2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Multi-agent distributed optimization algorithms for partition-based linear programming (LP) problems

Ruggero Carli, *Member, IEEE,* Kasım Sinan Yıldırım, and Luca Schenato, *Member, IEEE*

***Abstract*— The paper addresses the problem of multi-agent distributed solutions for a class of linear programming (LP) problems which include box constraints on the decision variables and inequality constraints. The major difference with existing literature on distributed solution of LP problems is that each agent is expected to compute only a single or few entries of the global minimizer vector, often referred as a partition-based optimization. This class of LP problems is relevant in different applications such as optimal power transfer in remotely powered battery-less wireless sensor networks, minimum energy LED luminaries control in smart offices, and optimal temperature control in start buildings. Via a suitable approximation of the original LP problem, we propose three different primal-dual distributed algorithms based on dual gradient ascent , on the methods of multipliers and on the Alternating Direction Methods of Multipliers. We discuss the computational and communication requirements of these methods and we provide numerical comparisons.**

I. Introduction

The continuous reduction of cost of electronic devices capable of sensing, actuation and computation and their increased connectivity via wireless, mobile and internet communication are posing the basis for the creation of large scale networks of smart devices that can cooperate to achieve a common goal, an area often referred by different communities with different names as multi-agent smart systems, internet-of-things, cyber-physical systems, networked control systems. Many cooperative tasks in these systems reduce to large-scale optimization problems that can be solved only via cooperation. One of the major challenges for these optimization problems is scalability, i.e. the computational, communication and memory complexity per device should be almost independent of the number of agents involved. Fortunately, many of these optimization problems require each agent to compute only a subset of the decision variables involved and these variables are only locally coupled via physical proximity, yet the problem is challenging since the decision of one agent can affect the behaviour of the network via cascading effects. Example of such problems are the control traffic lights in a smart cities, the control of power transmission in wireless sensor networks, the control of active and reactive power injection of photovoltaic panels in smart energy grids, and environment control in large smart building.

In this work, we address a specific class of optimization problems, namely linear programming (LP) problems where the decision variables needs to satisfy box constraints and a number of inequalities each involving a small set of the decision variables. This specific optimization problems arise from diverse applications such as optimal power transfer in remotely powered wireless sensor networks [1], optimal LED lighting control in smart offices [2], and optimal temperature control in smart buildings [3], as explicitly shown in Section III. Distributed optimization has received the attention of large community of researches and many solutions and algorithms have been proposed: distributed subgradient methods [4], Lagrangian-based methods [5], consensus-based methods [6], and distributed linear programming [7], just to name few popular classes. However, most of these approaches require that each agent involved computes the whole global minimizer. This is reasonable for some specific distributed optimization problems such as map-building and classification [4], [5] where the number of decision variables is independent of the number of agents involved and the difficulty arises from the fact that data is physically distributed among these agents. However, these approaches are of little use in our context since the number of decision variables scale with the number of agents, thus giving rise to an unscalable solution. Recently, alternative algorithms have appeared exploiting the observation that in some optimization problems each agent is required to compute only a subset of the decision variables [8], [9], [10]. These approaches are often referred as *partition-based*. Nonetheless, they are not directly applicable to our specific problem since they are either developed for unconstrained problems [8], or admit only equality constraints [9], [10].

In this work is to propose three Lagrangian-based algorithms based on well-known optimization strategies, namely dual gradient ascent (DGA), multiplier methods (MM), and alternating direction multiplier methods (ADMM) that are suitable for partition-based implementation. The first contribution of this work is to strongly-convexify the original LP problem into a quadratic programming problem (QP) which exploits the box constraints to explicitly quantify bounds on the performance degradation incurred by this approximation. The second contribution is to transform the inequality constraints into equality constraints via the introduction of additional slack variables so that MM and ADMM ideas can be applied. The third contribution is to substitute the standard optimization step in the primal variable in MM and ADMM, which cannot be implemented with a partition-based architecture, with a gradient descent step. For the DGA algorithm we provide convergence guarantees as well as step-size tuning procedures, while for MM and ADMM we

Ruggero Carli and Luca Schenato are with the Department of Information Engineering, University of Padova, Italy. Kasım Sinan Yıldırım is with the Embedded Software Group, Delft University of Technology, The Netherlands and the Department of Computer Engineering, Ege University, Turkey.

provide only numerical simulations. Numerical comparisons show that these algorithms have pros and cons in terms of speed of convergence, computational and communication complexity, which deserve further exploration given the wide applicability and relevance of the original LP problem.

