A UNIFIED ALGORITHMIC APPROACH TO DISTRIBUTED OPTIMIZATION

João F. C. Mota[1,2], *João M. F. Xavier*[2], *Pedro M. Q. Aguiar*[2], *and Markus Püschel*[3]

[1] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA
[2] Institute of Systems and Robotics, Instituto Superior Técnico, Technical University of Lisbon, Portugal
[3] Department of Computer Science, ETH Zurich, Switzerland

ABSTRACT

We address general optimization problems formulated on networks. Each node in the network has a function, and the goal is to find a vector $x \in \mathbb{R}^n$ that minimizes the sum of all the functions. We assume that each function depends on a set of components of x, not necessarily on all of them. This creates additional structure in the problem, which can be captured by the classification scheme we develop. This scheme not only to enables us to design an algorithm that solves very general distributed optimization problems, but also allows us to categorize prior algorithms and applications. Our general-purpose algorithm shows a performance superior to prior algorithms, including algorithms that are application-specific.

Index Terms— Distributed optimization, sensor networks

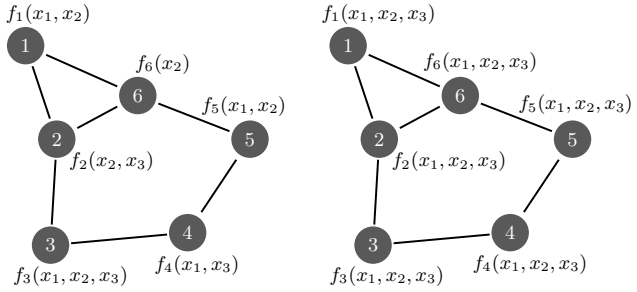
1. INTRODUCTION

Optimization algorithms are a fundamental tool in information processing. As information becomes more frequently generated in networks, processing it will require distributed optimization algorithms. We address a general class of optimization problems formulated on a network. Specifically, given a network with P nodes, we solve

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f_1(x_{S_1}) + f_2(x_{S_2}) + \cdots + f_P(x_{S_P}), \qquad (1)$$

where the function $f_p : \mathbb{R}^{n_p} \to \mathbb{R} \cup \{+\infty\}$ is known only at node p (see Fig. 1). Each function f_p depends on a subset of components of the variable $x \in \mathbb{R}^n$, $S_p \subseteq \{1, \ldots, n\}$. We use x_{S_p} to denote those components. Fig. 1(a) shows a network with $P = 6$ nodes where all the sets S_p are specified. For example, the function at node 4, f_4, depends on x_1 and x_3; therefore, $S_4 = \{1, 3\}$ and $f_4(x_{S_4}) = f_4(x_1, x_3)$. We assume that if node p depends on components x_{S_p}, then it is interested in computing the optimal value for those components only, and not for any of the other components. Node 4 in Fig. 1(a) then computes the optimal value of x_1 and x_3, but not the optimal value of x_2. Most of the literature on distributed optimization addresses the very particular case of (1) that is illustrated in Fig. 1(b): namely, each function f_p depends on *all* the components of the variable $x \in \mathbb{R}^n$, i.e., $S_p = \{1, \ldots, n\}$, for all $p = 1, \ldots, P$. We will say that problem (1), in this case, has a *global variable*. While several applications can be written as (1) with a global variable, many others are instances of (1) with a non-global variable. Examples include network utility maximization (NUM) [1, 2, 3], network flows [4], distributed model predictive control (D-MPC) [5], and state estimation in power networks [6]; see also [7]. These applications have motivated the design of distributed algorithms that solve (1) with a variable that is non-global, but *star-shaped*. We will define a star-shaped variable when we introduce our classification scheme for problem (1). This scheme, shown in Fig. 2,

Work supported by the grants CMU-PT/SIA/0026/2009, PTDC/EEA-ACR/73749/2006, SFRH/BD/33520/2008 (Carnegie Mellon/Portugal Program by ICTI), and by PEst-OE/EEI/LA0009/2013 from FCT.



