due to copies $x_i^{(j)}$, $j \in \mathcal{N}_i$, of the variable x_i.

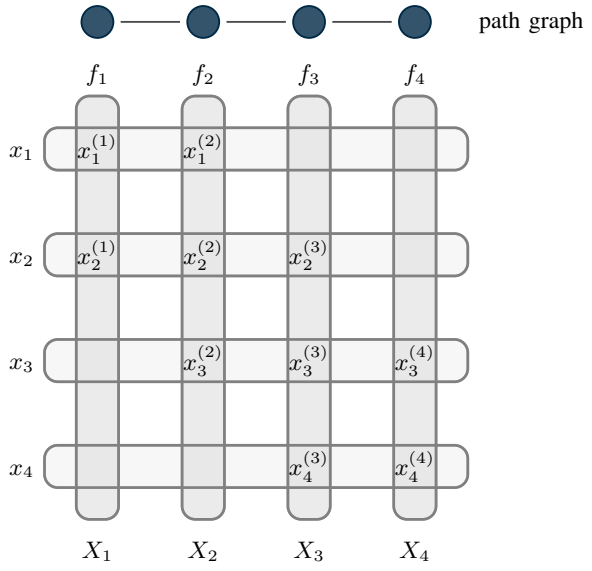

Fig. 2. Partitioned optimization problem over a path graph of $n = 4$ nodes.

Before proceeding with presentation of the algorithms, we discuss two key features in the structure of the above problem. First, it is worth noting that the problem formulations (7) and (8), although equivalent, are different. In fact, (8) will lead to our partitioned algorithm. Second, we point out that a constraint $x_i^{(i)} = x_i^{(j)}$ for a pair of agents i and j appears two times. This redundant formulation is not accidental, but plays an important role in exploiting the partitioned structure of the proposed algorithm.

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

$$y^{(i)} := \big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \tag{9}$$

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

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

To tackle problem (8) in a distributed way, we start by deriving its dual problem. The *partial* Lagrangian for problem (8) is given by

$$\begin{aligned} \mathcal{L}(\mathbf{x},\Lambda) = \sum_{i=1}^{n} \Bigg(& f_i\big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ & + \sum_{j\in\mathcal{N}_i} \Big(\lambda_i^{(i,j)\top}(x_i^{(i)} - x_i^{(j)}) + \lambda_j^{(i,j)\top}(x_j^{(i)} - x_j^{(j)})\Big)\Bigg), \end{aligned} \tag{10}$$

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

$$\Lambda = \big[\Lambda_1^\top, \dots, \Lambda_n^\top\big]^\top,$$

with block $\Lambda_i := [\{\lambda_i^{(i,j)}\}_{j\in\mathcal{N}_i}, \{\lambda_j^{(i,j)}\}_{j\in\mathcal{N}_i}]$, $i \in \{1,\dots,n\}$.

By exploiting the undirected nature and the connectivity of graph $\mathcal{G}$, the Lagrangian (10) can be rewritten as

$$\begin{aligned} \mathcal{L}(\mathbf{x},\Lambda) = \sum_{i=1}^{n} \Bigg(& f_i\big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ & + x_i^{(i)\top} \sum_{j\in\mathcal{N}_i} (\lambda_i^{(i,j)} - \lambda_i^{(j,i)}) + \sum_{j\in\mathcal{N}_i} x_j^{(i)\top} \big(\lambda_j^{(i,j)} - \lambda_j^{(j,i)}\big)\Bigg) \end{aligned} \tag{11}$$

which is separable with respect to $y^{(i)}$, $i \in \{1,\dots,n\}$.

Remark III.1. *It is worth noting that we have not dualized the local constraints $(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}) \in X_i$ (thus the notion of partial Lagrangian) since each of them will be handled by the agents in their local optimization problem.* □

The dual function of (8) is obtained by minimizing the Lagrangian with respect to the primal variables, which gives

$$\begin{aligned} q(\Lambda) &= \min_{\mathbf{x}\in X_1\times\cdots\times X_n} \mathcal{L}(\mathbf{x},\Lambda) \\ &= \sum_{i=1}^{n} q_i\Big(\big\{\lambda_i^{(i,j)}, \lambda_i^{(j,i)}, \lambda_j^{(i,j)}, \lambda_j^{(j,i)}\big\}_{j\in\mathcal{N}_i}\Big) \end{aligned}$$

with

$$\begin{aligned} & q_i\Big(\big\{\lambda_i^{(i,j)}, \lambda_i^{(j,i)}, \lambda_j^{(i,j)}, \lambda_j^{(j,i)}\big\}_{j\in\mathcal{N}_i}\Big) = \\ & \min_{\left(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\right)\in X_i} \Big(f_i\big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big) \\ & \quad + x_i^{(i)\top} \sum_{j\in\mathcal{N}_i} (\lambda_i^{(i,j)} - \lambda_i^{(j,i)}) + \sum_{j\in\mathcal{N}_i} x_j^{(i)\top} (\lambda_j^{(i,j)} - \lambda_j^{(j,i)})\Big). \end{aligned} \tag{12}$$