II. Problem Description

Consider the following linear programming problem subject to linear constrains

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{a}^T\mathbf{x} \\ s.t. \quad & B\mathbf{x} \leq \mathbf{d} \\ & x_{min}\mathbf{1} \leq \mathbf{x} \leq x_{max}\mathbf{1} \end{aligned} \tag{1}$$

where $\mathbf{a} = [a_1, \ldots, a_N]^T \in \mathbb{R}^N$, $\mathbf{x} = [x_1, \ldots, x_N]^T \in \mathbb{R}^N$, $B \in \mathbb{R}^{M \times N}$, $\mathbf{d} = [d_1, \ldots, d_N]^T \in \mathbb{R}^M$, $\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^N$ and $x_{min}, x_{max} \in \mathbb{R}$, and where the inequalities are component-wise. Assume that the set

$$\mathcal{X} = \left\{\mathbf{x} \in \mathbb{R}^N : \ B\mathbf{x} \leq \mathbf{d}, \ \ x_{min}\mathbf{1} \leq \mathbf{x} \leq x_{max}\mathbf{1}\right\},$$

is not empty, namely, the optimization problem is feasible. Observe that, under the assumption $\mathcal{X} \neq \emptyset$, the optimization problem in (1) might not have a unique minimizer $\mathbf{x}^*$ being the cost function $\mathbf{a}^T\mathbf{x}$ simply convex.

Consider now two sets of nodes $V = \{v_1, \ldots, v_N\}$ and $\mathcal{S} = \{s_1, \ldots, s_M\}$. Nodes in V are responsible for the state variable $\mathbf{x}$; specifically, state variable x_i is stored in memory by node v_i that can modify its value according to the output of some algorithm aiming at solving the optimization problem in (1). Instead nodes in $\mathcal{S}$ are responsible for the set of constraints $B\mathbf{x} \leq \mathbf{d}$ to be satisfied; in particular constraint $\sum_{j=1}^N B_{hj}x_j \leq d_h$ is *monitored* by node s_h. Constraints $B\mathbf{x} \leq \mathbf{d}$ are *soft constraints*, meaning that they might be violated during the transients of algorithms. Instead constraints $x_{min}\mathbf{1} \leq \mathbf{x} \leq x_{max}\mathbf{1}$ are *hard constraints*, namely, each node v_i must guarantee that $x_{min} \leq x_i \leq x_{max}$ is not violated at any time.

We assume the nodes can periodically communicate with each other, and based on the exchanged information, they take actions to collaboratively solve the above optimization problem. In particular, the admissible communications among nodes are described by the communication graph $\mathcal{G}_{V\mathcal{S}}$ which models the communication between nodes in V with nodes in $\mathcal{S}$; more precisely $\mathcal{E}_{V\mathcal{S}} \subseteq V \times \mathcal{S}$ and $(v_i, s_h) \in \mathcal{E}_{V\mathcal{S}}$ if and only if $B_{ih} \neq 0$, namely, nodes $v_i \in V$ and $s_h \in \mathcal{S}$ can exchange information with each other if and only if $B_{ih} \neq 0$.

Let us define $\mathcal{N}_{v_i} = \{s_j \in \mathcal{S} \,|\, (v_i, s_j) \in \mathcal{E}_{V\mathcal{S}}\}$, the set of neighbors of v_i, and $\mathcal{N}_{s_h} = \{v_j \in V \,|\, (v_j, s_h) \in \mathcal{E}_{V\mathcal{S}}\}$, the set of neighbors of node s_h. Observe that, if node $s_h \in \mathcal{N}_{v_i}$ then $v_i \in \mathcal{N}_{s_h}$ and vice-versa.

The goal of this paper is to design iterative and distributed algorithms that leads the state $\mathbf{x}$ to an optimal solution of the problem in (1). Beside the fact that each single node has just a local knowledge of the optimization problem to be solves (i.e., nodes V stores in memory only a component of the state variable and nodes in $\mathcal{S}$ monitor only a constraint), by distributed we mean that each node in V is allowed to communicate only with its neighbors in $\mathcal{S}$ and vice-versa. Observe that we do not require communications either among nodes in V or among nodes in $\mathcal{S}$. In this sense we seek for algorithms with limited communication requirements.

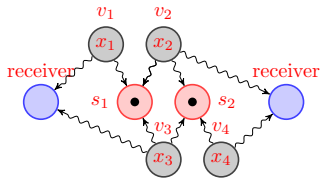

Fig. 1. A sample wireless power transfer network where the black nodes represent the wireless chargers, blue nodes represent RF-powered devices and the red nodes represent the sensors. While wireless chargers are charging the receivers, they also contribute to the EMR value measured by the sensors.

III. Application Examples

In this section we present three examples where the problems to be solved fits into the above scenario.