(a) Non-connected variable (b) Global variable

Fig. 1. Instance of (1) with (a) a non-connected variable, and (b) a global variable. In (b), each function depends on all the components of x. In both cases, the variable x has three components: $x = (x_1, x_2, x_3)$.

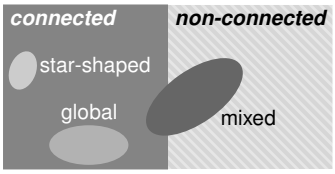


Fig. 2. Our classification scheme for x, which is either connected or non-connected. Global and star-shaped variables are particular instances of a connected variable, and a mixed variable can be connected or non-connected.

will help us achieve our goal: to design a distributed algorithm that solves (1) in full generality. The algorithm is distributed in the sense that no central node is allowed, and each node communicates only with its neighbors. This implies that all-to-all communications are forbidden. We also enforce local processing by requiring any operation involving function f_p to take place at node p. Despite the generality of our algorithm, it surprisingly can outperform in terms of communication-efficiency prior methods that were designed for particular instances of (1), or even for specific applications. This is illustrated in our experimental results for the average consensus problem and for D-MPC. Next, we formally state our problem and then present our classification scheme. This scheme will then help us review prior work.

Problem statement. We define *communication network* as the network through which the nodes communicate directly. We represent it with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, P\}$ is the set of nodes (with cardinality P) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (with cardinality E). If $(i, j) \in \mathcal{E}$, then nodes i and j can exchange messages directly. We assume $\mathcal{G}$ is connected and time-invariant. Each node $p \in \mathcal{V}$ has associated a function $f_p : \mathbb{R}^{n_p} \to \mathbb{R} \cup \{+\infty\}$, $1 \leq n_p \leq n$, which we assume closed, proper, and convex.

We solve the following problem: *given an arbitrary communication network with P nodes and P arbitrary sets $S_p \subseteq \{1, \ldots, P\}$, design an algorithm that solves problem* (1) *in a distributed way.* As

said before, distributed means central nodes or all-to-all communications are forbidden. Also, each function f_p is known only at node p.

Classification scheme. We solve problem (1) in a divide-and-conquer manner by using the classification scheme shown in Fig. 2. This classification scheme will also help us organize prior work and applications. Essential to our classification is the concept of induced subgraph. Let $x_l \in \mathbb{R}$ be the lth component of $x \in \mathbb{R}^n$. 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 $\mathcal{E}_l$ only if both $i, j \in \mathcal{V}_l$. For example, the subgraph induced by x_1 in Fig. 1(a) consists of the nodes that depend on x_1, $\mathcal{V}_1 = \{1, 3, 4, 5\}$, and contains the edges $(3, 4)$ and $(4, 5)$. We say that x_l is *connected* if $\mathcal{G}_l$ is connected, and is *non-connected* otherwise. The component x_1 in Fig. 1(a) is non-connected, because $\mathcal{G}_1$ is a non-connected subgraph (node 1 is "isolated"). In contrast, x_2 and x_3 are connected components, because their induced subgraphs are connected. We also say that a component is *star-shaped* if its induced subgraph is a star, i.e., if it consists of one central node to which all the other nodes in the subgraph, not being neighbors between themselves, are connected. For example, component x_3 in Fig. 1(a) is star-shaped, because nodes 2 and 4 are neighbors of node 3, but not between themselves. Finally, a component is *global* if it appears in all the functions of the network.

Using this component-wise classification, we now classify variable x of problem (1) according to Fig. 2. Namely, we say that x is *connected* if all its components are connected, and it is *non-connected* if it has at least one non-connected component. Hence, the variable in Fig. 1(a) is non-connected, because component x_1 is non-connected. We say that a variable is *star-shaped* when all its components are star-shaped, which implies that it is connected. As we had seen before, a variable is *global* if all its components are global, i.e., they appear in every function of the network. Since we assume a connected communication network, a global variable is always connected. Finally, a variable is *mixed* if it has both global and non-global components. Since the induced subgraphs of the non-global components can be either connected or non-connected, a mixed variable can also be either connected or non-connected.

