Distributed Optimization With Local Domains: Applications in MPC and Network Flows

João F. C. Mota, João M. F. Xavier, Pedro M. Q. Aguiar, and Markus Püschel

***Abstract*—We consider a network where each node has exclusive access to a local cost function. Our contribution is a communication-efficient distributed algorithm that finds a vector $x^\star$ minimizing the sum of all the functions. We make the additional assumption that the functions have intersecting local domains, i.e., each function depends only on some components of the variable. Consequently, each node is interested in knowing only some components of $x^\star$, not the entire vector. This allows improving communication-efficiency. We apply our algorithm to distributed model predictive control (D-MPC) and to network flow problems and show, through experiments on large networks, that the proposed algorithm requires less communications to converge than prior state-of-the-art algorithms.**

***Index Terms*—Alternating direction method of multipliers, distributed algorithms, model predictive control, network flows.**

I. Introduction

Consider a network with P nodes and the following problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f_1(x)+f_2(x)+\cdots+f_P(x) \tag{1}$$

where each function f_p is known only at node p. We say an algorithm solving (1) is *distributed* if it uses no central node and no all-to-all communications. In a typical distributed algorithm, each node holds an estimate of a solution $x^\star$ and iteratively updates and exchanges it with its neighbors. Such an algorithm implicitly assumes that each node is interested in knowing all the components of a solution $x^\star$. While this holds for problems like consensus or distributed SVMs, there are important problems where it does not hold, especially in the context of large networks. Two examples we explore are distributed model predictive control (D-MPC) [1]–[3] and network flows [4].

We solve (1) assuming that function f_p depends only on a subset of the components of the variable $x\in\mathbb{R}^n$. This is illustrated in Fig. 1(a) with $n=3$; there, for example, f_1 depends on x_1 and x_2, but not on x_3. To capture these dependencies, we write x_S, $S\subseteq\{1,\ldots,n\}$, to denote a subset of the components of x. For example, if $S=\{2,4\}$, then $x_S=(x_2,x_4)$. With this notation, our goal is to solve

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f_1\left(x_{S_1}\right)+f_2\left(x_{S_2}\right)+\cdots+f_P\left(x_{S_P}\right) \tag{2}$$

Manuscript received May 9, 2013; revised January 28, 2014 and August 26, 2014; accepted October 26, 2014. Date of publication October 29, 2014; date of current version June 24, 2015. This work was supported by the grants from Fundação para a Ciência e TecnologiaCMU-PT/SIA/0026/2009, PEst-OE/EEI/LA0009/2011, and SFRH/BD/33520/2008 (Carnegie Mellon/Portugal Program, ICTI). Recommended by Associate Editor C. M. Lagoa.

J. F. C. Mota was with Institute of Systems and Robotics, Instituto Superior Técnico, Technical University of Lisbon, 1049-001 Lisbon, Portugal. He is now with the Electronic and Electrical Engineering Department, University College London, London WC1E 6BT, U.K. (e-mail: j.mota@ucl.ac.uk).

J. M. F. Xavier and P. M. Q. Aguiar are with Institute of Systems and Robotics, Instituto Superior Técnico, Technical University of Lisbon, 1049-001 Lisbon, Portugal (e-mail: jxavier@isr.ist.utl.pt; aguiar@isr.ist.utl.pt).

M. Püschel is with the Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland (e-mail: pueschel@inf.ethz.ch).

Digital Object Identifier 10.1109/TAC.2014.2365686

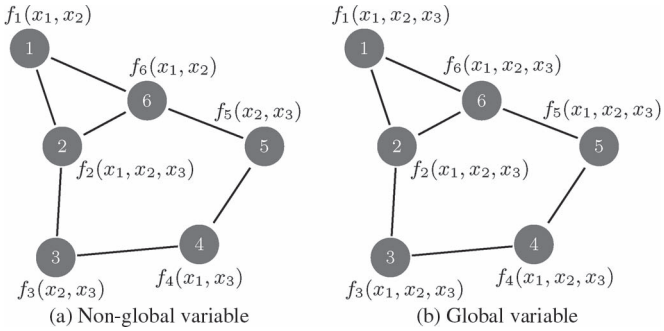

Fig. 1. Example of a (a) non-global variable and a (b) global variable, for a variable $x=(x_1,x_2,x_3)$ with 3 components. While in (a) only node 2 depends on all of them, in (b) all nodes depend on all the components.

where S_p is the set of components that the function f_p depends on. Accordingly, node p is interested in finding only the components of a solution $x^\star$ that are indexed by S_p, i.e., $x^\star_{S_p}$. Fig. 1 illustrates two instances in a network with $P=6$ nodes and a variable of size $n=3$. In Fig. 1(a), all nodes but node 2 depend on a strict subset of the components of the variable; in Fig. 1(b), all nodes depend on all the components. Thus, Fig. 1(b) is an instance of problem (1), which we say has a *global variable*. Problem (2) generalizes problem (1), since it becomes (1) when $S_p=\{1,\ldots,n\}$ for all nodes p. However, any algorithm designed to solve (1) can also solve (2) by making all nodes estimate all the components of a solution $x^\star$. This approach, however, introduces unnecessary communications, since nodes exchange more components than necessary. Our goal is to design distributed algorithms for (2) that use its structure to reduce the number of communications.

Contributions: We propose a distributed algorithm that solves (2) in full generality, for arbitrary sets S_p. This is done by classifying its variable into two categories and designing algorithms for both. We then apply our algorithms to D-MPC and network flow problems. We show that, with our general solution, we can outperform prior algorithms, even application-specific ones. Due to space constraints, we omit some proofs and a detailed derivation for the non-connected case. Both, however, can be found in [5].

Related Work: Many algorithms have been proposed for the global variable problem (1), including methods based on the *alternating direction method of multipliers* (ADMM) [6], [7]. However, solving (2) with an algorithm designed for (1) introduces unnecessary communications.

