52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

Distributed partition-based optimization via dual decomposition

Ruggero Carli and Giuseppe Notarstefano

Abstract— **In this paper we consider a novel *partition-based* framework for distributed optimization in peer-to-peer networks. In several important applications the agents of a network system have to solve an optimization problem with two important features: (i) the dimension of the decision variable is a function of the network size, and (ii) the cost function and the constraints have a sparsity structure that is related to the sparsity of the graph. For this class of problems a straightforward application of existing methods would result in all the nodes reaching consensus on the minimizer. This approach has two inefficiencies: poor scalability and redundancy of shared information. Indeed, the dimension of the vector stored by each node and the size of the local problem to be solved depend on the network size. Furthermore, all the nodes compute the entire solution. In this paper we provide a preliminary contribution in developing and analyzing novel partition based algorithms. We propose a partition-based algorithm based on dual decomposition. We show that, exploiting the problem structure, the solution can be partitioned among the nodes so that each node stores a local copy of just 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.

Distributed optimization algorithms have a long history in the parallel and distributed computation literature, see, e.g. [1]. Recently they have been reconsidered in a new peer-to-peer set-up in which all the nodes take on the same role and perform the same actions. For example, no master is allowed and the communication topology is aimed to be the most general one. The first class of algorithms appearing in this new literature relies on primal sub-gradient iterations [2], [3]. Recently convergence proof for switching topologies has been proposed [4]. Following early results on Network Utility Maximization in communication networks [5], [6], see also [7] for a tutorial, dual decomposition methods have been applied in order to develop distributed algorithms in a pure peer-to-peer set-up. In [8] and references therein a tutorial on network optimization via dual decomposition can be found. A recent reference handling equality and inequality constraints is [9]. 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 recently proposed in the network context. A first distributed ADMM algorithm was proposed in [10], while a survey on this technique is [11]. More recently a convergence rate proof for distributed ADMM with sequential updates has been proposed [12]. A third class of algorithms is based on the exchange of active constraints among the network nodes. A constraints consensus has been proposed in [13] to solve linear, convex and general abstract programs, see also [14]. In [15] a distributed simplex has been proposed as an ad-hoc dual version of the constraints consensus for degenerate linear programs. Recently the constraint exchange idea was combined with dual decomposition and cutting-plane methods to solve distributed robust convex optimization problems via polyhedral approximations [16]. Finally, distributed algorithms based on Newton methods have been recently proposed to speed-up the computation [17], [18]. A fast Lipschitz approach is proposed in [19] for a suitable class of problems with the same speed-up aim.

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.

In this paper we investigate a class of problems of interest in several multi-agent applications in which this drawback can be overcome. In particular, the main contribution of this paper is twofold. First, we identify a new distributed optimization set-up for peer-to-peer network systems following the idea introduced in [20]. In the proposed framework the decision variable grows as the number of nodes in the network, but the cost function and the constraints have a special partitioned structure. That is, the cost function is separable and each local cost function depends only on the decision variables associated to the corresponding node and to its neighbors. Also, each local constraint involves only the decision variable of the associated node and its neighbors. We show that such structure is not derived just as a pure academic exercise, but vice-versa appears in several important application scenario. 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). Second, we provide an algorithm, based on dual decomposition, that is scalable. That is, the information stored and the computation performed by each node does not depend on the number of nodes as long as the node degree is bounded. We derive the algorithm by writing a suitable equivalent formulation of the original optimization problem. This equivalent formulation exploits the partitioned structure, thus leading to the scalable algorithm through a dual decomposition approach. We remark that our novel partition-based dual decomposition algorithm is inspired by [20] where the partition-based approach is applied to

The work by G. Notarstefano is partially supported by the project SOCIAL-ROBOTS under the program "5 per mille per la ricerca". The work by R. Carli is supported by the European Community's Seventh Framework Program under grant agreement n. 257462 HYCON2 Network of Excellence.

Ruggero Carli is with Department of Information Engineering, Università di Padova, Via Gradenigo 6/b, 35131 Padova, Italy carlirug@dei.unipd.it

Giuseppe Notarstefano is with the Department of Engineering, Università del Salento (University of Lecce), Via per Monteroni, 73100 Lecce, Italy, giuseppe.notarstefano@unisalento.it.

the ADMM algorithm.