Related work. To our best knowledge, (1) has been solved only with the following types of variable: global, mixed (whose non-global components are star-shaped), and star-shaped. A global variable is actually the most popular instance of (1) and has been solved, for example, with subgradient- and gradient-based methods [8, 9]. These algorithms are well characterized theoretically, but exhibit slow convergence. This contrasts with algorithms based on the Alternating Direction Method of Multipliers (ADMM) [10, 11, 12], which converge faster, but have less theoretical guarantees. The algorithm we propose here is based on an extended version of ADMM and generalizes the algorithm in [12], currently the most communication-efficient algorithm for (1) with a global variable.

Distributed algorithms for (1) with a star-shaped variable have been motivated by several applications, for example, D-MPC [5], network flows [4], NUM [1, 2, 3], and state estimation in power systems [6]. We mention that, with a star-shaped variable, the ADMM-based algorithm in [13, §7.2], which generally requires a central node, becomes distributed. Indeed, [5] has applied it to D-MPC and compared its performance against fast gradient methods. Although D-MPC has only been solved with a star-shaped variable, it easily extends to a generic connected, or even non-connected, variable, as we will see. Distributed Newton methods have also been proposed to solve (1) with a star-shaped variable, for example, in NUM [2] and in network flows [4]. All the algorithms mentioned so far only apply when the variable is star-shaped and cannot be easily generalized to a generic connected variable. The exception is the algorithm in [6], which is based on ADMM. It was proposed for estimating the state of power systems, a problem formulated as (1) with a star-shaped variable. In [7], we noticed that that algorithm can be generalized to a generic connected variable, and even to a non-connected one. The algorithm in [6] thus solves (1) in full generality, as the algorithm we propose here. Our experimental results, however, show that it requires systematically more communications to converge to a solution of (1) than the algorithm we propose.

Finally, we mention that [3] proposed a gradient algorithm solving an instance of (1) with a mixed variable: a NUM with coupled objectives. We are unaware of other algorithms in the literature that consider a mixed variable. In the next section, we introduce a class of problems that can be recast as (1) with a mixed variable and, thus, solved with the algorithm we propose.

Contributions. This paper builds on our previous work [12, 7] to design an algorithm that solves (1) in full generality. The algorithm in [12] solves (1) only with a global variable, and [7] solves it with variables that have no global components. In this paper, we notice that the derivation of algorithm [7] does not require the assumption that forbids global components and, thus, we obtain an algorithm more generic that both [12] and [7]. However, here we only focus on the case of a connected variable, since the adaptation to a non-connected variable is exactly as described in [7]. Note that the classification scheme in [7] differs from the one we propose here.

2. APPLICATION PROBLEMS

This section presents some applications that can be written as (1). We will focus on distributed model predictive control (D-MPC) for two reasons: it arises naturally with a generic variable, i.e., either connected or non-connected, and we will use it in our experiments.

Global variable. Several problems can be formulated as (1) with a global variable. For example, estimating a parameter $\theta \in \mathbb{R}^n$ using measurements of a sensor network can be formulated as (1), if each f_p measures the error between the estimated value and the measurements at node p. An example is average consensus [10, 11, 12, 14], where $f_p(x) = (x - \theta_p)^2$, and $\theta_p \in \mathbb{R}$ represents the measurement at node p. Another example is support vector machines where the databases are distributed over the nodes of a network [12].

Mixed variable. To our best knowledge, [3] is the only reference proposing an algorithm for (1) with a mixed variable. The problem addressed there is a particular case of:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & g_1(x) + g_2(x) + \cdots + g_P(x) \\ \text{subject to} & h_1(x) + h_2(x) + \cdots + h_P(x) \leq 0\,, \end{array} \tag{2}$$