A. Wireless Charging

Removing batteries from wireless networks completely and powering the embedded devices forming these networks directly from energy harvesting alone is an area of great interest. In particular, a new class of RF-powered embedded devices that can sense, compute and communicate by means of radio frequency (RF) energy harvesting are becoming popular [11]. In order to provide continuous energy to the RF-powered devices, a dedicated network of wireless chargers should be deployed so that they can charge nearby devices collaboratively to maximize the total transmitted power. However, the wireless chargers should also ensure that the electromagnetic radiation (EMR)—measured by the sensors deployed at particular points—always satisfy the RF-exposure regulations [1]. Therefore, the operation of the wireless charging network can be formulated as an optimization problem where the individual chargers maximize the total transmitted power meanwhile satisfying the EMR regulations [12], [13]. Since new chargers as well as sensors and energy receiver devices can be introduced to the charging system dynamically, a one-shot centralized solution of the aforementioned problem is not suitable. Figure 1 presents a sample wireless charging network, and in turn an instance of the distributed optimization problem. In this case, the state variables $\mathbf{x} = [x_1, \ldots, x_N]^T$ represent the individual power transmission levels of the wireless chargers in $V = [v_1, \ldots, v_N]$. Note that $x_{\min}$ and $x_{\max}$ denote the minimum and the maximum power levels of the wireless chargers—they are the hard constraints which are hardware dependent and not modifiable. We can rewrite the objective function as $\mathbf{a}^T = \mathbf{1}^T A$ where the matrix A holds the relation between the individual charger power and the received power by each single energy receiver inside the power transmission range—which is inversely proportional to the distance between them. The set $\mathcal{S} = [s_1, \ldots, s_M]$ denotes the set of sensors that can measure the EMR value and the matrix B holds the relationship between the charger power and EMR value measured by the sensors. Hence, sensor s_h measures $\sum_{j=1}^N B_{hj}x_j$ that

should be smaller than the d_h— the h-th entry of the vector $\mathbf{d}$ holding the EMR threshold, which is the soft constraint that might be violated occasionally. The entries of the matrix B are a decreasing function of the distance from the transmitters to the receiver and represent the power loss coefficient due to the medium.

B. Lighting control

In large offices up to hundreds of LED luminaries are being installed in new buildings or are being replacing more traditional neon luminaries in old buildings. The intensity of the light (lumens) of these luminaries can be independently controlled and the total amount of energy consumed to illuminate the office is proportional to the sum of the light intensity of all the lamps. The objective in future smart buildings is to minimize this energy while satisfying a minimum light intensity at each desk depending when a person is present [2]. Typically, the number of desks is smaller then the number of luminaries and not all desks are occupied. Moreover, additional illumination could arrive from windows during the day, which we indicate with d_i^{sun} for the i-th desk. Since the light intensity of the desk can be well approximated as a linear combination of the light intensity generated by local luminaries, the objective falls exactly in the framework described in section II where $\mathbf{a} = \mathbf{1}$, $x_{min} = 0, x_{max} = P_{max}$ where P_{max} is the maximum luminance power that a luminary can provide, $B = -B', \mathbf{d} = -\mathbf{d}' + \mathbf{d}^{sun}$ where $B'\mathbf{x} + \mathbf{d}^{sun} \geq \mathbf{d}'$ represents the requirement that the light intensity measured at each desk being greater than a specific threshold depending whether a person is present or not. The intensity of light of i-th desk is the sum of the contribution of all luminaries that affects it corresponding to the entries of the i-th row of B and the light coming from the windows d_i^{sun}.

C. Temperature control

In large buildings multiple heaters or air conditioning inlets are presents. The use of wireless sensor networks equipped with temperature sensors could provide a finer resolution of temperature distribution among different rooms and areas than what is typically achieved today with fewer cabled temperature sensors. Such additional resolution can be used to improve comfort as well as minimizing the energy used to heat or cool them. Today's temperature is achieved by means of simple decentralized controller, i.e. the air conditioning fan velocity is adjusted based an a single temperature measurement. A more refined approach would try to model the effect of each air conditioning heat flow in the temperature at multiple locations [3]. Let us indicate with x_i the heat flow generated by each of the N inlet. Assuming a steady-state condition, a reasonable model for the temperature at some M location $\mathbf{T} = [T_1, \ldots, T_M]^T$ can be written as $\mathbf{T} = B'\mathbf{x} + \mathbf{q}T_a$ where the coefficient of B and $\mathbf{q}$ depends on the building materials and planimetry, and T_a is the external temperature. The objective is to regulate the heat flows given by $\mathbf{x}$ to guarantee $\mathbf{T}_{min} \leq \mathbf{T} \leq \mathbf{T}_{max}$ where $\mathbf{T}_{min}, \mathbf{T}_{max}$ are based on the preference of the people or the presence of people in a specific area. Assuming that the energy expenditure is simply the sum of the heat flow provided, this problem can be easily cast as in Section II.

IV. A DUAL-ASCENT LIKE APPROACH

Observe that the LP problem (1) can be equivalently reformulated as

$$\begin{aligned} \mathbf{x}^* := \underset{\mathbf{x}\in\bar{\mathcal{X}}}{\operatorname{argmin}} \quad & \mathbf{a}^T\mathbf{x} \\ s.t. \quad & B\mathbf{x} \leq \mathbf{d} \end{aligned} \tag{2}$$

where $\bar{\mathcal{X}} := \{\mathbf{x} |\ x_{min}\mathbf{1} \leq \mathbf{x} \leq x_{max}\mathbf{1}\}$ is a closed convex set, more specifically a hypercube. A standard approach in constrained convex optimization is to introduce the Lagrangian

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{a}^T\mathbf{x} + \boldsymbol{\lambda}^T(B\mathbf{x} - \mathbf{d})$$

where $\boldsymbol{\lambda} \in \mathbb{R}^M$, and the corresponding dual function

$$q(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\bar{\mathcal{X}}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