To our knowledge, this is the first time that problem (2) has been explicitly stated in a distributed context. For example, [8, Sec. 7.2] proposes an algorithm for (2), but is not distributed in our sense, since it requires either a platform that supports all-to-all communications (in other words, a central node) or running consensus algorithms on each induced subgraph at each iteration [8, Sec. 10.1]. Thus, that algorithm is distributed only when every component induces subgraphs that are stars, in which case we say the variable is *star-shaped*. Actually, we only found one algorithm in the literature, [9], that is distributed (or can easily be made distributed) for all the scenarios considered in

this paper. Yet, that algorithm was proposed for a problem with a star-shaped variable: power system state estimation (our algorithm can be applied to this problem as well). Our simulations show that the algorithm in [9] requires always more communications than the algorithm we propose. Although we found just one (communication-efficient) distributed algorithm solving (2), many other algorithms solve particular instances of it; see, e.g., [5], [10]. For example, in network flow problems, each component of the variable is associated to an edge of the network which, as we will see, enables writing them as (2) with a star-shaped variable. In this case, not only [8, Sec. 7.2] becomes distributed, but also gradient/subgradient methods can be applied directly and yield distributed algorithms [12]. Distributed Model Predictive Control (D-MPC) [1]–[3] is another problem that has been addressed with algorithms solving (2), again in the special case of a star-shaped variable. Such algorithms include fast gradient [13] and ADMM-based [13] methods (which applies [8, Sec. 7.2]). Akin to [8, Sec. 7.2], these methods were designed for the special case of star-shaped variables and become inefficient when applied to more generic cases. In spite of its generality, the algorithm we propose requires less communications than previous algorithms that were designed specifically for D-MPC or for network flow problems.

II. Terminology and Problem Statement

We first define *communication network* and *variable connectivity*:

Communication Network: A communication network is represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, P\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Two nodes communicate if there is an edge connecting them in $\mathcal{G}$. We assume:

Assumption 1: $\mathcal{G}$ is connected and its topology does not change over time; also, a coloring scheme $\mathcal{C}$ of $\mathcal{G}$ is available beforehand.

A coloring scheme $\mathcal{C}$ is a set of numbers, called colors, assigned to the nodes such that two neighbors never have the same color. Given its importance in TDMA, a widespread protocol for avoiding packet collisions, there is a large literature on coloring networks; [5, Sec. 3.1]. Our algorithm integrates naturally with TDMA, since both use coloring as a synchronization scheme: nodes work sequentially according to their colors, and nodes with the same color work in parallel. The difference is that TDMA uses a more restrictive coloring, as nodes within two hops cannot have the same color. Note that packet collision is often ignored in the design of distributed algorithms, as confirmed by the ubiquitous assumption that all nodes can communicate simultaneously. We associate with each node p a function $f_p : \mathbb{R}^{n_p} \to \mathbb{R} \cup \{+\infty\}$, where $1 \leq n_p \leq n$, and make the

Assumption 2: Each function f_p is closed, proper, and convex over $\mathbb{R}^{n_p}$, and is known only at node p. The neighbors of node p know the set of components S_p that f_p depends on.

Since we allow f_p to take infinite values, constraints can be imposed via indicator functions, i.e., functions that evaluate to $+\infty$ when the constraints are not satisfied, and to 0 otherwise.

Variable Connectivity: Although each function f_p is available only at node p, each component of the variable x may be associated with several nodes. Let x_l be a given component. The *subgraph induced by* x_l is $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l) \subseteq \mathcal{G}$, where $\mathcal{V}_l$ is the set of nodes whose functions depend on x_l, and an edge $(i, j) \in \mathcal{E}$ belongs to $\mathcal{E}_l$ if both i and j are in $\mathcal{V}_l$. For example, the subgraph induced by x_1 in Fig. 1(a) consists of $\mathcal{V}_1 = \{1, 2, 4, 6\}$ and $\mathcal{E}_1 = \{(1, 2), (1, 6), (2, 6)\}$. We say x_l is *connected* if its induced subgraph is connected, and *non-connected* otherwise. Likewise, a *variable is connected* if all its components are connected, and *non-connected* if it has at least one non-connected component. In Fig. 1(a), the variable is non-connected, because x_1 induces a non-connected subgraph.

Problem Statement: Given a network and a set of functions satisfying Assumptions 1 and 2, we *design a distributed, communication-efficient algorithm that solves* (2), *with either a connected or a non-connected variable*. Recall that a distributed algorithm uses neither a central node nor all-to-all communications.

III. Connected Case

In this section we derive a distributed algorithm for (2) assuming a connected variable. The main idea is to manipulate (2) to make the Extended ADMM (E-ADMM) [14] applicable. Our algorithm generalizes [7], which proposed an algorithm for (1).

Problem Manipulation: Let x_l be a given component and $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ be the respective induced subgraph, assumed connected. Since all nodes in $\mathcal{V}_l$ are interested in x_l, we create copies of x_l in each of those nodes: $x_l^{(p)}$ is the copy at node p, and $x_{S_p}^{(p)} := \{x_l^{(p)}\}_{l \in S_p}$ denotes all copies at node p. We then rewrite (2) as

$$\begin{aligned} &\underset{\{\bar{x}_l\}_{l=1}^n}{\text{minimize}} \; f_1\left(x_{S_1}^{(1)}\right) + f_2\left(x_{S_2}^{(2)}\right) + \cdots + f_P\left(x_{S_P}^{(P)}\right) \\ &\text{subject to } x_l^{(i)} = x_l^{(j)}, \quad (i, j) \in \mathcal{E}_l, \; l = 1, \dots, n \end{aligned} \tag{3}$$

where the optimization variable $\{\bar{x}_l\}_{l=1}^n$ is the set of all copies. We used $\bar{x}_l := \{x_l^{(p)}\}_{p \in \mathcal{V}_l}$ to denote all copies of x_l, which are located only in the nodes of $\mathcal{G}_l$. The constraints in (3) enforce equality among the copies of the same component: if two neighboring nodes i and j depend on x_l, then $x_l^{(i)} = x_l^{(j)}$ appears in the constraints of (3). We assume that any edge in the communication network is represented as the ordered pair $(i, j) \in \mathcal{E}$, with $i < j$. As such, there are no repeated equations in (3). Problems (2) and (3) are equivalent because each induced subgraph is connected. We observe that $x_l^{(i)} = x_l^{(j)}$, $(i, j) \in \mathcal{E}_l$, can be written as $A_l \bar{x}_l = 0$, where A_l is the transposed node-arc incidence matrix of the subgraph $\mathcal{G}_l$. The node-arc incidence matrix associates each edge of a graph with a column of a matrix. The column associated with the edge (i, j) has 1 in the ith entry, -1 in the jth entry, and zeros elsewhere. We next partition the optimization variable according to the coloring scheme: for each $l = 1, \dots, n$, $\bar{x}_l = (\bar{x}_l^1, \dots, \bar{x}_l^C)$, where $\bar{x}_l^c = \{x_l^{(p)}\}_{p \in \mathcal{V}_l \cap \mathcal{C}_c}$ if $\mathcal{V}_l \cap \mathcal{C}_c \neq \emptyset$, and $\bar{x}_l^c = \emptyset$ if $\mathcal{V}_l \cap \mathcal{C}_c = \emptyset$. Also, $\mathcal{C}_c$ is the set of nodes that have color c. Thus, $\bar{x}_l^c$ is the set of copies of x_l held by the nodes that have color c. If no node with color c depends on x_l, then $\bar{x}_l^c$ is empty. A similar notation for the columns of the matrix A_l enables writing $A_l \bar{x}_l$ as $\bar{A}_l^1 \bar{x}_l^1 + \cdots + \bar{A}_l^C \bar{x}_l^C$, and thus (3) equivalently as