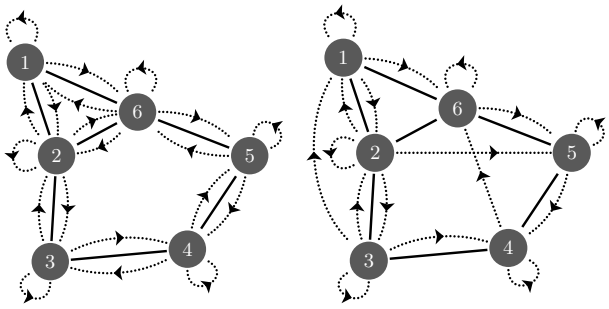
where $g_p : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $h_p : \mathbb{R}^n \to \mathbb{R}^m$ are closed, proper, and convex functions, known only at node p. To recast (2) as (1), create P copies of the variable $x \in \mathbb{R}^n$ and denote by $x_p \in \mathbb{R}^n$ the copy at node p. This enables writing it as

$$\begin{array}{ll} \underset{\{x_p\}}{\text{minimize}} & g_1(x_1) + g_2(x_2) + \cdots + g_P(x_P) \\ \text{subject to} & h_1(x_1) + h_2(x_2) + \cdots + h_P(x_P) \leq 0 \\ & x_i = x_j\,, \quad (i,j) \in \mathcal{E}\,, \end{array} \tag{3}$$

where the last constraint enforces equality of the copies. If each g_p is strictly convex and strong duality holds, then solving (3) is equivalent to solving its dual problem, which can be written as (1) with

$$f_p(\mu, \{\lambda_{pj}\}_{j\in\mathcal{N}_p}) = -\inf_{x_p}\Big[f_p(x_p) + \mu^\top h_p(x_p) + \big(\sum_{j\in\mathcal{N}_p} \text{sign}(j-p)\lambda_{pj}\big)^\top x_p\Big] + \text{i}_{\mathbb{R}^m_+}(\mu) \tag{4}$$

as the function of node p, where $\text{sign}(a) = 1$ if $a \geq 0$ and $\text{sign}(a) = 1$ if $a < 0$. Also, i_S is the indicator function of the set S, i.e.,



(a) Connected star-shaped variable (b) Non-connected variable

Fig. 3. Two D-MPC scenarios that yield (a) a connected variable whose induced subgraphs are stars, and (b) a non-connected variable. Solid lines are communication network links and dotted arrows are system interactions.

$\mathrm{i}_S(x) = 0$ if $x \in S$, and $\mathrm{i}_S(x) = +\infty$ if $x \notin S$. In this case, $S = \mathbb{R}^m_+$ is the set of nonnegative numbers in $\mathbb{R}^m$. The variable has global components $\mu \in \mathbb{R}^m$, which is the dual variable associated to the first constraint of (3), and the components $\{\lambda_{ij}\}_{(i,j)\in\mathcal{E}}$, where each λ_{ij} is a dual variable associated to the second constraint. Note that the subgraph induced by each λ_{ij} is star-shaped.

Generic variable: D-MPC. In D-MPC, each node p represents a system described by a time-dependent state vector $x_p[t]$ and is controlled by an input vector $u_p[t]$. The current state $x_p[t]$ can be influenced, not only by the past state and input of node p, but also by states and inputs of other nodes in the network, denoted with $\Omega_p \subseteq \mathcal{V}$. Hence, $x_p[t+1] = \Theta^t_p\big(\{x_j[t], u_j[t]\}_{j\in\Omega_p}\big)$, for some function Θ^t_p. The goal in D-MPC is to drive each state to a given target using minimum input energy, that is, to solve

$$\begin{array}{ll} \min\limits_{\{\overline{x}_p,\overline{u}_p\}} & \sum_{p=1}^{P}\Big[\phi_p(x_p[T+1]) + \sum_{t=1}^{T}\Phi^t_p(u_p[t])\Big] \\ \text{s.t.} & x_p[t+1] = \Theta^t_p\big(\{x_j[t], u_j[t]\}_{j\in\Omega_p}\big)\,,\ t = 1,\dots,T, \\ & x_p[1] = x^1_p\,, \\ & p = 1,\dots,P\,, \end{array} \tag{5}$$