Let $\boldsymbol{\lambda}^*$ be the maximizer of $q(\boldsymbol{\lambda})$, namely,

$$\boldsymbol{\lambda}^* := \arg\max_{\boldsymbol{\lambda}\geq 0} q(\boldsymbol{\lambda}).$$

From convex optimization theory it also follows that there is no duality gap between the primal and the dual problem, i.e.: $q(\boldsymbol{\lambda}^*) = \mathbf{a}^T\mathbf{x}^*$. At this point, one might be tempted to apply a standard primal-dual coupled iterative algorithm to find a solution to both the primal and dual problem as follows:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\bar{\mathcal{X}}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^k) \tag{3}$$

$$\boldsymbol{\lambda}^{k+1} = \max\{\mathbf{0}, \boldsymbol{\lambda}^k + \rho^k(B\mathbf{x}^{k+1} - \mathbf{d})\} \tag{4}$$

where ρ^k is a (possibly time-varying) step size for the dual ascent, and the $\max$ operator has to be interpreted component-wise. The previous algorithm however does not guarantee to provide an optimal solution of the primal problem, the problem being that $\mathbf{x}^{k+1}$ might not be unique since $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^k)$ is simply convex in $\mathbf{x}$. In fact, although it can be shown that for a suitable decreasing step-size ρ^k, we have $\boldsymbol{\lambda}^k \to \boldsymbol{\lambda}^*$, this does not help to guarantee $\mathbf{x}^k \to \mathbf{x}^*$ if $\operatorname{argmin}_{\mathbf{x}\in\bar{\mathcal{X}}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*)$ does not provide a unique solution.

As a consequence, we propose to approximate the objective function by adding a *regularization term* which would make the primal problem *strongly convex*.[1] Consider the following approximated optimization problem:

$$\begin{aligned} \mathbf{x}_\theta^* := \underset{\mathbf{x}\in\bar{\mathcal{X}}}{\operatorname{argmin}} \quad & J_\theta(\mathbf{x}) := \mathbf{a}^T\mathbf{x} + \tfrac{\theta}{2}\|\mathbf{x} - x_{ave}\mathbf{1}\|^2 \\ s.t. \quad & B\mathbf{x} \leq \mathbf{d} \end{aligned} \tag{5}$$

where $x_{ave} = (x_{min} + x_{max})/2$. Moreover consider the corresponding Lagrangian and dual functions:

$$\mathcal{L}_\theta(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{a}^T\mathbf{x} + \boldsymbol{\lambda}^T(B\mathbf{x} - \mathbf{d}) + \frac{\theta}{2}\|\mathbf{x} - x_{ave}\mathbf{1}\|^2 \tag{6}$$

$$q_\theta(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\bar{\mathcal{X}}} \mathcal{L}_\theta(\mathbf{x}, \boldsymbol{\lambda}). \tag{7}$$

[1] We refer the reader to [13], [14] for the detailed explanation and proofs.

Let $\mathbf{x}^\theta(\lambda) := \text{argmin}_{\mathbf{x}\in\bar{\mathcal{X}}}\,\mathcal{L}_\theta(\mathbf{x},\boldsymbol{\lambda})$ where $\boldsymbol{\lambda}\in\mathbb{R}^M, \theta > 0$. Then the vector $\mathbf{x}^\theta(\boldsymbol{\lambda}) = [x_1^\theta(\lambda),\ldots,x_N^\theta]^T$ is unique and it is given by:

$$\mathbf{x}^\theta(\lambda) = \text{Proj}_{\bar{\mathcal{X}}}\left(x_{ave}\mathbf{1} + \frac{1}{\theta}(-\mathbf{a} - B^T\boldsymbol{\lambda})\right) \tag{8}$$

where $\text{Proj}_{\bar{\mathcal{X}}}$ is the projection operator on the convex set $\bar{\mathcal{X}}$. The algorithm we propose to solve the approximated optimization problem is given by the following two iterative updates:

$$\begin{aligned}\mathbf{x}^{k+1} &= \text{Proj}_{\bar{\mathcal{X}}}\left(x_{ave}\mathbf{1} + \frac{1}{\theta}(-\mathbf{a} - B^T\boldsymbol{\lambda}^k)\right)\\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \rho(B\mathbf{x}^{k+1} - \mathbf{d})\end{aligned} \tag{9}$$

Notice that the vector $\boldsymbol{\lambda}$ is composed by the M Lagrange multipliers $\lambda_1,\ldots,\lambda_M$ where the multiplier λ_ℓ is associated to the constraint $\sum_{j=1}^N B_{\ell j}x_j \le d_\ell$; we assume that λ_ℓ is stored in memory and updated by node s_ℓ. Observe that Equations in (9) can be rewritten component-wise as:

$$x_i^{k+1} = \text{Proj}_{[x_{min},x_{max}]}\left[x_{ave} - \frac{1}{\theta}\left(a_i + \sum_{h\,:\,s_h\in\mathcal{N}_{v_i}} B_{hi}\lambda_h^k\right)\right] \tag{10}$$

$$\lambda_\ell^{k+1} = \lambda_\ell^k + \rho\left(\sum_{j\,:\,v_j\in\mathcal{N}_{s_\ell}} B_{\ell j}x_j^{k+1} - d_\ell\right). \tag{11}$$

It follows that node v_i to update its state needs to receive the values of the Lagrange multipliers only from nodes in $\mathcal{N}_{v_i}$; similarly node s_ℓ to update the corresponding Lagrange multiplier, requires the values of the states stored in memory only by the nodes in $\mathcal{N}_{s_\ell}$.