The paper is organized as follows. In Section II we present the partition-based optimization framework and describe two motivating applications. In Section III we introduce the partition-based dual decomposition algorithm, discuss its scalability properties and analyze its convergence. In Section IV we provide a simulation scenario showing the effectiveness of the proposed algorithm.

II. Problem set up and motivating scenarios

A. Problem set up

We consider a network with set of nodes $I = \{1, \dots, n\}$ and fixed undirected communication graph $G = (I, E)$. We denote N_i the set of neighbors of node i, that is $N_i = \{j \in I \mid (i,j) \in E\}$.

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,

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

where each $f_i : \mathbb{R}^N \to \mathbb{R}$ is a strongly convex function and each $X_i \subset \mathbb{R}^N$ is a compact convex set. Observe that these assumptions on each f_i and X_i guarantee that problem in (1) has a unique solution. In our setup the local objective function f_i and the local constraint set X_i are known only to agent i.

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

$$x = \left[x_1^T, \dots, x_n^T\right]^T$$

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, referred to, hereafter, 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

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

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

B. Motivating examples

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

1) Distributed estimation in power networks: To describe this example we follow the treatment in [21].

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 $x \in R^n$ and $z \in R^p$ 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 $h()$ is a nonlinear measurement function, and where η is the noise measurement, which is traditionally assumed to be a zero mean random vector satisfying $\mathbb{E}[\eta\eta^T] = \Sigma > 0$. An optimal estimate of the network state coincides with the most likely vector x^* that solves equation (3). This static state estimation problem can be simplified by adopting the approximated estimation model presented in [22], which follows from the linearization around the origin of equation (3). Specifically,

$$z = Hx + v$$

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

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

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

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

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 m 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 $[H_{ij}]_{i,j=1}^m$, $x = \left[x_1^T, \dots, x_m^T\right]^T$ and $z = \left[z_1^T, \dots, z_m^T\right]^T$, where H_{ij}, $z_i \in \mathbb{R}^{p_i}$, $x_i \in \mathbb{R}^{n_i}$ and $\sum_{i=1}^m n_i = n$, $\sum_{i=1}^m p_i = p$. 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}, $j \in \{1, \dots, m\}$ and it is interested only in estimating the sub-state x_i. Moreover let $N_i = \{j \in \{1, \dots, m\} \mid H_{ij} \neq 0\}$. Observe that in general if $H_{ij} \neq 0$ then also $H_{ji} \neq 0$. Then by defining

$$f_i(x_i, \{x_j\}_{j \in N_i}) = \left(z_i - \sum_{j \in N_i} H_{ij} x_j\right)^T \left(z_i - \sum_{j \in N_i} H_{ij} x_j\right)$$

problem in (4) can be equivalently rewritten as

$$\operatorname*{argmin}_x f_i(x_i, \{x_j\}_{j \in N_i})$$

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 [5] and studied in [6] 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, $l \in L$. The network is shared by a set of n sources. Each source has a utility function $U_i(x_i)$ that is concave

increasing in its transmission rate x_i. The goal is to calculate source rates that maximize the sum of the utilities $\sum_{i=1}^{n} U_i(x_i)$ over x_i subject to capacity constraints. Formally, using a notation consistent with [6], let $L(i) \subset L$ be the set of links that source i uses and $N(l) = \{i \in \{1,\dots,n\} \mid l \in L(i)\}$ be the set of sources that use link l. Note that $l \in L(i)$ if and only if $i \in N(l)$. Also, let $I_i = [m_i, M_i]$ with $0 < m_i < M_i$ be the interval of minimum and maximum transmission rates of node i. The network flow optimization problem is given by

$$\begin{aligned} &\max_{x_i \in I_i} \sum_{i=1}^{n} U_i(x_i) \\ &\text{subj. to} \sum_{i \in N(l)} x_i \le c_l \qquad l \in \{1,\dots,L\}. \end{aligned} \tag{5}$$