where $\overline{x}_p$ (resp. $\overline{u}_p$) is the set of states (resp. inputs) of node p from $t = 1$ to $t = T+1$ (resp. from $t = 1$ to $t = T$) and T is the time-horizon. While the function ϕ_p penalizes deviations from the goal of the final state $x_p[T+1]$ of node p, Φ^t_p measures the energy consumed by the input $u_p[t]$ at time t. Each node p in D-MPC performs the following actions: first, it measures its current state x^1_p; then, it cooperates with the other nodes to solve (5); finally, it applies $u_p[1]$, and repeats the process. Problem (5) can be written as (1) by setting

$$f_p(\{\overline{x}_j, \overline{u}_j\}_{j\in\Omega_p\cup\{p\}}) = \phi_p(x_p[T+1]) + \mathrm{i}_{x_p[1]=x^1_p}(\overline{x}_p) + \sum_{t=1}^{T}\Big(\Phi^t_p(u_p[t]) + \mathrm{i}_{\Gamma^t_p}(\{\overline{x}_j, \overline{u}_j\}_{j\in\Omega_p})\Big) \tag{6}$$

as the function of node p, where $\Gamma^t_p = \big\{\{\overline{x}_j, \overline{u}_j\}_{j\in\Omega_p} : x_p[t+1] = \Theta^t_p\big(\{x_j[t], u_j[t]\}_{j\in\Omega_p}\big)\big\}$. It is then clear that, if no node influences all the other nodes, (5) is an instance of (1) with a generic variable. Fig. 3 illustrates two cases. In both plots, we represent the communication network with solid lines and the interactions, i.e., the sets Ω_p, with dotted arrows. In Fig. 3(a), each node is influenced only by the states/inputs of its neighbors and itself. That is, the subgraph induced by $x_p[t]$, for any $t = 2,\dots,T+1$, is star-shaped, with node p in the center. This is the case for which D-MPC algorithms have been designed. In contrast, in Fig. 3(b), node 1 is influenced by node 3, which is not its neighbor. However, since node 2 is also influenced by node 3, there is an indirect communication between nodes 1 and 3. In other words, the subgraph induced by $(\overline{x}_3, \overline{u}_3)$ is connected. A different case is the influence of node 2 in node 5. Here, since node 2 does not influence neither node 4 nor node 6, there is not a path (of nodes influenced by node 2) between nodes 2 and 5. That is, the subgraph induced by $(\overline{x}_2, \overline{u}_2)$ is non-connected.

Algorithm 1 Algorithm for a connected variable

Initialization: for $p \in \mathcal{V}$, $l \in S_p$, set $\gamma^{(p),1}_l = x^{(p),1}_l = 0$; $k = 1$

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

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

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

$$\underset{x^{(p)}_{S_p}=\{x^{(p)}_l\}_{l\in S_p}}{\arg\min} \; f_p(x^{(p)}_{S_p}) + \sum_{l\in S_p} v^{(p),k\,\top}_l x^{(p)}_l + \frac{\rho}{2}\sum_{l\in S_p} D_{p,l}\big(x^{(p)}_l\big)^2$$

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

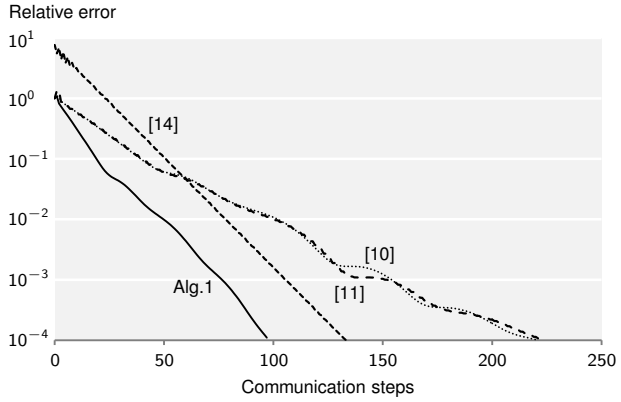
$$\gamma^{(p),k+1}_l = \gamma^{(p),k}_l + \rho\textstyle\sum_{j\in\mathcal{N}_p\cap\mathcal{V}_l}(x^{(p),k+1}_l - x^{(j),k+1}_l)$$