Proposition 1: Consider the algorithm (9). If

$$\rho \le \frac{2\,\theta}{\|B\|_\infty\|B\|_1},$$

then $\lim_{k\to\infty}\mathbf{x}^k = \mathbf{x}_\theta^*$.

V. Multipliers method

As in the previous algorithm, each node s_ℓ stores in memory a Lagrange multiplier λ_ℓ associated to the constraint $\sum_{j=1}^N B_{\ell j}x_j \le d_\ell$. Now, given $c > 0$, we introduce the augmented Lagrangian as follows:

$$\mathcal{L}_c(\mathbf{x},\boldsymbol{\lambda}) = \mathbf{a}^T\mathbf{x} + \frac{1}{2c}\sum_{\ell=1}^M\left\{\left[\max\left\{0,\lambda_\ell + c\left(\sum_{j=1}^N B_{\ell j}x_j - d_\ell\right)\right\}\right]^2 - \lambda_\ell^2\right\}.$$

Next, we describe an algorithm that iteratively updates the pair $(\mathbf{x},\boldsymbol{\lambda})$ to reach a saddle point of $\mathcal{L}_c$.[2]

Let $(\mathbf{x}^k,\boldsymbol{\lambda}^k)$ denote the values of the powers and of the multipliers at the k-th iteration. Then, the vector of the Lagrange multipliers is updated as

$$\boldsymbol{\lambda}^{k+1} = \max\left\{0, \boldsymbol{\lambda}^k + c\left(B\mathbf{x}^k - \mathbf{d}\right)\right\}. \tag{12}$$

Based on $\boldsymbol{\lambda}^{k+1}$ we would like to set $\mathbf{x}^{k+1}$ equal to the minimizer of the following problem:

$$\begin{aligned}\min_x \quad & \mathcal{L}_c\left(\mathbf{x},\boldsymbol{\lambda}^{k+1}\right)\\ s.t. \quad & x_{min}\mathbf{1} \le \mathbf{x} \le x_{max}\mathbf{1}\end{aligned} \tag{13}$$

[2]We refer the reader to [12] for the details.

In general, computing in a distributed way the minimizer of the above problem is a difficult task. For this reason, we limit ourselves to compute an approximated solution. In particular we resort to a projected gradient algorithm to update the values of the powers toward the optimal solution of (13). Observe that

$$\frac{\partial\mathcal{L}}{\partial\mathbf{x}} = \mathbf{a} + B^T\max\left\{0,\lambda^{k+1} + c\left(B\mathbf{x}^k - \mathbf{d}\right)\right\}$$

where the max is meant component-wise. Hence, given a pair $(\mathbf{x},\lambda)$ we first compute $\bar{\mathbf{x}} = \mathbf{x}^k - \alpha\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ where α is a given step size. Then, since x_i^{k+1}, $i = 1,\ldots,N$, must be within the interval $[x_{min},x_{max}]$, due to the hard constraints on the state variables, we set $\mathbf{x}^{k+1} = \text{Proj}_{\bar{\mathcal{X}}}(\bar{\mathbf{x}})$ where the operator Proj projects each component of $\bar{\mathbf{x}}$ into the interval $[x_{min},x_{max}]$. Now, observe that (12) can be rewritten component-wise as

$$\lambda_h^{k+1} = \max\left\{0,\lambda_h^k + c\,m_h^k\right\}, \tag{14}$$

where

$$m_h^k = \sum_{j\,:\,v_j\in\mathcal{N}_{s_h}} B_{hj}x_j^k - d_h. \tag{15}$$

Observe that node s_h, to locally update the variable λ_h, requires state information of the nodes within $\mathcal{N}_{s_h}$. In addition observe that

$$x_i^{k+1} = \text{Proj}_{[x_{\min};x_{\max}]}(\bar{x}_i) \tag{16}$$

where

$$\bar{x}_i = x_i^k - \alpha\left[\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\right]_i \tag{17}$$

with $\left[\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\right]_i = a_i + \sum_{h\,:\,s_h\in\mathcal{N}_{v_i}} B_{hi}\max\left\{0,\lambda_h^{k+1} + c\,m_h^k\right\}$.

It follows that, node v_i in order to compute the i-th component of $\bar{x}$, i.e., $\bar{x}_i$, needs the all the nodes s_h within $\mathcal{N}_{v_i}$ transmit to it the updated values of the Lagrange multiplier λ_h and of the quantity m_h.