Notice that, since each X_i is compact and nonempty, the minimum in (12) is (uniquely) attained, so that q_i is always finite. Thus, the dual problem of (8) is the following unconstrained optimization problem

$$\max_{\Lambda} \ \sum_{i=1}^{n} q_i\Big(\big\{\lambda_i^{(i,j)}, \lambda_i^{(j,i)}, \lambda_j^{(i,j)}, \lambda_j^{(j,i)}\big\}_{j\in\mathcal{N}_i}\Big). \tag{13}$$

Remark III.2. *Let $\varphi : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, its conjugate function $\varphi^* : \mathbb{R}^d \to \mathbb{R}$ is defined as*

$$\varphi^*(z) := \sup_x \big(z^\top x - \varphi(x)\big).$$

Then,

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

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

Remark III.3. *It is worth noting that each q_i does not depend on the entire set of dual variables Λ, but it exhibits a sparse structure, i.e., it is a function of the dual variables of the neighbors $\mathcal{N}_i$ only.* □

B. (Synchronous) Partitioned Dual Decomposition (PDD) distributed algorithm

With the dual problem in hand, a gradient algorithm on the dual problem, [35, Chapter 6], can be applied. This results into a minimization on the primal variables and a linear update on the dual variables. As we will show in the analysis, this gives rise to the PDD distributed algorithm, which is formally stated, from the perspective of node i, in the following table.

We point out that each node $i\in\{1,\dots,n\}$ stores and updates the primal variables $x_i^{(i)}$ and $x_j^{(i)}$, $j\in\mathcal{N}_i$, and the dual variables $\lambda_i^{(i,j)}$ and $\lambda_j^{(i,j)}$, $j\in\mathcal{N}_i$.

Distributed Algorithm 1 PDD

Processor states: $(x_i^{(i)},\{x_j^{(i)}\}_{j\in\mathcal{N}_i})$ and $\{\lambda_i^{(i,j)},\lambda_j^{(i,j)}\}_{j\in\mathcal{N}_i}$

Evolution:

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

Compute and broadcast primal variables

$$\big(x_i^{(i)}(t+1),\{x_j^{(i)}(t+1)\}_{j\in\mathcal{N}_i}\big) = \underset{(x_i,\{x_j\}_{j\in\mathcal{N}_i})\in X_i}{\operatorname{argmin}} \Big(f_i\big(x_i,\{x_j\}_{j\in\mathcal{N}_i}\big) + x_i^\top\sum_{j\in\mathcal{N}_i}\Big(\lambda_i^{(i,j)}(t)-\lambda_i^{(j,i)}(t)\Big) + \sum_{j\in\mathcal{N}_i}x_j^\top\Big(\lambda_j^{(i,j)}(t)-\lambda_j^{(j,i)}(t)\Big)\Big). \tag{14}$$

Update and broadcast dual variables via

$$\begin{aligned}\lambda_i^{(i,j)}(t+1) &= \lambda_i^{(i,j)}(t)+\alpha_i\Big(x_i^{(i)}(t+1)-x_i^{(j)}(t+1)\Big)\\ \lambda_j^{(i,j)}(t+1) &= \lambda_j^{(i,j)}(t)+\alpha_i\Big(x_j^{(i)}(t+1)-x_j^{(j)}(t+1)\Big)\end{aligned} \tag{15}$$

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

Before studying the convergence properties of the proposed algorithm, let us comment on its scalability and how it compares with standard dual gradients algorithms. First, observe that each node has to keep in memory the set of variables $x_i^{(i)}$, $\{x_j^{(i)}\}_{j\in\mathcal{N}_i}$, $\{\lambda_i^{(i,j)},\lambda_j^{(i,j)}\}_{j\in\mathcal{N}_i}$, namely a number of variables equal to $1+3|\mathcal{N}_i|$. Second, the step-sizes α_i, $i\in\{1,\dots,n\}$ are constant, local and can be initialized via local computations. More details are given in Theorem III.5.