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

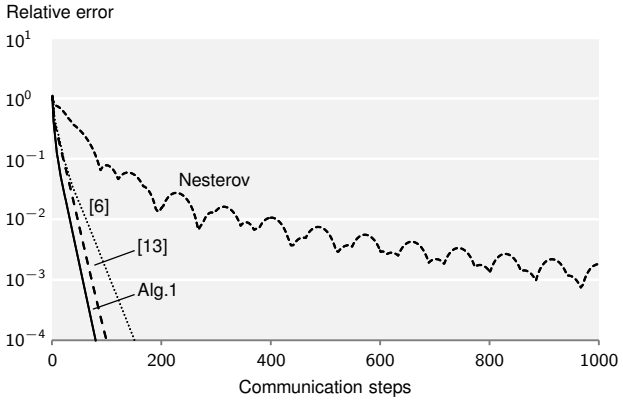
3. ALGORITHM AND EXPERIMENTAL RESULTS

The algorithm we propose is outlined in Algorithm 1. The difference with respect to the algorithm in [7] is that here we assume that the variable can have global components. Consequently, Algorithm 1 applies to all the classes in Fig. 2 and is, indeed, more general than [7]. For simplicity, and due to space constraints, we only present the algorithm for a connected variable. The procedure described in [7] can be easily extended to Algorithm 1 with a non-connected variable. We notice that when the variable is global, Algorithm 1 becomes D-ADMM [12], which is currently the state-of-the-art for a global variable in terms of the number of communications.

Brief description. Before executing Algorithm 1, each node is assigned a number c, called color, such that no neighbors have the same color. These colors, which we assumed given, are numbered as $c = 1,\dots,C$ and synchronize the nodes in the same way as the TDMA protocol does: the nodes work sequentially, but nodes with the same color can work in parallel. This makes Algorithm 1 integrate naturally with TDMA. Most of the algorithms for distributed optimization assume all the nodes work in parallel. In a shared medium access such as wireless communications, that is not possible without using collision avoidance protocols, e.g., TDMA. The algorithm works as follows. Consider a given node p with color c, and assume it has received, from the neighbors with smaller colors, estimates of their common components. That is, node p has received estimates of x^{k+1}_l from all $j \in \mathcal{N}_p$ such that $\mathcal{C}(j) < \mathcal{C}(p)$ and $l \in \mathcal{S}_j \cap \mathcal{S}_p$, where $\mathcal{C}(i)$ is the color of node i. Node p can then compute $v^{(p)}_l$ as shown in step 4. There, $\gamma^{(p),k}_l$ is a (dual) variable internal to the node, and $\rho > 0$ is the augmented Lagrangian parameter, known by all the nodes. Next, node p updates the estimates of the components it depends on, as in step 6. There, $D_{p,l}$ is the number of neighbors of node p that also depend on x_l. After that, node p sends its new estimates selectively to its neighbors, i.e., it sends $x^{(p),k+1}_l$ to $j \in \mathcal{N}_p$ only if that node also depends on x_l. Af-



(a) Average consensus, geometric network, 2000 nodes



(b) D-MPC, power grid network, 4941 nodes

Fig. 4. Results for (a) an average consensus problem, which has a global variable, and (b) an D-MPC problem with a star-shaped variable.

ter the neighbors of node p have updated their estimates in a similar way and sent the relevant ones to node p, node p can update its dual variables γ_l as in step 10. All nodes perform the described procedure and then they move to the next iteration. It can be checked that the proofs for the convergence results in [7] also hold for Algorithm 1, i.e., when the variable is allowed to have global components.