In [6] 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. That is $(i,j) \in E$ if and only if there exists $l \in L$ such that $l \in L(i)$ and $l \in L(j)$. Under this condition optimization problem (5) can be rewritten as

$$\begin{aligned} &\max_{x_i \in I_i} \sum_{i=1}^{n} U_i(x_i) \\ &\text{subj. to} \sum_{j \in N(i)} a_{j,l}\, x_j \le c_l \quad l \in \{1,\dots,L(i)\}, \\ &\qquad\qquad\qquad\qquad\qquad i \in \{1,\dots,n\}, \end{aligned} \tag{6}$$

where $a_{j,l} = 1$ if agent j can use link l and $a_{j,l} = 0$ otherwise. Except for the maximization versus minimization, this problem is partition-based, 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 utility function to maximize. The resulting optimization problem turns to be

$$\begin{aligned} &\max \sum_{i=1}^{n} U(x_i) \\ &\text{subj. to} \sum_{j \in N_i \cup i} x_j \le r_i \qquad i \in \{1,\dots,n\}, \end{aligned}$$

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 [7] 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 [6], 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 partition-based approach the dual decomposition is used to enforce the compatibility constraints, whereas the capacity constraints are taken into account in the primal local minimization. This will be more clear in the next section after the presentation of the partition-based algorithm.

III. Partition based dual decomposition for distributed optimization

Before introducing our partition based algorithm, we recall the standard dual decomposition approach for distributed optimization. In order to solve an optimization problem with structure as in (1) in a distributed way, a common approach consists of writing an equivalent formulation of the problem, that is

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

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 network connectedness ensures equality between all $x^{(i)}$ and, in turn, the equivalence to (1). Notice that, since the constraints in (7) are set on the whole vectors, it turns out that the information on the whole vectors is spread and used over network, thus making this approach not scalable in the partition-based framework.

However, when considering a partition-based problem as in (2), because of the structure of f_i and X_i, $i \in \{1,\dots,n\}$, this wide diffusion is redundant. The idea is to exploit the partition-based structure to modify (7) in order to limit the range of equivalences among the auxiliary variables, and, in turn, their diffusion over the network. Specifically we enforce the state x_i to be indentical only between the neighboring nodes $j \in \mathcal{N}_i \cup \{i\}$ which potentially are interested to and use this information. Formally, we reformulate problem (2) as

$$\begin{aligned} &\min_{x} \sum_{i=1}^{n} f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j \in N_i}) \\ &\text{subj. to } (x_i^{(i)}, \{x_j^{(i)}\}_{j \in N_i}) \in X_i \qquad i \in \{1,\dots,n\}, \\ &\qquad x_i^{(i)} = x_i^{(j)} \\ &\qquad x_j^{(i)} = x_j^{(j)} \qquad\qquad j \in N_i,\ i \in \{1,\dots,n\}. \end{aligned} \tag{8}$$

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

Before proceeding with the algorithm presentation we discuss two key features in the structure of the above problem. First, following the standard approach in distributed dual decomposition we have reformulated problem (2) by introducing copies of the decision variables and added a set of "compatibility constraints" having the same sparsity as the communication graph. Differently from the standard approach, we do not add n copies of the entire decision vector x. For each agent i we add a local copy of the $i-th$ decision vector component and a local copy for each of the components of the neighboring agents. It is worth noting that the problem formulations (7) and (8), although equivalent, are different. In fact, (8) will lead to our partition-based algorithm.

Second, the constraints $x_i^{(i)} = x_i^{(j)}$ and $x_j^{(i)} = x_j^{(j)}$ for each agent i and neighboring agent j have the following meaning. The local copy of the component x_i (respectively x_j) hold by

agent i, $x_i^{(i)}$ (respectively $x_j^{(i)}$) must be equal to the local copy hold by j, $x_i^{(j)}$ (respectively $x_j^{(j)}$). 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.

We are now ready to introduce our partition based algorithm. To tackle problem (8), we derive a dual ascent algorithm that can be implemented distributedly by the nodes. To do that let us introduce the partial Lagrangian for problem (8), that is

$$L(\mathbf{x},\lambda) = \sum_{i=1}^{n} \Big[f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i}) + \sum_{j\in N_i} \big[\lambda_i^{(i,j)}(x_i^{(i)} - x_i^{(j)}) + \lambda_j^{(i,j)}(x_j^{(i)} - x_j^{(j)})\big]\Big]. \tag{9}$$

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

Next, we exploit the separability of the partial Lagrangian, which is formally established in the following lemma and that will lead to the proposed partition-based distributed algorithm.