Remark III.4. *Notice that, differently from existing dual decomposition schemes, our algorithms do not enforce any symmetry in the dual variables, i.e., in general $\lambda_i^{(i,j)}(t)\neq-\lambda_i^{(j,i)}(t)$. The symmetry, although not necessary, can be imposed if the agents select a common step-size $\alpha_i=\alpha$, for all $i\in\{1,\dots,n\}$, and properly initialize their dual variables. As a consequence, the algorithm can be simplified to have only one communication round to perform both the local minimization and the ascent.* □

The convergence properties of PDD (Distributed Algorithm 1) are established in the following theorem.

Theorem III.5. *Let Assumptions II.1, II.2 and II.3 hold true and assume the step-sizes α_i, $i\in\{1,\dots,n\}$, to be constant and such that $0<\alpha_i\leq\frac{1}{nL_i}$, with*

$$L_i=\sqrt{2\sum_{j\in\mathcal{N}_i}\Big(\frac{1}{\sigma_i}+\frac{1}{\sigma_j}\Big)^2},\quad\forall\, i\in\{1,\dots,n\}. \tag{16}$$

Then, the sequence $\{\Lambda_1(t),\dots,\Lambda_n(t)\}$ generated by PDD (Distributed Algorithm 1) converges in objective value to the optimal cost $f^\star$ of problem (2)*. Moreover, let $x^\star=(x_1^{\star\top},\dots,x_n^{\star\top})^\top$ be the unique optimal solution of* (2)*, then each primal sequence $x_i^{(i)}(t)$ generated by PDD is such that*

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

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

Proof. We structure the proof of the first statement in three parts in which we show that: (i) the dual gradient has a block structure and smoothness, (ii) the distributed algorithm implements a diagonally-scaled gradient method, and (iii) strong duality holds. First, consider the dual problem (13) and a block partitioning of dual variables $\Lambda=[\Lambda_1,\dots,\Lambda_n]$, with

$$\Lambda_i:=\Big(\{\lambda_i^{(i,j)}\}_{j\in\mathcal{N}_i},\{\lambda_j^{(i,j)}j\in\mathcal{N}_i\}\Big) \tag{17}$$

representing the local variables of node i, for all $i\in\{1,\dots,n\}$. Under Assumption II.1, the dual function $q(\Lambda)$ is guaranteed to have block-coordinate Lipschitz continuous gradient $\nabla q(\Lambda)$ with block constants L_i, $i\in\{1,\dots,n\}$, given in (16). In fact, we can explicitly compute the components of $\nabla q(\Lambda)$ associated to each block Λ_i, denoted hereafter as $\nabla_{\Lambda_i}q(\Lambda)$, by using the chain rule of derivation and the conjugate function notation. We have that

$$\begin{aligned}\frac{\partial q(\Lambda)}{\partial\lambda_i^{(i,j)}} &= (\nabla f_i^*)_i-(\nabla f_j^*)_i,\ j\in\mathcal{N}_i\\ \frac{\partial q(\Lambda)}{\partial\lambda_j^{(i,j)}} &= (\nabla f_i^*)_j-(\nabla f_j^*)_j,\ j\in\mathcal{N}_i,\end{aligned} \tag{18}$$

where $(\nabla f_i^*)_i$ denotes the i-th component of ∇f_i^* and we omit the argument of ∇f_i^* to take light the notation.

Since for all $i\in\{1,\dots,n\}$, each f_i is a strongly convex function, then the gradient of its conjugate function ∇f_i^* is Lipschitz continuous with constant $1/\sigma_i$, [36, Chapter X, Theorem 4.2.2]. By considering the Euclidean 2-norm, in light

of (18) and by simple algebraic manipulation, we can conclude that also $\nabla_{\Lambda_i} q(\Lambda)$ is Lipschitz continuous with constant

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

which matches (16).

Second, we show that our PDD distributed algorithm implements a scaled gradient ascent method to solve problem (13).

Consider a diagonal positive definite matrix defined as $W := \operatorname{diag}(\alpha_1,\dots,\alpha_n) \preceq \operatorname{diag}(\frac{1}{nL_1},\dots,\frac{1}{nL_n})$. Formally, the scaled gradient ascent method can be written as

$$\Lambda(t+1) = \Lambda(t) + W\nabla q(\Lambda(t)), \tag{19}$$

where t denotes the iteration counter. Since each entry of the scaling matrix satisfies $0 < \alpha_i \leq \frac{1}{nL_i}$ for all $i \in \{1,\dots,n\}$, then the following condition holds [17, Theorem 8]

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

for every perturbation δ. Thus, using the same line of proof of the gradient algorithm [35, Chapter 2], we can conclude that the sequence $\{\Lambda(t)\}$ generated by iteration (19) converges in objective value to the optimal cost $q^\star$ of (13).