VI. An ADMM-based algorithm

Alternating direction multipliers method (in short ADMM) is a very popular algorithm used to solve optimization problem of the form:

$$\begin{aligned}\min_{x\in\mathcal{X},y\in\mathcal{Y}} \quad & f(x) + g(y)\\ \text{s.t.} \quad & Ax + By = c\end{aligned}$$

ADMM is applied in several applications (see for instance [5] and reference therein). In the following we manipulate problem in (1) in such a way to implement an ADMM-based algorithm to solve it. Observe that, by introducing the slack variable $\mathbf{y}\in\mathbb{R}^M$, problem in (1) can be reformulated into the equivalent problem:

$$\begin{aligned}\min_P \quad & \mathbf{a}^T\mathbf{x}\\ s.t. \quad & B\mathbf{x} + \mathbf{y} = \mathbf{d}\\ & x_{min}\mathbf{1} \le \mathbf{x} \le x_{max}\mathbf{1}\\ & \mathbf{y} \ge 0\end{aligned} \tag{18}$$

Now let us introduce the augmented Lagrangian:

$$\mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda}) = \mathbf{a}^T\mathbf{x} + \boldsymbol{\lambda}^T(B\mathbf{x} + \mathbf{y} - \mathbf{d}) + \frac{\rho}{2}\|B\mathbf{x} + \mathbf{y} - \mathbf{d}\|^2.$$

The ADMM algorithm keeps alternating the following steps:

$$\mathbf{x}^{k+1} = \underset{x_{min}\mathbf{1}\leq\mathbf{x}\leq x_{max}\mathbf{1}}{\operatorname{argmin}} \mathcal{L}\left(\mathbf{x}, \mathbf{y}^k, \boldsymbol{\lambda}^k\right) \quad (19)$$

$$\mathbf{y}^{k+1} = \underset{\mathbf{y}\geq 0}{\operatorname{argmin}} \mathcal{L}\left(\mathbf{x}^{k+1}, \mathbf{y}, \boldsymbol{\lambda}^k\right) \quad (20)$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho\left(B\mathbf{x}^{k+1} + \mathbf{y}^{k+1} - \mathbf{d}\right). \quad (21)$$

Computing the first step in a distributed way is an hard problem. A reasonable approximated solution might be computed taking the derivative of $\mathcal{L}$ with the respect to $\mathbf{x}$, i.e., $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$, solving $\frac{\partial\mathcal{L}}{\partial\mathbf{x}} = 0$ w.r.t. $\mathbf{x}$ and projecting the obtained solution into the box constraints $x_{min}\mathbf{1} \leq \mathbf{x} \leq x_{max}\mathbf{1}$. Precisely

$$\frac{\partial\mathcal{L}}{\partial\mathbf{x}} = \mathbf{a} + B^T\boldsymbol{\lambda}^k + \rho\left(B^TB\mathbf{x} + B^T\mathbf{y}^k - B^T\mathbf{d}\right) = 0$$

from which we get

$$\bar{\mathbf{x}} = \left(B^TB\right)^{-1}\left(-\frac{1}{\rho}\mathbf{a} - \frac{1}{\rho}B^T\boldsymbol{\lambda}^k - B^T\mathbf{y}^k + B^T\mathbf{d}\right).$$

Observe that the matrix B is typically sparse and, in turn, also B^T; however the product B^TB might be, in general, a full matrix. It follows that, in order to compute $\bar{\mathbf{x}}$ with only one exchange of information, an all-to-all communication graph is required thus compromising a distributed implementation. In our paper we approximate the first step in an alternative way which works as follows. We first compute a gradient descent step of the form

$$\bar{\mathbf{x}} = \mathbf{x}^k - \alpha\left(\mathbf{a} + B^T\boldsymbol{\lambda}^k + \rho B^TB\mathbf{x}^k + \rho B^T\mathbf{y}^k - \rho B^T\mathbf{d}\right)$$

and, then, we project $\bar{\mathbf{x}}$ into the box constraints

$$\mathbf{x}^{k+1} = \text{Proj}_{[x_{min}\mathbf{1};x_{max}\mathbf{1}]}(\bar{\mathbf{x}}).$$

We perform similar steps for the update of the variable $\mathbf{y}$. Observing that

$$\frac{\partial\mathcal{L}}{\partial\mathbf{y}} = \boldsymbol{\lambda} + \rho\left(B\mathbf{x} + \mathbf{y} - \mathbf{d}\right),$$

we first compute $\bar{\mathbf{y}} = \mathbf{y}^k - \beta\left(\boldsymbol{\lambda}^k + \rho\left(B\mathbf{x}^{k+1} + \mathbf{y}^k - \mathbf{d}\right)\right)$ and then the projection step $\mathbf{y}^{k+1} = \text{Proj}_{[\mathbf{y}\geq 0]}(\bar{\mathbf{y}})$. The update of λ is performed as above reported.