Lemma 3.1: Given problem (8), the associated partial Lagrangian (9) is separable when the Lagrange multipliers are fixed. That is, it can be rewritten equivalently as

$$L(\mathbf{x},\lambda) = \sum_{i=1}^{n} \Big[f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i}) + x_i^{(i)} \sum_{j\in N_i} (\lambda_i^{(i,j)} - \lambda_i^{(j,i)}) + \sum_{j\in N_i} x_j^{(i)}(\lambda_j^{(i,j)} - \lambda_j^{(j,i)})\Big]. \tag{10}$$

Proof: The separability of the cost function follows directly by the structure of problem (2) and the choice of the new optimization variables. The second part needs some manipulation. We can rewrite the last two terms in (9) as follows:

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j\in N_i} \lambda_i^{(i,j)}(x_i^{(i)} - x_i^{(j)}) = \\
&\quad \sum_{j\in N_1} \lambda_1^{(1,j)} x_1^{(1)} - \sum_{j\in N_1} \lambda_1^{(1,j)} x_1^{(j)} + \\
&\quad \sum_{j\in N_2} \lambda_2^{(2,j)} x_2^{(2)} - \sum_{j\in N_2} \lambda_2^{(2,j)} x_2^{(j)} + \\
&\quad \dots + \\
&\quad \sum_{j\in N_n} \lambda_n^{(n,j)} x_n^{(n)} - \sum_{j\in N_n} \lambda_n^{(n,j)} x_n^{(j)}
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j\in N_i} \lambda_j^{(i,j)}(x_j^{(i)} - x_j^{(j)}) = \\
&\quad \sum_{j\in N_1} \lambda_j^{(1,j)} x_j^{(1)} - \sum_{j\in N_1} \lambda_j^{(1,j)} x_j^{(j)} + \\
&\quad \sum_{j\in N_2} \lambda_j^{(2,j)} x_j^{(2)} - \sum_{j\in N_2} \lambda_j^{(2,j)} x_j^{(j)} + \\
&\quad \dots + \\
&\quad \sum_{j\in N_n} \lambda_j^{(n,j)} x_j^{(n)} - \sum_{j\in N_n} \lambda_j^{(n,j)} x_j^{(j)}
\end{aligned}$$

From this exploited structure, it is clear that if an edge is present, say $(1,2) \in E$, the associated contributions in the two summations are

$$\begin{aligned}
&\lambda_1^{(1,2)}(x_1^{(1)} - x_1^{(2)}) + \lambda_2^{(2,1)}(x_2^{(2)} - x_2^{(1)}) \\
&\qquad + \lambda_2^{(1,2)}(x_2^{(1)} - x_2^{(2)}) + \lambda_1^{(2,1)}(x_1^{(2)} - x_1^{(1)}),
\end{aligned}$$

which can be rewritten as

$$\begin{aligned}
&x_1^{(1)}(\lambda_1^{(1,2)} - \lambda_1^{(2,1)}) + x_2^{(2)}(\lambda_2^{(2,1)} - \lambda_2^{(1,2)}) \\
&\qquad + x_2^{(1)}(\lambda_2^{(1,2)} - \lambda_2^{(2,1)}) + x_1^{(2)}(\lambda_1^{(2,1)} - \lambda_1^{(1,2)}).
\end{aligned}$$

The proof follows by rearranging the terms for all the edges in the same way. ∎

With the two equivalent expressions for the Lagrangian in hands, we are ready to introduced our *Partition-Based Dual Decomposition* algorithm (denoted hereafter as the PBDD algorithm).

Each node $i \in \{1,\dots,n\}$ stores and updates the primal variables $x_i^{(i)}$ and $x_j^{(i)}$, $j \in N_i$, and the dual variables $\lambda_i^{(i,j)}$ and $\lambda_j^{(i,j)}$, $j \in N_i$. The local estimates $x_i^{(i)}(t+1)$ and $x_j^{(i)}(t+1)$, $j \in N_i$, of the primal variables are computed by solving the local optimization problem

$$\begin{aligned}
\min_{(x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i})\in X_i} \quad & f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i}) \\
& + \sum_{j\in N_i} \Big[x_i^{(i)}(\lambda_i^{(i,j)}(t) - \lambda_i^{(j,i)}(t)) \\
& \qquad + x_j^{(i)}(\lambda_j^{(i,j)}(t) - \lambda_j^{(j,i)}(t))\Big].
\end{aligned} \tag{11}$$