Since W is diagonal, then (19) splits in a component-wise fashion giving

$$\Lambda_i(t+1) = \Lambda_i(t) + W_{ii}\nabla_{\Lambda_i} q(\Lambda(t)),\ i\in\{1,\dots,n\}, \tag{20}$$

where W_{ii} denotes the (i,i)-th entry of W.

By using the following property of conjugate functions

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

we have that the primal minimization (14) computes ∇f_i^* evaluated at the point $\big(\sum_{j\in\mathcal{N}_i}(\lambda_i^{(i,j)}(t) - \lambda_i^{(j,i)}(t)), \{(\lambda_j^{(i,j)}(t) - \lambda_j^{(j,i)}(t))\}_{j\in\mathcal{N}_i}\big)$. Then

$$\begin{aligned}\frac{\partial q(\Lambda(t))}{\partial\lambda_i^{(i,j)}} &= x_i^{(i)}(t+1) - x_i^{(j)}(t+1),\ j\in\mathcal{N}_i\\ \frac{\partial q(\Lambda(t))}{\partial\lambda_j^{(i,j)}} &= x_j^{(i)}(t+1) - x_j^{(j)}(t+1),\ j\in\mathcal{N}_i,\end{aligned} \tag{21}$$

so that update (15) is the scaled gradient ascent (20).

Third and final, by Assumption II.3 (Slater's condition), strong duality between problems (8) and (13) holds. Moreover, since problems (8) and (2) are equivalent, then they both have optimal cost $q^\star = f^\star$. Thus, the sequence $\{\Lambda(t)\}$ generated by PDD converges in objective value to the optimal cost $f^\star$ of (2).

For the second part of the statement, we first notice that in light of Assumptions II.1 and II.2, problem (2) has a unique optimal solution $x^\star = (x_1^{\star\top},\dots,x_n^{\star\top})^\top$. Further, since problem (8) is equivalent to problem (2), then $x^\star$ is the unique optimal solution also for problem (8). Finally, the first order optimality condition for the (unconstrained) dual problem (13) is $\nabla q(\Lambda^\star) = 0$, where $\Lambda^\star$ is a limit point of the sequence $\{\Lambda(t)\}$ (which exists by the Lipschitz continuity of $\nabla q(\Lambda)$). This allows us to conclude, by equation (21), that the limit point of the primal sequences $\{x_i^{(i)}(t), \{x_j^{(i)}\}_{j\in\mathcal{N}_i}(t)\}$ satisfy the primal coherence constraints. Thus, in the limit the copies $x_i^{(i)}, \{x_i^{(j)}\}_{j\in\mathcal{N}_i}$ of the variable x_i are equal to the (unique) optimal $x_i^\star$. Iterating on $i\in\{1,\dots,n\}$ the proof follows. □

Remark III.6. *Alternative expressions for L_i in* (16) *can be used. Larger upper bounds on the step-sizes α_i can be established by exploiting tailored descent conditions. See, e.g., works [14], [17], [26].* □

C. Asynchronous Partitioned Dual Decomposition (AsynPDD) distributed algorithm

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

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

Assumption III.7 (Exponential i.i.d. local timers)**.** *The waiting times between consecutive triggering events are i.i.d. random variables with same exponential distribution.* □

Informally, the asynchronous distributed optimization algorithm is as follows. When a node i is in idle, it continuously receives messages from awake neighbors. If the local timer τ_i triggers or new dual variables $\lambda_i^{(j,i)}$, $\lambda_j^{(j,i)}$ are received, it wakes up. When node i wakes up, it updates and broadcasts its primal variable $y^{(i)} = \big(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}\big)$, computed through a local constrained minimization. Moreover, if the transition was due to the local timer triggering, then it also updates and broadcasts its local dual variables $\lambda_i^{(i,j)}$ and $\lambda_j^{(i,j)}$, $j\in\mathcal{N}_i$. Since there is no global iteration counter, we highlight the difference between updated and not updated values during the "awake" phase, by means of a "$^+$" superscript symbol, e.g., we denote the updated primal variable as $x_i^{(i)+}$.

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

The AsynPDD distributed algorithm is formally described in the following table.