$$\begin{aligned} &\underset{\bar{x}^1, \dots, \bar{x}^C}{\text{minimize}} \; \sum_{p \in \mathcal{C}_1} f_p\left(x_{S_p}^{(p)}\right) + \cdots + \sum_{p \in \mathcal{C}_C} f_p\left(x_{S_p}^{(p)}\right) \\ &\text{subject to } \bar{A}^1 \bar{x}^1 + \cdots + \bar{A}^C \bar{x}^C = 0 \end{aligned} \tag{4}$$

where $\bar{x}^c = \{\bar{x}_l^c\}_{l=1}^n$, and $\bar{A}^c := \operatorname{diag}(\bar{A}_1^c, \bar{A}_2^c, \dots, \bar{A}_n^c)$ is the diagonal concatenation of the matrices $\bar{A}_1^c, \bar{A}_2^c, \dots, \bar{A}_n^c$. The format of (4) is exactly the one to which E-ADMM applies, as explained next.

E-ADMM: The Extended ADMM (E-ADMM) is a natural generalization of the *Alternating Direction Method of Multipliers* (ADMM). Given a set of closed, convex functions $g_1, \dots, g_C$, and a set of full column rank matrices $E_1, \dots, E_C$, E-ADMM solves

$$\begin{aligned} &\underset{x_1, \dots, x_C}{\text{minimize}} \; g_1(x_1) + \cdots + g_C(x_C) \\ &\text{subject to } E_1 x_1 + \cdots + E_C x_C = 0. \end{aligned} \tag{5}$$

It consists of iterating on k the following equations:

$$x_1^{k+1} = \arg\min_{x_1} \; L_\rho\left(x_1, x_2^k, \ldots, x_C^k; \lambda^k\right) \tag{6}$$

$$\vdots$$

$$x_C^{k+1} = \arg\min_{x_C} \; L_\rho\left(x_1^{k+1}, x_2^{k+1}, \ldots, x_{C-1}^{k+1}, x_C; \lambda^k\right) \tag{7}$$

$$\lambda^{k+1} = \lambda^k + \rho \sum_{c=1}^{C} E_c x_c^{k+1} \tag{8}$$

where $L_\rho(x;\lambda) = \sum_{c=1}^{C}(g_c(x_c) + \lambda^\top E_c x_c) + (\rho/2)\| \sum_{c=1}^{C} E_c x_c \|^2$ is the augmented Lagrangian of (5), λ is the dual variable, and $\rho > 0$. The original ADMM is recovered whenever $C = 2$, i.e., when there are only two terms in the sums of (5). The following theorem gathers some known convergence results for (6)–(8).

Theorem 1 [14]–[17]: For each $c = 1, \ldots, C$, let $g_c : \mathbb{R}^{n_c} \to \mathbb{R} \cup \{+\infty\}$ be closed and convex over $\mathbb{R}^{n_c}$ and $\operatorname{dom} g_c \neq \emptyset$. Let each E_c be an $m \times n_c$ matrix. Assume (5) is solvable and that either 1) $C = 2$ and each E_c has full column rank, or 2) $C \geq 2$ and each g_c is strongly convex. Then, the sequence $\{(x_1^k, \ldots, x_C^k, \lambda^k)\}$ generated by (6)–(8) converges to a primal-dual solution of (5). Furthermore, if each function g_c is strongly convex, differentiable, and its gradient is Lipschitz-continuous, then linear convergence holds whenever $C = 2$, or $C > 2$ and ρ in (8) is replaced by a small constant.

It is believed that (6)–(8) converges even when $C > 2$, each g_c is closed and convex (not necessarily strongly convex), each matrix E_c has full column rank, and is sufficiently orthogonal to the other matrices [18]. Such belief is supported by empirical evidence [5], [14] but proving it remains an open problem.

Applying E-ADMM: The clear correspondence between (4) and (5) makes (6)–(8) directly applicable to (4). Associate a dual variable λ_l^{ij} to each constraint $x_l^{(i)} = x_l^{(j)}$ in (3). Translating (8) component-wise, λ_l^{ij} is updated, for a given $(i,j) \in \mathcal{E}$, as

$$\lambda_l^{ij,k+1} = \lambda_l^{ij,k} + \rho\left(x_l^{(i),k+1} - x_l^{(j),k+1}\right) \tag{9}$$

where $x_l^{(p),k+1}$ is the estimate of x_l at node p after iteration k. This estimate is obtained from the sequence (6)–(7), where we focus our attention now. This sequence yields the synchronization mentioned in Section II: nodes work sequentially according to their colors, with the same-colored nodes working in parallel. In fact, each problem in (6)–(8) corresponds to one color. Moreover, each of these problems decomposes into $|\mathcal{C}_c|$ problems that can be solved in parallel, each by a node with color c. For example, the copies of the nodes with color 1 are updated by solving (6):

$$\min_{\bar{x}^1} \quad \sum_{p\in\mathcal{C}_1} f_p\left(x_{S_p}^{(p)}\right) + {\lambda^k}^\top \bar{A}^1 \bar{x}^1 + \frac{\rho}{2}\left\| \bar{A}^1 \bar{x}^1 + \sum_{c=2}^{C} \bar{A}^c \bar{x}^{c,k} \right\|^2 \tag{10}$$

or, equivalently (see [5, Lemma 4.6])

$$\min_{\bar{x}^1} \quad \sum_{p\in\mathcal{C}_1} f_p\left(x_{S_p}^{(p)}\right) + \sum_{l\in S_p}\left[\sum_{j\in\mathcal{N}_p\cap\mathcal{V}_l} {\eta_l^{pj,k}}^\top x_l^{(p)} + \frac{\rho D_{p,l}}{2}\left(x_l^{(p)}\right)^2 \right] \tag{11}$$