The dual variables are obtained by applying a gradient ascent rule given by

$$\begin{aligned}
\lambda_i^{(i,j)}(t+1) &= \lambda_i^{(i,j)}(t) + \alpha(t)(x_i^{(i)}(t+1) - x_i^{(j)}(t+1)) \\
\lambda_j^{(i,j)}(t+1) &= \lambda_j^{(i,j)}(t) + \alpha(t)(x_j^{(i)}(t+1) - x_j^{(j)}(t+1)).
\end{aligned} \tag{12}$$

where $\{\alpha(t)\}_{t\in\mathbb{N}}$ is a sequence of positive real numbers defining the step-sizes of the dual variables updates. Before studying the convergence properties of the proposed algorithm, let us comment on its scalability and how it compares with standard dual subgradients algorithms.

First, observe that each node has to keep in memory the set of variables $x_i^{(i)}$, $\left\{x_j^{(i)}\right\}_{j\in N_i}$, $\left\{\lambda_i^{(i,j)}, \lambda_j^{(i,j)}\right\}_{j\in N_i}$, namely a number of variables equal to $1 + 3|N_i|$.

Second, as it is stated, the algorithm needs two communication rounds to perform the local minimization and the gradient ascent step. Indeed, at every iteration of the PBDD algorithm each agent i has to perform the following four actions in order

(i) it gathers the values $\lambda_i^{(j,i)}, \lambda_j^{(j,i)}$ from its neighbors $j \in N_i$;
(ii) it updates the value $x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i}$ according to (11);
(iii) it gathers the updated values $x_i^{(j)}, x_j^{(j)}$ from its neighbors $j \in N_i$;
(iv) it updates the variables $\{\lambda_i^{(i,j)}, \lambda_j^{(i,j)}\}_{j\in N_i}$ according to (12).

Next we provide a simplification on the communication complexity of the PBDD algorithm. In particular, the following lemma is useful to show that the partition-based dual decomposition algorithm needs only one communication round to perform both the ascent and local minimization parts.

Lemma 3.2: Consider the dual ascent algorithm defined in (11) and (12). Let us assume that, for $i \in \{1,\dots,n\}$ and for $j \in N_i$,

$$\begin{aligned}
\lambda_i^{(i,j)}(0) &= -\lambda_i^{(j,i)}(0) \\
\lambda_j^{(i,j)}(0) &= -\lambda_j^{(j,i)}(0)
\end{aligned} \tag{13}$$

Then $\lambda_i^{(i,j)}(t) = -\lambda_i^{(j,i)}(t)$ and $\lambda_j^{(i,j)}(t) = -\lambda_j^{(j,i)}(t)$, $j \in N_i$, for all $t \geq 0$.

Proof: The proof follows straight by the structure of the multipliers update in equation (12). ■

The main consequence of the above lemma is that node i does not need to gather the values $\lambda_i^{(j,i)}$ and $\lambda_j^{(j,i)}$ from its neighbors $j \in N_i$, provided the initializations in (13) hold true. Indeed, in this case, $\lambda_i^{(j,i)}(t)$ and $\lambda_j^{(j,i)}(t)$ are the opposite of, respectively, $\lambda_i^{(i,j)}(t)$ and $\lambda_j^{(i,j)}(t)$ and the latter two variables are updated by node i based only on the primal variables $x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i}$.

Notice that, due to the partitioned structure of the multipliers, the initialization condition in (13) can be ensured by a local agreement among neighboring agents.

Thus, the PBDD algorithm can be formally described as follows.