Distributed Algorithm 2 AsynPDD

Processor states: $(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i})$ and $\{\lambda_i^{(i,j)}, \lambda_j^{(i,j)}\}_{j\in\mathcal{N}_i}$

Set $\tau_i = 0$ and get a realization T_i

Evolution:

IDLE:

WHILE: $\tau_i < T_i$ DO:

Receive $\lambda_i^{(j,i)}$, $\lambda_j^{(j,i)}$ and/or $x_i^{(j)}$, $x_j^{(j)}$ from $j \in \mathcal{N}_i$.

IF: dual variables are received go to ***AWAKE***.

go to ***AWAKE***.

AWAKE:

Compute and broadcast

$$\begin{aligned}\big(x_i^{(i)+}, \{x_j^{(i)+}\}_{j\in\mathcal{N}_i}\big) = \\ \operatorname*{argmin}_{(x_i,\{x_j\}_{j\in\mathcal{N}_i})\in X_i} f_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) \\ + x_i^\top \sum_{j\in\mathcal{N}_i} \big(\lambda_i^{(i,j)} - \lambda_i^{(j,i)}\big) + \sum_{j\in\mathcal{N}_i} x_j^\top \big(\lambda_j^{(i,j)} - \lambda_j^{(j,i)}\big)\end{aligned}$$

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

$$\begin{aligned}\lambda_i^{(i,j)+} &= \lambda_i^{(i,j)} + \alpha_i\big(x_i^{(i)+} - x_i^{(j)}\big),\ \forall\, j \in \mathcal{N}_i, \\ \lambda_j^{(i,j)+} &= \lambda_j^{(i,j)} + \alpha_i\big(x_j^{(i)+} - x_j^{(j)}\big),\ \forall\, j \in \mathcal{N}_i,\end{aligned} \tag{22}$$

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

Go to ***IDLE***.

It is worth pointing out that being the algorithm asynchronous, for the analysis we need to carefully formalize the concept of algorithm iterations. We will use a nonnegative integer variable t indexing a change in the whole state $\Lambda = [\Lambda_1 \dots \Lambda_n]$ of the distributed algorithm. In particular, each triggering will induce an iteration of the distributed optimization algorithm and will be indexed with t. We want to stress that this (integer) variable t does not need to be known by the agents. That is, this timer is not a common clock and is only introduced for the sake of analysis.

Theorem III.8. *Let Assumptions II.1, II.2 and II.3 hold true. Let the timers τ_i satisfy Assumption III.7 and step-sizes α_i be constant and such that $0 < \alpha_i \le 1/L_i$, with*

$$L_i = \sqrt{2\sum_{j\in\mathcal{N}_i}\Big(\frac{1}{\sigma_i} + \frac{1}{\sigma_j}\Big)^2}, \quad \forall\, i \in \{1,\dots,n\}. \tag{23}$$

Then, the random sequence $\{\Lambda_1(t), \dots, \Lambda_n(t)\}$ generated by the AsynPDD (Distributed Algorithm 2), converges with high probability in objective value to the optimal cost $f^\star$ of problem (2)*, i.e., for any $\varepsilon \in (0, q_0)$, with $q_0 := q(\Lambda(0))$, and target confidence $0 < \rho < 1$, there exists $\bar{t}(\varepsilon,\rho)$ such that for all $t \ge \bar{t}(\varepsilon,\rho)$ it holds*

$$\Pr\Big(\big|q(\Lambda(t)) - f^\star\big| \le \varepsilon\Big) \ge 1 - \rho.$$

Proof. Our proof strategy is based on showing that the iterations of the asynchronous distributed algorithm can be written as the iterations of an ad-hoc version of the coordinate method [16], applied to the dual problem (13).

Let the optimization variable Λ be partitioned in n blocks $[\Lambda_1, \dots, \Lambda_n]$ as in (17), then a coordinate approach consists in an iterative scheme in which only a block-per-iteration, say Λ_{i_t} at time t, of the entire optimization variable Λ is updated at time t, while all the other components Λ_j with $j \in \{1,\dots,n\}\setminus\{i_t\}$ stay unchanged. Formally, a coordinate iteration can be summarized as

$$\begin{aligned}\Lambda_{i_t}(t+1) &= \Lambda_{i_t}(t) + \nabla_{\Lambda_{i_t}} q(\Lambda(t)) \\ \Lambda_j(t+1) &= \Lambda_j(t), \qquad\qquad j \ne i_t.\end{aligned} \tag{24}$$

In the following, we show that the AsynPDD distributed algorithm implements (24) with a uniform random selection of the blocks. Since the timers τ_i trigger independently according to the same exponential distribution, then from an external, global perspective, the induced awaking process of the nodes corresponds to the following: only one node per iteration, say i_t, wakes up randomly, uniformly and independently from previous iterations. Thus, each triggering, which induces an *iteration* of the distributed optimization algorithm and is indexed with t, corresponds to the (uniform) selection of a node in $\{1,\dots,n\}$ that becomes awake.

