Efficient Distributed Algorithms For Structured Optimization Problems

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

***Abstract*—We propose a distributed algorithm for solving optimization problems formulated on networks. Each network node has a private cost function and the goal is to minimize the sum of all these functions. We assume the optimization problem has some structure, namely, that the function at each node might depend only on a subset of the components of the variable. Experimental results show that the proposed algorithm requires less communications than previous algorithms to achieve a given error, including algorithms that were designed for a specific application. We apply the algorithm to distributed Model Predictive Control and to Network Utility Maximization.**

I. Introduction

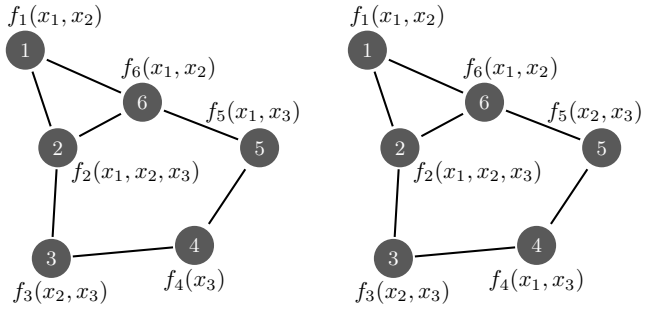
Consider the following optimization problem:

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

where $x \in \mathbb{R}^n$ is the variable and $S_p \subseteq \{1,\ldots,n\}$ is the set of components of x that function f_p depends on. Given $S \subseteq \{1,\ldots,n\}$, x_S denotes the components indexed by S; for example, if $S = \{2,7\}$, then $x_S = (x_2, x_7)$. We associate to problem (1) a network with P nodes, represented with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1,\ldots,P\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Whenever $(i,j) \in \mathcal{E}$, nodes i and j can communicate directly. We enforce local processing by requiring that each function f_p is known only at node p: then, any operation performed on f_p has to take place at node p. Our goal is to solve (1) using a minimal number of communications.

Most of the previous work on distributed optimization, e.g., [1], [2], assumes all f_p's depend on all components of x, i.e., $S_p = \{1,\ldots,n\}$ for all p. In many problems, however, each function depends only on a subset of these components. This occurs in some large-scale network problems, for example, in *Network Utility Maximization* (NUM), modeling the Internet traffic control, and in distributed *Model Predictive Control* (MPC), which has many applications in the power grid.

To address the generic problem (1), we distinguish between two cases: a connected and a non-connected variable. While in a connected variable all the components induce connected subgraphs, in a non-connected variable at least one component induces a non-connected subgraph. The subgraph induced by a component x_l is $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$, where $\mathcal{V}_l \subset \mathcal{V}$ is the set of nodes whose functions depend on x_l, and $(i,j) \in \mathcal{E}_l \subset \mathcal{E}$ iff both $i \in \mathcal{V}_l$ and $j \in \mathcal{V}_l$ and $(i,j) \in \mathcal{E}$. For example, in the subgraph induced by x_1 in Fig. 1(a), $\mathcal{V}_1 = \{1,2,5,6\}$ and $\mathcal{E}_1 = \{(1,2),(1,6),(2,6),(5,6)\}$. In this case, $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ is connected, as there is a path between any node in $\mathcal{G}_1$. The same is true for x_2 and x_3, and thus the variable in Fig. 1(a) is connected. In contrast, the subgraph induced by x_1 in Fig. 1(b), whose nodes are $\mathcal{V}_1 = \{1,2,4,6\}$, is not connected (node 4 is isolated). Hence, the variable is non-connected in this case. Solving (1) in this case is challenging because node 4 has to communicate somehow with the other nodes in $\mathcal{V}_1$, so that they agree on an optimal value for x_1.



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

Figure 1. Networks where each node depends on a subset of components of the variable $x = (x_1, x_2, x_3)$.

We propose a distributed algorithm solving (1) for a generic variable (connected or non-connected). Our algorithm is based on an extended version of the *Alternating Direction Method of Multipliers* (ADMM) [3] and generalizes our previous algorithm [2], in the sense that it becomes [2] when we make the variable global, i.e., we make each function depend on all the components: $S_p = \{1,\ldots,n\}$ for all p. In fact, [2] is currently the most communication-efficient algorithm for (1) when the variable is global. However, it is easily outperformed by other algorithms, e.g., the gradient method, when the variable is not global [4]. This is because it makes all the nodes exchange full estimates of the variable and not just the components they are interested in, obviously using communications wastefully. The algorithm proposed here makes the nodes exchange only the components they are interested in, if the variable is connected; if it is non-connected, some nodes (Steiner nodes) will exchange some additional components, required to solve the problem. To the

João F. C. Mota, João M. F. Xavier, and Pedro M. Q. Aguiar are with Instituto de Sistemas e Robótica (ISR), Instituto Superior Técnico (IST), Technical University of Lisbon, Portugal.

Markus Püschel is with the Department of Computer Science at ETH Zurich, Switzerland.