Processor states: For $i \in \{1,\dots,N\}$, node i stores in memory the variables $x_i^{(i)}$, $\left\{x_j^{(i)}\right\}_{j\in N_i}$, $\left\{\lambda_i^{(i,j)}, \lambda_j^{(i,j)}\right\}_{j\in N_i}$;

Initialization: Every nodes initializes the variables it stores in memory to arbitrary values;

Primal variable update: For $t \in \mathbb{N}$ node i computes $x_i^{(i)}(t+1)$ and $x_j^{(i)}(t+1)$, $j \in N_i$ by solving the optimization problem

$$\min_{(x_i^{(i)},\{x_j^{(i)}\}_{j\in N_i})\in X_i} f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j\in N_i}) + \sum_{j\in N_i}\left[2\,x_i^{(i)}\,\lambda_i^{(i,j)}(t) + 2x_j^{(i)}\lambda_j^{(i,j)}(t)\right]. \tag{14}$$

Transmission action: Node i, $i \in \{1,\dots,N\}$, broadcasts to all its neighbors the updated values $x_i^{(i)}(t+1), \{x_j^{(i)}(t+1)\}_{j\in N_i}$. It also gathers from its neighbors the estimates, $\{x_i^{(j)}(t+1), \{x_j^{(j)}(t+1)\}_{j\in N_i}$.

Dual variables update: Node i, $i \in \{1,\dots,N\}$, based on the information received from its neighbors, updates the variables $\{\lambda_i^{(i,j)}, \lambda_j^{(i,j)}\}_{j\in N_i}$ as in (12).

The convergence properties of the PBDD algorithm are established in the following theorem.

Theorem 3.3: Consider Problem (2) and the PBDD algorithm. Assume the functions f_i, $i \in \{1,\dots,n\}$, are strongly convex and the set X_i, $i \in \{1,\dots,n\}$ are compact and convex. Let $\bar{x} = \left[\bar{x}_1^T,\dots,\bar{x}_n^T\right]^T$ be the unique optimal solution of (2). Assume the step-sizes $\alpha(t)$ used in the dual variables updates of the PBDD algorithm are constant, i.e, $\alpha(t) = \alpha$ for some $\alpha \in \mathbb{R}$. Then there exist $\alpha^* > 0$, such that, if $0 < \alpha < \alpha^*$, then the estimates $\hat{x}_i^{(i)}(t)$, $i \in \{1,\dots,n\}$, converge to the optimal solution, i.e, for all $i \in \{1,\dots,n\}$,

$$\lim_t \hat{x}_i^{(i)}(t) = \bar{x}_i.$$

Proof: To prove the statement we let $\mathbf{x}$ be the vector obtained by stacking together all the primal variables $x_i^{(i)}$ and $x_j^{(i)}$, $j \in N_i$, for all $i \in \{1,\dots,n\}$. Denoting $y^{(i)}$ the vector containing $x_i^{(i)}$ and $x_j^{(i)}$, $j \in N_i$, we can write $\mathbf{x} = [(y^{(1)})^T \dots (y^{(n)})^T]^T$. Consistently, we let λ be the vector obtained by stacking together all the local dual variables $\lambda_i^{(i,j)}$ and $\lambda_j^{(i,j)}$, $j \in N_i$ for all $i \in \{1,\dots,n\}$. Thus, we can formally write the Lagrangian of problem (8) as $L(\mathbf{x},\lambda)$ with expression as in (9). Now, to solve problem (8) a dual ascent algorithm can be applied. The primal minimization can be formally written as

$$\min_{\mathbf{x}\in\cap_{i=1}^n X_i} L(\mathbf{x},\lambda).$$

By Lemma 3.1 the above problem has a separable structure and can be written as

$$\min_{y^{(1)}\in X_1 \dots y^{(n)}\in X_n} \sum_{i=1}^n L_i(y^{(i)},\lambda),$$

where $L_i(y^{(i)},\lambda)$ is suitably defined according to (10). Thus, the primal minimization can be decomposed into n separated subproblems each one having the structure in (11).

To conclude the proof, we just observe that by applying the gradient ascent update to the Lagrangian in the form (9), equations (12) are obtained. ■

Remark 3.4: As previously stated, the PBDD algorithm has been inspired by [20], where the partition based approach has been applied to the ADMM algorithm. It is worth mentioning that, in the ADMM algorithm the augmented Lagrangian present additional quadratic terms multiplied by suitable parameters. Tuning these parameters is of fundamental importance in the ADMM algorithm to establish its convergence properties. To the best of our knowledge there are no distributed strategies to set the value of these additional parameters making the computational effort required by the ADMM algorithm higher than the dual approach considered in this paper. For this reason we believe that, in some scenarios, the PBDD algorithm might be preferable to the partition based ADMM algorithm even though the latter might be faster to converge than the former.