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

Thanks to the argument just shown and by noticing that $\lambda_{i_t}^{(i_t,j)}$ and $\lambda_j^{(i_t,j)}$, $j \in \mathcal{N}_{i_t}$ are the components of Λ_{i_t}, we have that step (22) corresponds to step (24) with i_t randomly uniformly distributed over $\{1,\dots,n\}$. Finally, recalling that (i) the cost function $q(\Lambda)$ of problem (13) has block-coordinate Lipschitz continuous gradient with respect to the blocks Λ_i (see proof of Theorem III.5) and (ii) the step-sizes α_i are constant and such that $0 < \alpha_i \le 1/L_i$ with L_i in (23), we can invoke [16, Theorem 5] to conclude that the coordinate method (24) (and equivalently the AsynPDD distributed algorithm) converges with high probability to the optimal cost $q^\star$ of problem (13). Recalling that strong duality between problems (2) and (13) holds (see proof of Theorem III.5), then $q^\star = f^\star$, and the proof follows. □

Remark III.9. *As highlighted in Theorem III.8 in order to set the local step-sizes α_i, each node i should know the convexity parameter σ_j of its neighbors but, differently from the synchronous case (cf. condition* (16)*), does not need to know the total number of agents n in the network.* □

To conclude this section, we notice that the asynchronous model employed in our distributed algorithm can be general-

ized. In fact, in the considered model timers are drawn from a common exponential distribution, while independent and completely uncoordinated rules might be more desiderable. This generalization is currently under investigation.

IV. Numerical Simulations

In this section we provide a numerical example showing the effectiveness of the proposed techniques. We test the proposed distributed algorithms on a quadratic program enjoying the partitioned structure described in the previous sections. Specifically, we consider a network of $n = 100$ agents communicating according to an undirected connected Erdős-Rényi random graph $\mathcal{G}$ with parameter $p = 0.2$. Thus, letting $\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right)$ denote a column vector, we consider the following partitioned optimization problem

$$\min_{x} \sum_{i=1}^{n} \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right)^\top Q_i \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) + r_i^\top \left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) \quad (25)$$

$$\text{subj. to } A_i\left(x_i, \{x_j\}_{j\in\mathcal{N}_i}\right) \preceq b_i, \qquad i \in \{1,\dots,n\},$$

where each $x_i \in \mathbb{R}^{m_i}$ and m_i is uniformly drawn from $\{1,2,3,4\}$. This optimization problem has the same partitioned structure discussed in Section III-A. In particular, we have quadratic cost functions $f_i(x_i, \{x_j\}_{j\in\mathcal{N}_i})$ and linear constraints $X_i = \{(x_i, \{x_j\}_{j\in\mathcal{N}_i}) \mid A_i(x_i, \{x_j\}_{j\in\mathcal{N}_i}) \preceq b_i\}$. The matrices Q_i are positive definite with eigenvalues uniformly generated in $[1, 20]$, while the vectors r_i have entries randomly generated in $[0, 100]$. Moreover, each pair A_i, b_i describes a linear constraint having a number of rows uniformly drawn from $\{1, 2\}$. Each A_i has entries normally distributed with zero mean and unitary variance, while b_i are suitably generated to always obtain feasible linear constraints.

For all $i \in \{1,\dots,n\}$, we use constant step-sizes $\alpha_i = L_i$ with L_i computed as in (16) for the synchronous algorithm and as in (23) for the asynchronous case. All the dual variables are initialized to zero.

In Figure 3 we show the convergence rate of the synchronous distributed algorithm by plotting the difference between the dual cost $q(\Lambda(t))$ at each iteration t and the optimal value $q^\star = f^\star$ of problem (25).

In Figure 4 we show the evolution of the difference between the generated primal sequence $\{x_1^{(1)}(t), \dots, x_n^{(n)}(t)\}$ and the (unique) optimal primal solution $x^\star$.

In Figure 5 we show the disagreement on the primal variable x_2 between neighboring nodes $\mathcal{N}_2 \cup \{2\}$. In particular, we plot the norm of $x_2^{(2)}(t) - x_2^{(j)}(t)$, for all $j \in \mathcal{N}_2 \cup \{2\}$.