where $\eta_l^{pj,k} := \operatorname{sg}(j-p)\lambda_l^{pj,k} - \rho\, x_l^{(j),k}$, and $\operatorname{sg}(\cdot)$ is the sign function, defined as $\operatorname{sg}(a) = 1$ if $a \geq 0$, and $\operatorname{sg}(a) = -1$ if $a < 0$. Also, $D_{p,l}$ is the degree of node p in the subgraph $\mathcal{G}_l$, i.e., the number of neighbors of node p that also depend on x_l. Note that (11) decomposes into $|\mathcal{C}_1|$ problems that can be solved in parallel. This is because $\bar{x}^1$ consists of the copies held by the nodes with color 1; and, since same-colored nodes are never neighbors, none of the copies in $\bar{x}^1$ appears as $x_l^{(j),k}$, through $\eta_l^{pj,k}$, in the second term of (11). Therefore, all nodes p in $\mathcal{C}_1$ can solve in parallel

$$\min_{x_{S_p}^{(p)}} \quad f_p\left(x_{S_p}^{(p)}\right) + \sum_{l\in S_p}\left[\sum_{j\in\mathcal{N}_p\cap\mathcal{V}_l} {\eta_l^{pj,k}}^\top x_l^{(p)} + \frac{\rho D_{p,l}}{2}\left(x_l^{(p)}\right)^2 \right] \tag{12}$$

where $x_{S_p}^{(p)} = \{x_l^{(p)}\}_{l\in S_p}$. However, node p can solve (12) only if it knows $\eta_l^{pj,k}$, i.e., $x_l^{(j),k}$ and $\lambda_l^{pj,k}$, for $j \in \mathcal{N}_p \cap \mathcal{V}_l$ and $l \in S_p$. This is possible if, in the previous iteration, it received the respective copies of x_l from its neighbors. This also allows knowing the dual variable $\lambda_l^{pj,k}$, although we will see later that no node needs to know each $\lambda_l^{pj,k}$ individually. For the other colors, the analysis is similar, except that in the definition of $\eta_l^{pj,k}$, we have $x_l^{(j),k+1}$ (resp. $x_l^{(j),k}$) if the color of node j is smaller (resp. larger) than the color of node p.

Algorithm 1 Algorithm for a Connected Variable

Initialization: for all $p \in \mathcal{V}$, $l \in S_p$, set $\gamma_l^{(p),1} = x_l^{(p),1} = 0$; $k = 1$

1: repeat
2: **for** $c = 1, \ldots, C$ **do**
3: **for all** $p \in \mathcal{C}_c$ [in parallel] **do**
4: **for all** $l \in S_p$ **do**

$$v_l^{(p),k} = \gamma_l^{(p),k} - \rho \sum_{\substack{j\in\mathcal{N}_p\cap\mathcal{V}_l \\ C(j)<c}} x_l^{(j),k+1} - \rho \sum_{\substack{j\in\mathcal{N}_p\cap\mathcal{V}_l \\ C(j)>c}} x_l^{(j),k}$$

5: **end for**
6: Set $x_{S_p}^{(p),k+1}$ as the solution of

$$\arg\min_{x_{S_p}^{(p)} = \left\{x_l^{(p)}\right\}_{l\in S_p}} f_p\left(x_{S_p}^{(p)}\right) + \sum_{l\in S_p} {v_l^{(p),k}}^\top x_l^{(p)} + \frac{\rho}{2}\sum_{l\in S_p} D_{p,l}\left(x_l^{(p)}\right)^2$$

7: For each component $l \in S_p$, send $x_l^{(p),k+1}$ to $\mathcal{N}_p \cap \mathcal{V}_l$
8: **end for**
9: **end for**
10: **for all** $p \in \mathcal{V}$ and $l \in S_p$ [in parallel] **do**

$$\gamma_l^{(p),k+1} = \gamma_l^{(p),k} + \rho \sum_{j\in\mathcal{N}_p\cap\mathcal{V}_l} \left(x_l^{(p),k+1} - x_l^{(j),k+1}\right)$$

11: **end for**
12: $k \leftarrow k + 1$
13: **until** some stopping criterion is met

The resulting algorithm is shown in Alg.1, whose structure matches equations (6)–(8): steps 2–9 correspond to (6)–(7), and the loop in step 10 corresponds to (8). In steps 2–9, nodes work according to their colors, with the same-colored nodes working in parallel. Each node computes the vector v in step 4, solves the optimization problem in step 6, and then sends the new estimates of x_l to the neighbors that also depend on x_l, for $l \in S_p$. We introduced new notation in step 4: $C(p)$ is the color of node p. The optimization problem in step 6 involves the private function of node p, f_p, to which a quadratic term is added. Finally, note that the update of the dual variables in step 10 differs from (9). In particular, all the λ's at node p were condensed into a single dual variable $\gamma^{(p)}$, since the optimization problem (12) does not depend on the individual λ_l^{pj}'s, but only on $\gamma_l^{(p),k} := \sum_{j\in\mathcal{N}_p\cap\mathcal{V}_l} \operatorname{sg}(j-p)\lambda_l^{pj,k}$. If we replace $\lambda_l^{ij,k+1} = \lambda_l^{ij,k} + \rho \operatorname{sg}(j-i)(x_l^{(i),k+1} - x_l^{(j),k+1})$ in the definition of $\gamma_l^{(p),k}$, we obtain the update of step 10. The extra "sign" in the previous expression (w.r.t. (9)) is necessary to take into account the extension of the definition of the dual variable λ_l^{ij} for $i > j$ (see Lemma 4.6 in [5]).

Convergence: Algorithm 1 results from the application of E-ADMM to problem (4). Consequently, the conclusions of Theorem 1 apply if we prove that (4) satisfies the conditions of that theorem.

Lemma 1: Each matrix $\bar{A}^c$ in (4) has full column rank.