Some experimental results. Fig. 4 shows the performance of Algorithm 1, measured as the relative error in the solution with respect to the number of communication steps. We say that a *communication step* has occurred when all the nodes in the network have updated their estimates and broadcast them to the neighbors. In Fig. 4(a), Algorithm 1 is applied to average consensus, which an instance of (1) with a global variable. We used a geometric network with $P = 2000$ nodes and parameter $\sqrt{\log(P)/P} \simeq 0.06$. That plot also shows the performance of the ADMM-based algorithms [10, 11] and [14], the fastest consensus algorithm. Note that while [10, 11] solve problems in the entire global class, [14] only solves consensus. We used $\rho = 1.1$ for the augmented Lagrangian parameter in Algorithm 1 and $\rho = 0.6$ for [10, 11]; these values are known to be within 2% from the optimal ρ. The plot shows that Algorithm 1 was the one that required the least amount of communications to achieve a 10^{-4} relative error. The line of [14], however, shows an offset. This is because the nodes using that algorithm had to be initialized with a special value, whereas in the other algorithms they were initialized with 0. For this particular instance, that offset was unfavorable to [14]. However, it can be noticed that the slopes of the error lines of Algorithm 1 and [14] are roughly the same; this indicates that both algorithms have about the same performance, if the nodes are initialized the same way. This is indeed the case, as confirmed by other experiments [12].

Fig. 4(b) shows the results for an D-MPC problem with a star-shaped variable (so that we could run other algorithms besides Algorithm 1 and [6]). The network had 4941 nodes and represents the US power grid (see [7] for more details). We compared Algorithm 1 with the ADMM-based algorithms [13, 6] and with a Nesterov gradient algorithm. The augmented Lagrangian parameter ρ was 25 for Algorithm 1 and 30 for the other two ADMM-based algorithms. In both cases, ρ is within 0.5 from the optimal one. Again, Algorithm 1 was the one requiring less communications to converge.

4. CONCLUSIONS

We proposed a distributed algorithm that solves general optimization problems formulated on networks. To accomplish this, we devised a scheme for classifying distributed optimization problems, and used it to categorize existing application problems and algorithms. Although our algorithm is very general, in the sense that it solves problems in all the classes, it shows an excellent communication-efficiency. In fact, our experimental results show that our algorithm performs better in this sense than other general-purpose algorithms and, for some specific applications, it performs as well as the best algorithms available.

5. REFERENCES

[1] S. Low, L. Peterson, and L. Wang, "Understanding Vegas: a duality model," *Journal of the ACM*, vol. 49, no. 2, pp. 207–235, 2002.

[2] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utility maximization, I: Algorithm," Tech. Rep., LIDS - 2832, 2011.

[3] C. Tan, D. Palomar, and M. Chiang, "Distributed optimization of coupled systems with applications to network utility maximization," in *IEEE Inter. Conf. Acoust., Speech, Signal Process.*, 2006, pp. 981–984.

[4] M. Zargham, A. Ribeiro, A. Jadbabaie, and A. Ozdaglar, "Accelerated dual descent for network optimization," `http://arxiv.org/abs/1104.1157`, 2012.

[5] C. Conte, T. Summers, M. Zeilinger, M. Morari, and C. Jones, "Computational aspects of distributed optimization in model predictive control," in *IEEE Conf. Decision and Contr.*, 2012, pp. 6819–6824.

[6] V. Kekatos and G. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. P. Sys.*, vol. 28, no. 2, pp. 1617–1626, 2012.

[7] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in MPC and network flows," `http://arxiv.org/abs/1305.1885`, 2013.

[8] A. Nedić and Ozdaglar, *Convex Optimization in Signal Processing and Communications*, chapter Cooperative distributed multi-agent optimization, Cambridge University Press, 2010.

[9] J. Chen and A. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Sig. Proc.*, vol. 60, no. 8, pp. 4289–4305, 2012.

[10] H. Zhu, G. Giannakis, and A. Cano, "Distributed in-network channel decoding," *IEEE Trans. Sig. Proc.*, vol. 57, no. 10, pp. 3970–3983, 2009.

[11] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in *ad hoc* wsns with noisy links - Part I: Distributed estimation of deterministic signals," *IEEE Trans. Sig. Proc.*, vol. 56, no. 1, pp. 350–364, 2008.

[12] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Trans. Sig. Proc.*, vol. 61, no. 10, pp. 2718–2723, 2013.

[13] 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, pp. 1–122, 2011.

[14] B. Oreshkin, M. Coates, and M. Rabbat, "Optimization and analysis of distributed averaging with short node memory," *IEEE Trans. Sig. Proc.*, vol. 58, no. 5, pp. 2850–2865, 2010.