We now discuss the distributed implementation of the ADMM-based algorithm above discuss. Again, let $m_h^k = \sum_{j\,:\,v_j\in\mathcal{N}_{s_h}} B_{hj}x_j^k - d_h$. As in the previous algorithms, the variable x_i is stored in memory by node v_i, while the quantities m_h, z_h, λ_h are stored by node s_h. Now, let us write the component-wise update of $\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}$. We have that

$$\bar{x}_i = x_i^k - \alpha\left(a_i + \rho\sum_{h\,:\,s_h\in\mathcal{N}_{v_i}} B_{hi}\left(\lambda_h^k + y_h^k + m_h^k\right)\right) \quad (22)$$

and, in turn,

$$x_i^{k+1} = \text{Proj}_{[x_{\min};x_{\max}]}(\bar{x}_i). \quad (23)$$

Now, let

$$m_h^{k+1} = \sum_{j\,:\,v_j\in\mathcal{N}_{s_h}} B_{hj}x_j^{k+1} - d_h. \quad (24)$$

Then, regarding the variable $\mathbf{y}$, we have that

$$\bar{y}_h = y_h^k - \beta\left(\lambda_h^k + \rho\left(m_h^{k+1} + y_h^k\right)\right) \quad (25)$$

and, in turn,

$$y_h^{k+1} = \max\left\{0, \bar{y}_h\right\}. \quad (26)$$

Finally, as far as the variable $\boldsymbol{\lambda}$ is concerned, we have that

$$\lambda_h^{k+1} = \lambda_h^k + \rho\left(m_h^{k+1} + y_h^{k+1}\right). \quad (27)$$

Summarizing the ADMM-based algorithm keeps alternating the following steps:

1) For $h = 1, \ldots, M$, s_h transmits the value of λ_h, m_h and y_h to all nodes in $\mathcal{N}_{s_h}$;
2) For $i = 1, \ldots, N$, v_i gathers all the quantities sent by the nodes in $\mathcal{N}_{v_i}$ and updates x_i as in (22) and (23);
3) For $i = 1, \ldots, N$, v_i transmits the updated value of x_i to all nodes s_h in $\mathcal{N}_{v_i}$;
4) For $h = 1, \ldots, M$ s_h gathers the values of the states sent by nodes in $\mathcal{N}_{s_h}$ and updates in order m_h as in (24), y_h as in (25) and (26), and λ_h as in (27).

Remark 1: Notice that, also in the ADMM-based scheme, we perform only one iteration of the projected gradient for both the updates of x_i and y_h, leading to values of x_i^{k+1} and y_h^{k+1} which are just approximations of the minimizers in (19) and (20). To the best of our knowledge there are no theoretical guarantees in literature for these approximated schemes. However, we might modify the algorithm to perform several iterations of the projected gradient steps between two consecutive updates of the Lagrange multipliers. Hopefully, this would allow nodes in V and nodes in $\mathcal{S}$ to compute a value of $\mathbf{x}^{k+1}$ and $\mathbf{y}^{k+1}$ which might be very close to the minimizers in (19) and (20), respectively. In this case, convergence results from in [5], would ensure theoretically the convergence of the proposed algorithm toward the optimal solution. However, in simulations, we perform only one iteration of the projected gradient algorithms in both the Multiplier Method and in the ADMM approach in order to provide a fair comparison in terms of communication requirements with the dual ascent algorithm.

VII. Numerical Simulations

In this section, we present a comparison of the aforementioned distributed Lagrangian-based optimization algorithms based on the numerical simulation results obtained from their MATLAB implementations.

Problem Generation: During simulations, we created 50×1 vector $\mathbf{a}$ and 150×50 matrix B by filling their entries using the random values from the interval $[-0.5, 0.5]$. We set several entries of the matrix A and B to zero so that we obtain a sparse matrices representing the communication network. Moreover, the entries of the 150×1 vector $\mathbf{d}$ and the initial 50×1 vector $\mathbf{x}$ are selected randomly from the interval $[0, 1]$. Finally, we set $x_{\min} = 0$ and assigned a random value to $x_{\max}$ from the interval $[0, 1]$. Therefore, we created a random instance of the linear programming problem described in (1).

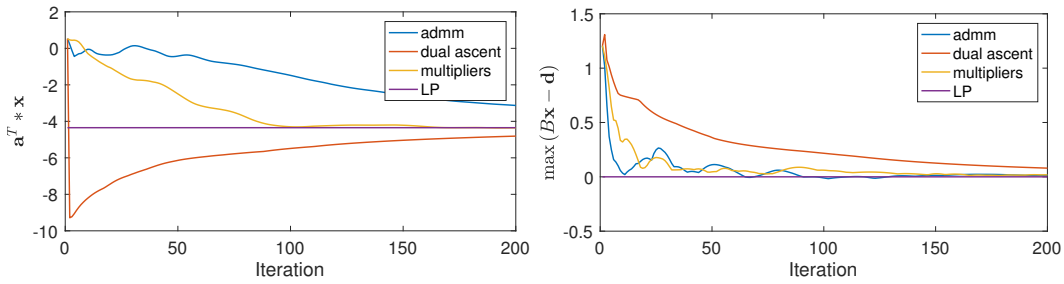

Fig. 2. The value of the objective function $\mathbf{a}^T\mathbf{x}$ (left) and the constraint satisfaction indicated by $\max(B\mathbf{x} - \mathbf{d})$ at each iteration.