Proof: Let c be any color. Since $\bar{A}^c = \mathrm{diag}(\bar{A}_1^c, \bar{A}_2^c, \ldots, \bar{A}_n^c)$, we have to prove that each $\bar{A}_l^c$ has full column rank. Fix c and l. If we prove that $(\bar{A}_l^c)^\top \bar{A}_l^c$ has full rank, then $\bar{A}_l^c$ has full column rank. Since $\bar{A}_l = [\bar{A}_1^c \ \bar{A}_2^c \ \cdots \ \bar{A}_n^c]$, $(\bar{A}_l^c)^\top \bar{A}_l^c$ corresponds to the lth block in the diagonal of $A_l^\top A_l$, the Laplacian of the subgraph $\mathcal{G}_l$. By assumption, $\mathcal{G}_l$ is connected, implying that each node in $\mathcal{G}_l$ has at least one neighbor also in $\mathcal{G}_l$; hence, each entry in the diagonal of $A_l^\top A_l$ is greater than zero. The same happens to the entries in the diagonal of $(\bar{A}_l^c)^\top \bar{A}_l^c$, which is a diagonal matrix. This is because $(\bar{A}_l^c)^\top \bar{A}_l^c$ corresponds to the Laplacian entries of nodes that have the same color, which are never neighbors. Thus, $(\bar{A}_l^c)^\top \bar{A}_l^c$ has full rank. ■

The following result is a consequence of Theorem 1 and Lemma 1.

Algorithm 2 Preprocessing Step for a non-Connected Variable

1: Set $S'_p = \emptyset$ for all $p \in \mathcal{V}$, and $\mathcal{V}'_l = \mathcal{V}_l$ for all $l = \{1, \ldots, n\}$
2: **for all** $l \in \{1, \ldots, n\}$ such that x_l is non-connected **do**
3: Compute a Steiner tree $(\mathcal{T}_l, \mathcal{F}_l)$ with $\mathcal{V}_l$ as required nodes
4: Set $\mathcal{V}'_l = \mathcal{T}_l$ and $\mathcal{S}_l := \mathcal{T}_l \setminus \mathcal{V}_l$ (Steiner nodes)
5: For all $p \in \mathcal{S}_l$, $S'_p \leftarrow S'_p \cup \{x_l\}$
6: **end for**

Corollary 1: Let Assumptions 1 and 2 hold and let the variable be connected. Let also one of the following conditions hold:

1) the network is bipartite, i.e., $C = 2$, or
2) each $\sum_{p \in \mathcal{C}_c} f_p(x_{S_p})$ is strongly convex, $c = 1, \ldots, C$.

Then, the sequence $\{x_{S_p}^{(p),k}\}_{k=1}^\infty$ at node p, produced by Alg.1, converges to $x_{S_p}^\star$, where $x^\star$ solves (2). Furthermore, if each $\sum_{p \in \mathcal{C}_c} f_p(x_{S_p})$ is strongly convex, differentiable, and has a Lipschitz-continuous gradient, then linear convergence holds in case 1); and it holds in case 2) whenever ρ in step 10 of Alg. 1 is replaced by a small constant.

As stated before, it is believed that E-ADMM converges for $C > 2$ even when none of the g_c's are strongly convex, just closed and convex; additionally, each E_c should have full column rank. This translates into the belief that Alg. 1 converges for any network, provided each f_p is closed and convex, and each matrix $\bar{A}^c$ has full column rank. The last condition is the content of Lemma 1.

Algorithm 1 is a generalization of D-ADMM [7], which assumes a global variable. Indeed, making $S_p = \{1, \ldots, n\}$ for all p, Alg. 1 becomes exactly D-ADMM. Note that, on the other hand, Alg. 1 cannot be obtained from D-ADMM. Each iteration of Alg. 1 (resp. D-ADMM) involves communicating $\sum_{p=1}^P |S_p|$ (resp. nP) numbers. When the variable is not global, $\sum_{p=1}^P |S_p| < nP$, and thus there is a clear per-iteration gain in solving (2) with Alg. 1.

IV. Non-Connected Case

When the variable is non-connected, problems (2) and (3) are no longer equivalent and, therefore, the previous derivations do not apply. We propose a small trick to make these problems equivalent.

Let x_l be a component whose induced subgraph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ is non-connected. Then, the constraint $x_l^{(i)} = x_l^{(j)}$, $(i,j) \in \mathcal{E}_l$, in (3) fails to enforce equality on all the copies of x_l. We propose replacing $\mathcal{G}_l$ with a connected subgraph $\mathcal{G}'_l \supset \mathcal{G}_l$, obtained by adding nodes and edges to $\mathcal{G}_l$. Since we seek to minimize communications, we want to add a minimal number of edges. This is exactly the *optimal Steiner tree* problem. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and let $\mathcal{R} \subseteq \mathcal{V}$ be a set of *required nodes*. A Steiner tree is any tree $(\mathcal{T}, \mathcal{F}) \subseteq \mathcal{G}$ that contains the required nodes: $\mathcal{R} \subseteq \mathcal{T}$. The set of nodes in the tree that are not required, $\mathcal{S} := \mathcal{T} \backslash \mathcal{R}$, are called *Steiner nodes*. In our case, the set of required nodes is $\mathcal{R} = \mathcal{V}_l$ and an *optimal* Steiner tree has a minimal number of edges. Although computing optimal Steiner trees is NP-hard, many approximation algorithms are available [19], even distributed ones [21]. Note that the solution returned by our algorithm is independent of the optimality of the Steiner tree.

We propose computing a (not necessarily optimal) Steiner tree for each induced subgraph $\mathcal{G}_l$ that is non-connected, as in Alg. 2. The Steiner tree for component x_l is $(\mathcal{T}_l, \mathcal{F}_l)$, where $\mathcal{V}_l \subseteq \mathcal{T}_l$ are the required nodes. The algorithm also computes a set S'_p containing the variables for which node p is Steiner. If node p is not Steiner for any variable, then $S'_p = \emptyset$. By defining new induced subgraphs as $\mathcal{G}'_l = (\mathcal{V}'_l, \mathcal{E}'_l)$, with $\mathcal{V}'_l := \mathcal{T}_l$ and $\mathcal{E}'_l := \mathcal{E}_l \cup \mathcal{F}_l$, we create copies of x_l in all nodes in $\mathcal{V}'_l$ and replace $\mathcal{E}$ in the constraints of (3) with $\mathcal{E}'$. The resulting problem is equivalent to (2). Algorithm 1 can then be applied with minor modifications: replace every instance of $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ with $\mathcal{G}'_l = (\mathcal{V}'_l, \mathcal{E}'_l)$, and every instance of S_p with $S_p \cup S'_p$ (see [5, Sec. 4.3.2] for details). Alg. 2 can be centralized or distributed (using, e.g., [21] to compute Steiner trees) and is meant to be executed once, before solving the problem (or a batch). It requires knowledge of the communication network $\mathcal{G}$ and the sets S_p, but not necessarily the functions f_p. This preprocessing step can also be applied to the algorithm in [9].