Finally, in Figure 6 we show the convergence rate for the AsynPDD distributed algorithm. Since we are dealing with an asynchronous algorithm, we normalize the iteration counter t with respect to the total number of agents n. It is worth noting the cost evolution is not monotone as expected for the class of randomized algorithms.

V. Conclusions

In this paper we have proposed a synchronous and an asynchronous distributed optimization algorithms, based on dual

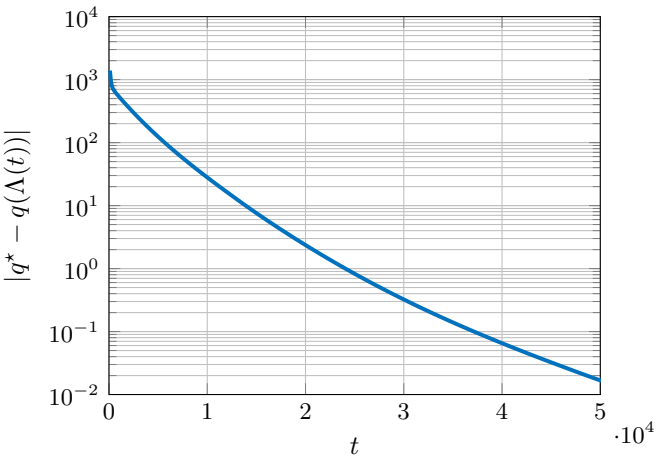

Fig. 3. Evolution of the cost error for the synchronous distributed algorithm.

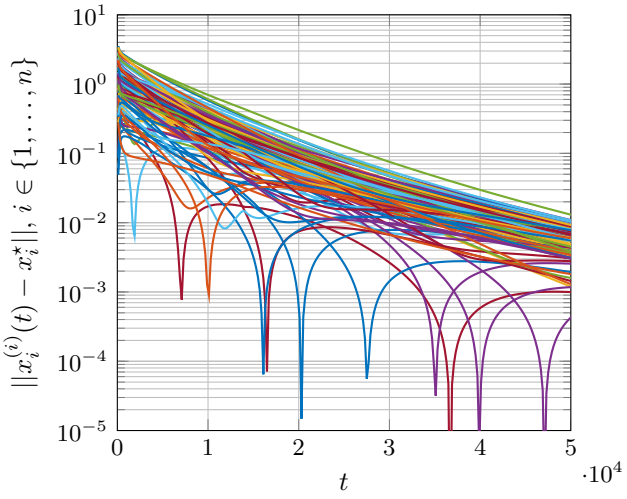

Fig. 4. Evolution of the error on primal variables $x_i^{(i)}$, $i \in \{1,\dots,n\}$, for the synchronous distributed algorithm.

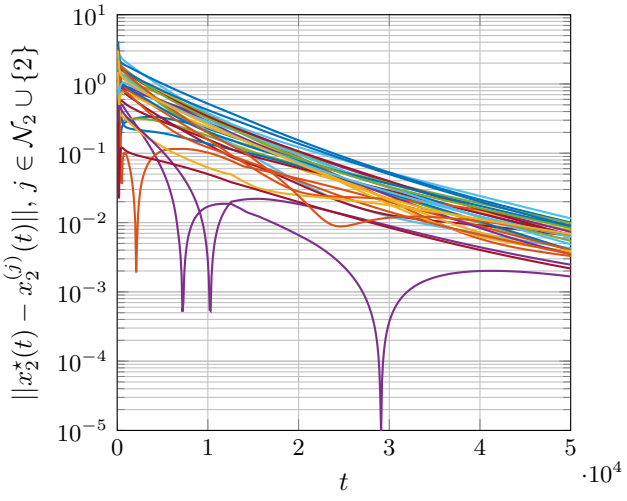

Fig. 5. Evolution of the disagreement on x_2 between agents 2 and its neighbors $j \in \mathcal{N}_2$ for the synchronous distributed algorithm.

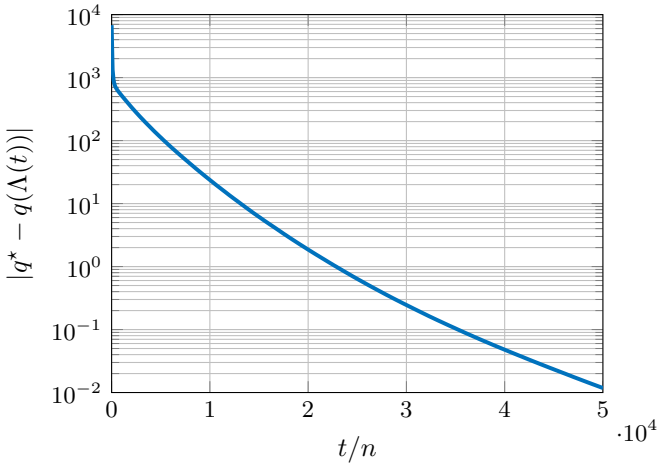

Fig. 6. Evolution of the cost error for the asynchronous distributed algorithm.