João F. C. Mota is also with the Department of Electrical and Computer Engineering at Carnegie Mellon University, USA.

This work was supported by the following grants from Fundação para a Ciência e Tecnologia (FCT): CMU-PT/SIA/0026/2009, PEst-OE/EEI/LA0009/2011, and SFRH/BD/33520/2008 (through the Carnegie Mellon/Portugal Program managed by ICTI).

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 for** $\boldsymbol{k = 1, 2, \ldots}$
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

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

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**

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

11: **end for**
12: **until** some stopping criterion is met

best of our knowledge, this is the first time an algorithm has been proposed to solve (1) in its full generality. Even so, the proposed algorithm requires fewer communications than many previous algorithms designed for very specific cases, e.g., when all induced subgraphs are stars.

II. Proposed Method

Algorithm 1 shows the proposed algorithm for a connected variable. For a non-connected variable, the algorithm requires an adaptation which includes solving a Steiner tree problem for each non-connected component. In Algorithm 1, the nodes are synchronized according to a scheme very similar to TDMA (and thus it perfectly integrates with this protocol). More specifically, the network is colored with C colors, i.e., node p is assigned a color $\mathcal{C}(p)$ such that none of its neighbors $\mathcal{N}_p$ in the communication graph has the same color; then, nodes with the same color work in parallel, with the different colors working sequentially. At each iteration, node p solves an optimization problem (step 6) involving its private function f_p. The solution to that problem is the current estimate of node p for the components x_{S_p}. In step 7, these estimates are sent to the neighbors who are also interested in the same components: namely, if $l \in S_p$ and $j \in \mathcal{N}_p \cap \mathcal{V}_l$, node p sends its estimate of x_l to node j. The convergence is guaranteed by:

Theorem 1. *Let $\mathcal{G}$ be connected and its topology fixed time-wise. Let each function $f_p : \mathbb{R}^{n_p} \to \mathbb{R} \cup \{+\infty\}$ be closed and convex over $\mathbb{R}^{n_p}$. 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 $\{x_{S_p}^{(p),k}\}_{k=1}^{\infty}$, produced by Algorithm 1, converges to $x_{S_p}^{\star}$, where $x^{\star}$ solves* (1)*.*

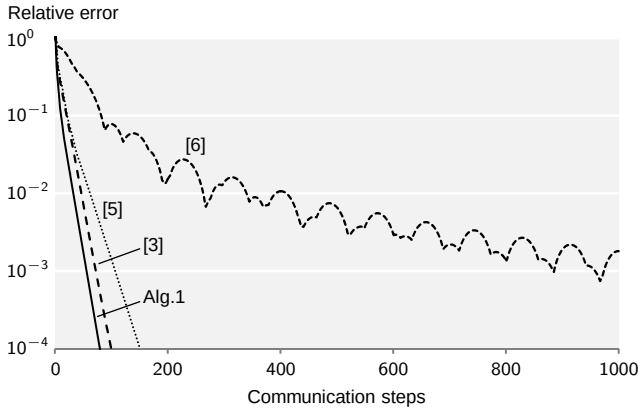


Figure 2. Results for an MPC problem on a network with 4941 nodes.

III. Experimental Results

We illustrate the efficiency of Algorithm 1 on a network with 4941 nodes, representing the Western states power grid [7]. We solved an MPC problem on this network, associating to each node p a state x_p and an input u_p. The interaction model we used was $x[t+1] = A_p x_p[t] + \sum_{j \in \mathcal{N}_p \cup \{p\}} B_{pj} u_j[t]$, i.e., the systems represented by the nodes are coupled through their inputs, with the state at a given node being influenced, not only by its own input, but also by its neighbors' inputs. The variable in this case is connected, and each induced subgraph is a star, thus, a very particular case of (1). Even though, as shown in Fig. 2, the proposed algorithm requires less communication steps to achieve any relative error between 10^{-1} and 10^{-4} than other algorithms that solve the same problem. A communication step corresponds to having all nodes transmit their current estimates to their neighbors. To our best knowledge, the algorithm in [5] is the only prior algorithm that can efficiently solve (1) when the variable is connected and the induced subgraphs are not stars. All the other algorithms, e.g., [3], [6], require additional, wasteful communications to handle that case. Our experiments show that Algorithm 1 requires always less communications to converge than [5], independently of the structure of the variable.

In conclusion, the algorithm we propose here, in addition to solving the very generic problem (1), is able to explore the structure of that problem to significantly reduce the number of required communications.

References

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

[2] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," to appear in IEEE Trans. Sig. Proc., 2013.

[3] 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.

[4] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Distributed ADMM for model predictive control and congestion control," in *CDC Conf.*, 2012.

[5] V. Kekatos and G. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Systems*, vol. PP, no. 99, pp. 1–10, 2012.

[6] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, 2003.

[7] M. Newman, "Power grid," http://www-personal.umich.edu/~Emejn/netdata/, retrieved on Feb 14, 2013.