V. Applications

We now write distributed model predictive control (D-MPC) and network flow problems as (2) and solve them with Alg. 1.

D-MPC: In model predictive control (MPC), a system is described at each time instant t by its state-space $x[t] \in \mathbb{R}^n$. This state evolves as $x[t+1] = \Theta^t(x[t], u[t])$, where $u[t] \in \mathbb{R}^m$ is a control input and $\Theta^t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a linear function that models the system dynamics at time t. Given a time-horizon T, MPC consists of measuring the state at time $t = 0$, computing optimal states and inputs for the next T time steps, applying $u[0]$, setting $t = 0$, and repeating the process. The step of computing optimal states and inputs typically requires solving

$$\begin{aligned} \underset{\bar{x}, \bar{u}}{\text{minimize}} \ & \Phi\left(x[T]\right) + \sum_{t=0}^{T-1} \Psi^t\left(x[t], u[t]\right) \\ \text{subject to } & x[t+1] = \Theta^t\left(x[t], u[t]\right), \quad t = 0, \ldots, T-1 \\ & x[0] = x^0 \end{aligned} \tag{13}$$

where $(\bar{x}, \bar{u}) := (\{x[t]\}_{t=0}^T, \{u[t]\}_{t=0}^{T-1})$. While $\Phi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ penalizes deviations of the final state $x[T]$ from our goal, $\Psi^t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ measures energy consumption at time t. Since the functions Φ and Ψ^t can be $+\infty$, we can constrain, through indicator functions, the state and control input at any time instant to lie in a closed convex set. The first constraint of (13) enforces the system dynamics, and the second one encodes the initial measurement x^0.

We solve (13) in a distributed scenario: there is a (communication) network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $P = |\mathcal{V}|$ systems, where each system has a state $x_p[t] \in \mathbb{R}^{n_p}$ and a local input $u_p[t] \in \mathbb{R}^{m_p}$, with $n_1 + \cdots + n_P = n$ and $m_1 + \cdots + m_P = m$. The state of system p evolves as $x_p[t+1] = \Theta_p^t(\{x_j[t], u_j[t]\}_{j \in \Omega_p})$, where $\Omega_p \subseteq \mathcal{V}$ is the set of nodes whose state and/or input influences x_p (we assume $\{p\} \subseteq \Omega_p$ for all p). We assume Ω_p is not necessarily a subset of the neighbors of node p. In other words, two systems that influence each other may be unable to communicate directly. This is illustrated in Fig. 2(b) where, for example, the state/input of node 3 influences the state evolution of node 1 (dotted arrow), but there is no communication link (solid line) between them. Finally, we assume functions Φ and Ψ^t in (13) can be decomposed, respectively, as $\Phi(x[T]) = \sum_{p=1}^P \Phi_p(\{x_j[T]\}_{j \in \Omega_p})$

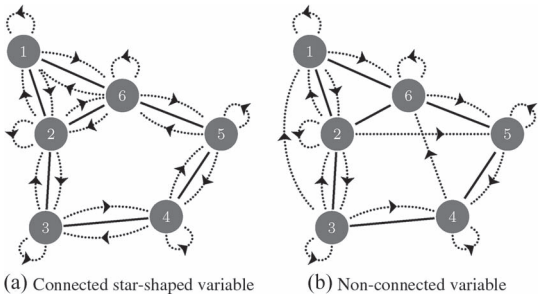

Fig. 2. Two D-MPC scenarios. Solid lines are links in the communication network, and dotted arrows represent system interactions. (a) Connected variable where each induced subgraph is a star. (b) Non-connected variable.

and $\Psi^t(x[t],u[t]) = \sum_{p=1}^{P} \Psi_p^t(\{x_j[t],u_j[t]\}_{j\in\Omega_p})$, where Φ_p and Ψ_p^t are both associated to node p. In sum, we solve

$$\begin{aligned} \min_{\bar{x},\bar{u}} \ & \sum_{p=1}^{P}\left[\Phi_p\Big(\{x_j[T]\}_{j\in\Omega_p}\Big) + \sum_{t=0}^{T-1}\Psi_p^t\Big(\{x_j[t],u_j[t]\}_{j\in\Omega_p}\Big)\right] \\ \text{s.t.} \ & x_p[t+1] = \Theta_p^t\left(\{x_j[t],u_j[t]\}_{j\in\Omega_p}\right), t=0,\dots,T-1 \\ & x_p[0] = x_p^0 \quad p=1,\dots,P \end{aligned} \tag{14}$$

where x_p^0 is the initial measurement at node p. The variable in (14) is $(\bar{x},\bar{u}) := (\{\bar{x}_p\}_{p=1}^P, \{\bar{u}_p\}_{p=1}^P)$, where $\bar{x}_p := \{x_p[t]\}_{t=0}^T$ and $\bar{u}_p := \{u_p[t]\}_{t=0}^{T-1}$. Problem (14) can be written as (2) by making $f_p(\{\bar{x}_j,\bar{u}_j\}_{j\in\Omega_p}) = \Phi_p(\{x_j[T]\}_{j\in\Omega_p}) + \mathrm{i}_{x_p[0]=x_p^0}(\bar{x}_p) + \sum_{t=0}^{T-1}(\Psi_p^t(\{x_j[t],u_j[t]\}_{j\in\Omega_p}) + \mathrm{i}_{\Gamma_p^t}(\{\bar{x}_j,\bar{u}_j\}_{j\in\Omega_p}))$, where $\mathrm{i}_{\Gamma_p^t}(\cdot)$ is the indicator function of the set Γ_p^t, and $\Gamma_p^t := \{\{\bar{x}_j,\bar{u}_j\}_{j\in\Omega_p} : x_p[t+1] = \Theta_p^t(\{x_j[t],u_j[t]\}_{j\in\Omega_p})\}$.

Fig. 2(a) illustrates the case where $\Omega_p \subseteq \mathcal{N}_p \cup \{p\}$, i.e., the state of node p is influenced by its own state/input and by the states/inputs of the systems it communicates directly with. This corresponds to a star-shaped variable where the center of the star is node p, whose state is x_p. Particular cases of this model have been considered, for example, in [1], whose solutions are heuristics, and in [13], whose solutions are optimization-based. The model we propose is more general, since it handles scenarios where interacting nodes do not need to communicate directly, or even scenarios with non-connected variables. Both cases are shown in Fig. 2(b). For example, the subgraph induced by $(\bar{x}_3,\bar{u}_3)$ contains nodes {1, 2, 3, 4} and is connected. (Connectivity refers always to the communication network, represented by solid lines in the plots.) Nodes 1 and 3, however, do not communicate directly. This is an example of an induced subgraph that is not a star. On the other hand, the subgraph induced by $(\bar{x}_2,\bar{u}_2)$ contains nodes {1, 2, 3, 5} and is not connected, implying a non-connected variable.