TABLE I

THE SELECTED PARAMETER VALUES.

	θ	ρ	c	α	β
Dual Ascent	0.9	Use Prop.(1)	-	-	-
Multipliers	-	-	2	0.01	-
ADMM	-	2	-	0.01	0.01

Algorithm Parameters: In order to get a baseline for comparison, we used MATLAB's `linprog` function to obtain the optimal solution. For the dual-ascent like approach described in Section IV; we selected θ as 0.9 and ρ accordingly using Proposition (1). For the multipliers method described in Section V, we set $\alpha = 0.01$ and $c = 2$. Finally, for ADMM, we set $\alpha = 0.01$, $\beta = 0.01$ and $\rho = 2$. The initial entries of the $M \times 1$ vector λ in dual-ascent and ADMM-based approaches are set to 0. Moreover, the entries of the $M \times 1$ vector $\mathbf{y}$ in ADMM are initially selected randomly from the interval $[0, 1]$. All of the selected parameter values during simulations are summarized in Table I.

Observations: We performed 200 iterations using the proposed approaches. We would like to mention that we performed only one gradient descent iteration for each iteration of the multipliers and ADMM methods—providing a fair comparison under the same communication complexity. Figure 2 presents the value of the objective function $\mathbf{a}^T\mathbf{x}$ and the constraint satisfaction indicated by $\max(B\mathbf{x} - \mathbf{d})$ with the aforementioned approaches during a sample simulation. From our results, we observed that the multipliers method had faster convergence rate while the ADMM approach was the slowest among them. On the other hand, the constraints were satisfied slower with dual-ascent like approach as compared to ADMM and multipliers methods.

VIII. CONCLUSIONS

In this work we addressed the problem of partition-based solutions of a specific class of LP problems which involve box constraints and inequality constraints in the decision variables. We have shown that this specific optimization problem arises from a number of diverse applications in the context of smart multi-agent systems. We proposed three Lagrangian-based algorithms based on dual gradient ascent, multiplier methods and alternating direction multiplier methods, each showing pros and cons in terms of speed of convergence and communication/computational complexity. Future research avenues involve the theoretical analysis of convergence properties of the proposed MM and ADMM algorithm since their implementation does not fall into the traditional convergence assumptions, and the extension of all three algorithms in the more realistic setting with asynchronous updating and lossy communication.

REFERENCES

[1] Q. Liu, K. S. Yıldırım, P. Pawełczak, and M. Warnier, "Safe and secure wireless power transfer networks: challenges and opportunities in rf-based systems," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 74–79, September 2016.

[2] S. Borile, A. Pandharipande, D. Caicedo, L. Schenato, and A. Cenedese, "A data-driven daylight estimation approach to lighting control," *IEEE Access*, vol. 5, pp. 21 461 – 21 471, 2017.

[3] S. Baldi, S. Yuan, P. Endel, and O. Holub, "Dual estimation: Constructing building energy models from data sampled at low rate," *Applied Energy*, vol. 169, pp. 81–92, 2016.

[4] A. Nedic and A. Ozdaglar, *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2010, ch. Cooperative Distributed Multi-Agent Optimization, pp. 340–386.

[5] 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.

[6] R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, "Analysis of newton-raphson consensus for multi-agent convex optimization under asynchronous and lossy communications," in *IEEE 54th Annual Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 418–424.

[7] 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.

[8] M. Todescato, N. Bof, G. Cavraro, R. Carli, and L. Schenato, "Generalized gradient optimization over lossy networks for partition-based estimation," *arXiv preprint arXiv:1710.10829*, 2017.

[9] R. Carli and G. Notarstefano, "Distributed partition-based optimization via dual decomposition," in *IEEE Conference on Decision and Control*, Firenze, Italy, 2013.

[10] V. Kekatos and G. Giannakis, "Distributed robust power system state estimation," *Distributed robust power system state estimation*, vol. 28, no. 2, pp. 1617–1626, 2013.

[11] S. Gollakota, M. Reynolds, J. Smith, and D. Wetherall, "The emergence of RF-powered computing," *Computer*, vol. 47, no. 1, pp. 32–39, Jan. 2014.

[12] K. S. Yıldırım, R. Carli, and L. Schenato, "Distributed control of wireless power transfer subject to safety constraints," in *IFAC-PapersOnLine*, vol. 50, no. 1, 2017, pp. 13 210–13 215.

[13] K. S. Yıldırım, R. Carli, L. Schenato, and M. Todescato, "A distributed dual-ascent approach for power control of wireless power transfer networks," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 3507–3512.

[14] K. S. Yıldırım, R. Carli, and L. Schenato, "Safe distributed control of wireless power transfer networks," *The IEEE Internet of Things*, to appear.