IV. Simulations

In this section we provide an example showing the effectiveness of the PBDD algorithm. Let $G = (I, E)$ be an undirected random geometric graph generated by choosing $N = 20$ points uniformly distributed in the unit square, and then placing an edge between each pair of points at distance less than 0.25. Assume that, for $i \in \{1,\dots,N\}$, $x_i \in \mathbb{R}$, and that the function f_i has the form

$$f_i\left(x_i, \{x_j\}_{j\in N_i}\right) = (x_i - a_{ii})^2 + \sum_{j\in N_i}(x_j - a_{ij})^2$$

where the real numbers a_{ii}, $\{a_{ij}\}_{j\in N_i}$ are randomly generated in the interval $[0, 10]$. Notice that the optimal solution of the problem

$$\min_x \sum_{i=1}^N f_i\left(x_i, \{x_j\}_{j\in N_i}\right)$$

is $x^* = [x_1^*,\dots,x_N^*]$ where

$$x_i^* = \frac{1}{|N_i|+1}\left(a_{ii} + \sum_{j\in N_i} a_{ji}\right).$$

We implemented the PBDD algorithm starting from initial conditions equal to 0. Moreover in the update of the dual variables, we used a constant step-size α equal to 0.1. The obtained results are reported in Figure 1, where we plotted the behavior of the quantity

$$J(t) = \sum_{i=1}^n \left(\hat{x}_i^{(i)}(t) - x_i^*\right)^2.$$

Observe that $J(t)$ tends to 0, namely, $\hat{x}_i(t) \to x_i^*$, for all $i \in \{1,\dots,n\}$.

V. Conclusions

In this paper we have proposed a dual decomposition based algorithm for a novel partition-based 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 the 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 resulting

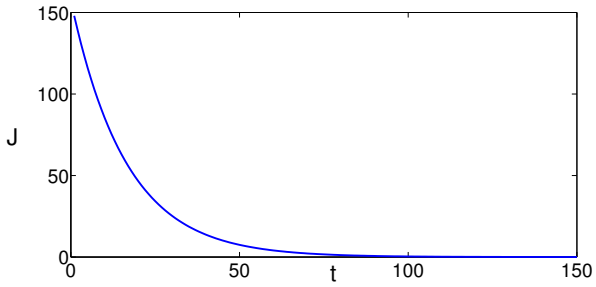

Fig. 1.

algorithm is distributed and scalable and it is shown to be provably convergent under standard assumptions on the cost functions and on the constraints sets.

The objective of our future research will be to provide asynchronous versions of the novel algorithm we proposed and to analyze its convergence speed and robustness to noisy and unreliable communications.

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computations: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[2] A. Nedic 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. Nedic, 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] P. Lin and W. Ren, "Distributed subgradient projection algorithm for multi-agent optimization with nonidentical constraints and switching topologies," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 6813–6818.

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[12] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 5445–5450.

[13] G. Notarstefano and F. Bullo, "Distributed abstract optimization via constraints consensus: Theory and applications," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2247–2261, October 2011.

[14] ——, "Network abstract linear programming with application to minimum-time formation control," in *IEEE Conf. on Decision and Control*, New Orleans, LA, December 2007, pp. 927–932.

[15] M. Bürger, G. Notarstefano, F. Allgöwer, and F. Bullo, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, p. 22982304, Sept 2012, available at dx.doi.org/10.1016/j.automatica.2012.06.040.

[16] M. Bürger, G. Notarstefano, and F. Allgöwer, "Distributed robust optimization via cutting-plane consensus," in *IEEE Conf. on Decision and Control*, Maui, Hawaii, USA, December 2012.

[17] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network optimization," in *Proc. American Control Conference*, San Francisco, CA, June 2011, pp. 266–2668.

[18] F. Zanella, D. Varagnolo, A. Cenedese, P. Gianluigi, and L. Scenato, "Newton-raphson consensus for distributed convex optimization," in *Proc. 50th IEEE Conf. on Decision and Control*, Orlando, Florida, December 2011, pp. 5917 – 5922.

[19] C. Fischione, "Fast-lipschitz optimization with wireless sensor networks applications," *Automatic Control, IEEE Transactions on*, vol. 56, no. 10, pp. 2319–2331, 2011.

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

[21] 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.

[22] 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.