Network Flows: A network flow problem is typically formulated on a network with arcs (directed edges), where an arc from node i to node j indicates a flow in that direction [4]. In the example given in Fig. 3, there can be a flow from node 1 to node 5, but not the opposite. Every arc $(i,j) \in \mathcal{A}$ has associated a non-negative variable x_{ij} representing the amount of flow in that arc (from i to j), and a cost function $\phi_{ij}(x_{ij})$ that depends only on x_{ij}. The goal is to minimize the sum of all costs, while satisfying conservation of flow. External flow can be injected or extracted from a node, making that node a source or a sink, respectively. We represent the network of flows with the node-arc incidence matrix B, where the column associated to an arc from node i to node j has a 1 in the ith entry, a -1 in the jth entry, and zeros elsewhere. We assume the components of the variable x and the columns of B are in lexicographic order: e.g., the variable in Fig. 3 is $x = (x_{12}, x_{15}, x_{23}, x_{34}, x_{35}, x_{46}, x_{56})$. Conservation of flow is expressed as $Bx = d$, where $d \in \mathbb{R}^P$ is the vector of external inputs/outputs. The entries of d sum up to zero and $d_p > 0$ (resp. $d_p < 0$) if node p is a source (resp. sink). When node p is neither a source nor a sink, $d_p = 0$. We solve

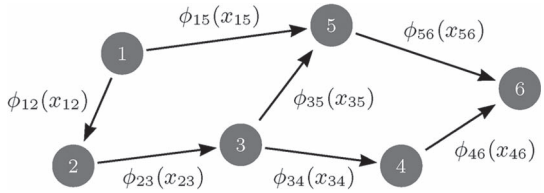

Fig. 3. Network flow problem: each edge has a variable x_{ij} representing the flow from node i to node j and also has a cost function $\phi_{ij}(x_{ij})$.

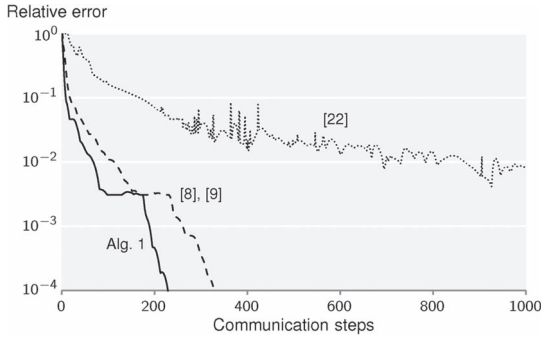

Fig. 4. Results for the network flow problem (15).

$$\underset{x}{\text{minimize}} \sum_{(i,j)\in\mathcal{A}} \phi_{ij}(x_{ij}) \quad \text{s.t.} \quad Bx = d,\ x \geq 0 \tag{15}$$

which is written as (2) with $f_p(\{x_{pj}\}_{(p,j)\in\mathcal{A}}, \{x_{jp}\}_{(j,p)\in\mathcal{A}}) = (1/2)\sum_{(p,j)\in\mathcal{A}} \phi_{pj}(x_{pj}) + (1/2)\sum_{(j,p)\in\mathcal{A}} \phi_{jp}(x_{jp}) + \mathrm{i}_{b_p^\top x = d_p}(\{x_{pj}\}_{(p,j)\in\mathcal{A}}, \{x_{jp}\}_{(j,p)\in\mathcal{A}})$, where $b_p^\top$ is the pth row of B. In words, f_p consists of the sum of the functions associated to all arcs involving node p, plus the indicator function of the set $\{x : b_p^\top x = d_p\}$. This indicator function enforces the conservation of flow at node p and it only involves the variables $\{x_{pj}\}_{(p,j)\in\mathcal{A}}$ and $\{x_{jp}\}_{(j,p)\in\mathcal{A}}$.

We assume the communication network $\mathcal{G} = (\mathcal{V},\mathcal{E})$ consists of the underlying undirected network. This means nodes i and j can exchange messages directly, i.e., $(i,j) \in \mathcal{E}$ for $i < j$, if either $(i,j) \in \mathcal{A}$ or $(j,i) \in \mathcal{A}$. Therefore, in contrast with the flows, messages do not necessarily need to be exchanged satisfying the direction of the arcs. In fact, messages and flows might represent different physical quantities, e.g., electricity and water. In problem (15), the subgraph induced by x_{ij} contains only nodes i and j and an edge connecting them. This makes the variable in (15) star-shaped.

VI. Experimental Results

Using the applications from the previous section, we now show some illustrative experimental results for a connected variable. A thorough description of the experimental setup and more results (including for non-connected variables) can be found in [5, Sec. 4.4]. All experiments simulate a distributed environment in a single computer.

Network Flows: For network flows we considered (15) with $\phi_{ij}(x_{ij}) = x_{ij}/(c_{ij} - x_{ij}) + \mathrm{i}_{x_{ij}\leq c_{ij}}(x_{ij})$, where c_{ij} is the capacity of the arc $(i,j) \in \mathcal{A}$, and used a randomly generated Barabasi network with 2000 nodes and 3996 edges. The results are in Fig. 4, which shows the relative error $\|x^k - x^\star\|_\infty / \|x^\star\|_\infty$ versus the number of communication steps. Here, x^k is the concatenation of the estimates at all nodes, and $x^\star$ is the optimal solution, computed in a centralized way. One *communication step* (CS) consists of all nodes communicating their current estimates to their neighbors; the total number of