decomposition, for a novel partitioned distributed optimization framework. In this framework each node in the network is assigned a local state, objective function and constraint. The objective function and the constraints only depend on the node state and on its neighbors' states. This scenario includes several interesting problems as network utility maximization and resource allocation, static state estimation in power networks, localization in wireless networks, and map building in robotic networks. The proposed algorithms are distributed and scalable and are shown to be convergent under standard assumptions on the cost functions and on the constraints sets.

REFERENCES

[1] R. Carli and G. Notarstefano, "Distributed partition-based optimization via dual decomposition," in *IEEE 52nd Conference on Decision and Control (CDC)*, 2013, pp. 2979–2984.

[2] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[3] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[4] B. Yang and M. Johansson, "Distributed optimization and games: A tutorial overview," *Networked Control Systems*, pp. 109–148, 2011.

[5] H. Terelius, U. Topcu, and R. M. Murray, "Decentralized multi-agent optimization via dual decomposition," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 245–11 251, 2011.

[6] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[7] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[8] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, vol. 84, pp. 149–158, 2017.

[9] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links – part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350 – 364, 2008.

[10] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[11] E. Wei and A. Ozdaglar, "On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *IEEE Global Conference on Signal and Information Processing*, 2013, pp. 551–554.

[12] M. Bürger, G. Notarstefano, and F. Allgöwer, "A polyhedral approximation framework for convex and robust distributed optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 384–395, 2014.

[13] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[14] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[15] ——, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[16] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Mathematical Programming*, vol. 144, no. 1-2, pp. 1–38, 2014.

[17] ——, "Parallel coordinate descent methods for big data optimization," *Mathematical Programming*, vol. 156, no. 1-2, pp. 433–484, 2016.

[18] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel selective algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1874–1889, 2015.

[19] A. Daneshmand, F. Facchinei, V. Kungurtsev, and G. Scutari, "Hybrid random/deterministic parallel algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 3914–3929, 2015.

[20] A. Patrascu and I. Necoara, "Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization," *Journal of Global Optimization*, vol. 61, no. 1, pp. 19–46, 2015.

[21] I. Necoara, "Random coordinate descent algorithms for multi-agent convex optimization over networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 2001–2012, 2013.

[22] P. Bianchi, W. Hachem, and F. Iutzeler, "A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2947–2957, Oct 2016.

[23] I. Notarnicola and G. Notarstefano, "Asynchronous distributed optimization via randomized dual proximal gradient," *IEEE Transactions on Automatic Control*, vol. 62, no. 5, pp. 2095–2106, May 2017.

[24] T. Erseghe, "A distributed and scalable processing method based upon ADMM," *IEEE Signal Processing Letters*, vol. 19, no. 9, pp. 563–566, 2012.

[25] ——, "A distributed and maximum-likelihood sensor network localization algorithm based upon a nonconvex problem formulation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 4, pp. 247–258, 2015.

[26] I. Necoara and D. Clipici, "Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 197–226, 2016.

[27] M. Todescato, G. Cavraro, R. Carli, and L. Schenato, "A robust block-Jacobi algorithm for quadratic programming under lossy communications," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 126–131, 2015.

[28] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in MPC and network flows," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 2004–2009, 2015.

[29] ——, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[30] F. Pasqualetti, R. Carli, and F. Bullo, "Distributed estimation via iterative projections with application to power network monitoring," *Automatica*, vol. 48, no. 5, pp. 747–758, 2012.

[31] F. C. Schweppe and J. Wildes, "Power system static-state estimation, part II: Approximate model." *IEEE Transactions on Power Apparatus and Systems*, vol. 89, no. 1, pp. 125–130, 1970.

[32] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, vol. 49, no. 3, pp. 237–252, 1998.

[33] S. H. Low and D. E. Lapsley, "Optimization flow control, I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[34] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[35] D. P. Bertsekas, *Nonlinear programming*. Athena scientific, 1999.

[36] J.-B. Hiriart-Urruty and C. Lemaréchal, "Convex analysis and minimization algorithms II: Advanced theory and bundle methods," *Grundlehren der mathematischen Wissenschaften*, vol. 306, 1993.