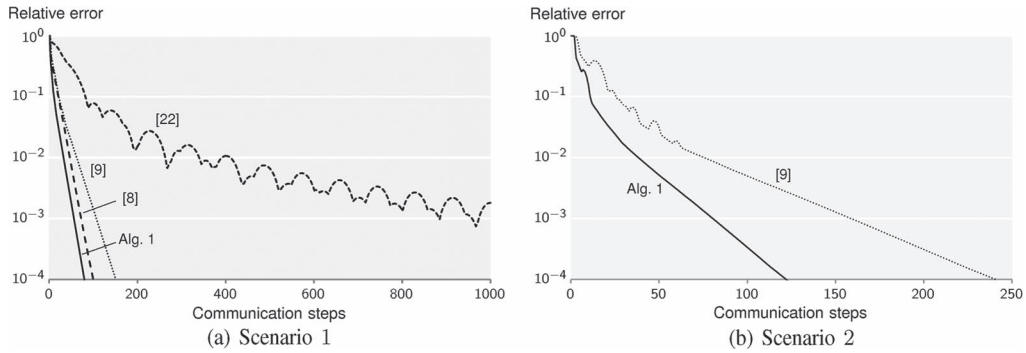

Fig. 5. Results for D-MPC. In (a), the network has 4941 nodes and 6594 edges, and all the components induce star-shaped subgraphs. In (b), the network has 100 nodes and 196 edges, and the variable is connected (but not star-shaped). All the systems in (a) are stable, and in (b) they are not necessarily stable.

CSs is thus proportional to the total number of communications. We compared the proposed algorithm (Alg. 1) with [8] and [9], whose algorithms coincide for this problem, and Nesterov's fast gradient algorithm [22]. In fact, when the variable is star-shaped, any gradient algorithm can be applied and becomes distributed (since the center of the star can act as a central node for that component). Fig. 4 shows that our proposed algorithm requires the least number of CSs to achieve any relative error between 10^{-1} and 10^{-4}.

D-MPC: The results for D-MPC are in Fig. 5, with two different scenarios: 1) the network has 4941 nodes and 6594 edges (topology of the Western States Power Grid [23]), the variable is star-shaped, and all subsystems are stable; 2) the network has 100 nodes and 196 edges (randomly generated Barabasi network), the variable is connected but not star-shaped, and most subsystems are unstable. See [5, Sec. 4.4] for a description on how we generated the system interactions in scenario 2). In both scenarios, we considered (14) with linear, time-invariant Θ_p^t's (system interactions) and quadratic Φ_p's and Ψ_p^t's. The size of the state (resp. input) at each node was always $n_p = 3$ (resp. $m_p = 1$), and the time-horizon was $T = 5$. This means the optimization variable in (14) had dimensions 24705 in scenario 1) and dimensions 500 in scenario 2). Fig. 5(a) shows the results for scenario 1) and Fig. 5(b) shows the results for scenario 2). Note that the only prior distributed algorithm that can solve problems for generic connected variables is [9], which is why only Alg. 1 and [9] appear in Fig. 5(b). For a star-shaped variable (Fig. 5(a)), both [8] and gradient-based methods [22] yield distributed algorithms. In both scenarios, Alg. 1 performed the best, requiring uniformly less CSs to achieve any relative error between 10^{-1} and 10^{-4}.

VII. Conclusions

We designed algorithms for distributed optimization problems where the function at each node depends on arbitrary components of the variable, rather than on all of them. We classify an optimization variable as connected or non-connected, and propose an algorithm for each. Our algorithms require a network coloring scheme, and their convergence is guaranteed only when the network is bipartite or when the objectives are strongly convex. In practice, however, the algorithms converge even when none of these conditions is met. Moreover, experimental results show that our algorithms require less communications than prior algorithms to solve D-MPC or network flow problems to arbitrary levels of accuracy.

References

[1] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Syst. Mag.*, vol. 22, no. 1, 2002.
[2] I. Necoara, V. Nedelcu, and I. Dumitrache, "Parallel and distributed optimization methods for estimation and control on networks," *J. Proc. Control*, vol. 21, 2011.
[3] R. Scattolini, "Architectures for distributed and hierarchical model predictive control—A review," *J. Proc. Control*, vol. 19, 2009.
[4] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
[5] J. Mota, "Communication-Efficient Algorithms For Distributed Optimization," Ph.D. thesis, Carnegie Mellon University, PA, Technical University of Lisbon, Lisbon, Portugal, 2013.
[6] H. Zhu, G. Giannakis, and A. Cano, "Distributed in-network channel decoding," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3790–3983, Oct. 2009.
[7] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2718–2723, Oct. 2013.
[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2011.
[9] V. Kekatos and G. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1617–1626, Feb. 2012.
[10] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Distributed ADMM for model predictive control and congestion control," in *Proc. 51st IEEE Conf. Decision and Control*, 2012, pp. 5110–5115.
[11] I. Necoara and D. Clipici, "Distributed Random Coordinate Descent Methods for Composite Minimization," University Politehnica Bucharest, Bucharest, Hungary, 2013, Tech. Rep.
[12] M. Zargham, A. Ribeiro, A. Jadbabaie, and A. Ozdaglar, "Accelerated dual descent for network optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905–920, 2014.
[13] C. Conte, T. Summers, M. Zeilinger, M. Morari, and C. Jones, "Computational aspects of distributed optimization in model predictive control," in *Proc. IEEE Conf. Decision and Control*, 2012.
[14] D. Han and X. Yuan, "A note on the alternating direction method of multipliers," *J. Optim. Theory Appl.*, 2012.
[15] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, A Proof of Convergence for the Alternating Direction Method of Multipliers Applied to Polyhedral-Constrained Functions 2011, arXiv: 1112.2295.
[16] W. Deng and W. Yin, "On the Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers," Dept. Computational and Applied Mathematics, Rice University, Houston, TX, 2012, Tech. Rep.
[17] M. Hong and Z. Luo, On the Linear Convergence of the Alternating Direction Method of Multipliers 2012, arXiv: 1208.3922.
[18] C. Chen, B. He, Y. Ye, and X. Yuan, The Direct Extension of ADMM for Multi-Block Convex Minimization Problems is not Necessarily Convergent, 2013. [Online]. Available: http://www.optimization-online.org/DB_FILE/2013/09/4059.pdf
[19] D. Williamson, "The primal-dual method for approximating algorithms," *Math. Program.*, vol. 91, no. B, pp. 447–478, 2002.
[20] G. Robins and A. Zelikovsky, "Improved Steiner tree approximation in graphs," in *11th anual ACM-SIAM symp. Discrete Algs.*, 2000.
[21] P. Chalermsook and J. Fakcharoenphol, "Simple distributed algorithms for approximating minimum Steiner trees," *Computing and Combinatorics, LNCS*, vol. 3595, pp. 380–389, 2005.
[22] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Norwell, MA, USA: Kluwer, 2003.
[23] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, 1998.