D-ADMM: A DISTRIBUTED ALGORITHM FOR COMPRESSED SENSING AND OTHER SEPARABLE OPTIMIZATION PROBLEMS

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 propose a distributed, decentralized algorithm for solving separable optimization problems over a connected network of compute nodes. In a separable problem, each node has its own private function and its own private constraint set. Private means that no other node has access to it. The goal is to minimize the sum of all nodes' private functions, constraining the solution to be in the intersection of all the private sets. Our algorithm is based on the alternating direction method of multipliers (ADMM) and requires a coloring of the network to be available beforehand. We perform numerical experiments of the algorithm, applying it to compressed sensing problems. These show that the proposed algorithm requires in general less iterations, and hence less communication between nodes, than previous algorithms to achieve a given accuracy.

***Index Terms*—** Distributed optimization, compressed sensing, basis pursuit, network optimization

1. INTRODUCTION

The interest in distributed processing methods has increased significantly over the last years. At least two scenarios contributed to this: the emergence of sensor networks that generate and process distributed data, and the increasing need for processing large amounts of data on large scale, distributed computing platforms. Since many data processing algorithms are based on optimization, there is a need for new, *distributed* optimization algorithms.

In this paper we consider separable optimization problems. A separable optimization problem has the form

$$\begin{array}{ll} \text{minimize} & f_1(x) + f_2(x) + \cdots + f_P(x) \\ \text{subject to} & x \in X_1 \cap X_2 \cap \cdots \cap X_P \,, \end{array} \tag{1}$$

where the minimization is with respect to (w.r.t.) $x \in \mathbb{R}^n$. We propose an algorithm for solving (1) in a distributed way. By distributed we mean that, given a network with P nodes, we associate with node p a function f_p and a set X_p and require these to be private to node p. This means that no other node has access to f_p or X_p at any time during the algorithm. We make some assumptions on the functions f_p, on the sets X_p, and on the given network.

Assumption 1. *Each function $f_p : \mathbb{R}^n \to \mathbb{R}$ is convex, and each set $X_p \subset \mathbb{R}^n$ is convex. Also, $\cap_{p=1}^P X_p \neq \emptyset$.*

Assumption 2. *The given network, or graph, is connected and its topology does not vary with time.*

Assumption 3. *A coloring of the graph is available.*

While Assumptions 1 and 2 are standard in algorithms solving (1), Assumption 3 is not as common, but is used many times at lower implementation levels for avoiding packet collisions. A coloring of a graph is an assignment of numbers, which we call colors, to each node of that graph such that no neighboring nodes have the same number.

Our algorithm is novel and based on the alternating direction of multipliers [1, §3.4] (ADMM). It is fully decentralized in the sense that it uses no special or central node. We present experimental results of our algorithm applied to compressed sensing (CS) [2] problems, which demonstrate that our algorithm requires in general less communication than its competitors.

Applications. There are many inherently distributed problems in control and signal processing that can be written as (1). Examples include: projected consensus [3], resource allocation problems [4], cognitive radio [5], and distributed support vector machines [6].

Here, we apply our algorithm to solve two CS problems that are essential for the reconstruction of an acquired/compressed signal. These are the Basis Pursuit (BP) [2]

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & Ax = b \end{array} \tag{2}$$

and the Basis Pursuit De-Noising (BPDN) [7]

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}\|Ax - b\|^2 + \beta\|x\|_1 \,, \tag{3}$$

where the variable in both problems is $x \in \mathbb{R}^n$, and the matrix $A \in \mathbb{R}^{m \times n}$ and the vector $b \in \mathbb{R}^m$ are given. We consider A and b to be partitioned by rows, i.e., $A = \begin{bmatrix} A_1^\top & \cdots & A_P^\top \end{bmatrix}^\top$ and $b = \begin{bmatrix} b_1^\top & \cdots b_P^\top \end{bmatrix}^\top$, where $A_p \in \mathbb{R}^{m_p \times n}$ and $b_p \in \mathbb{R}^{m_p}$, with $m_1 + \cdots + m_P = m$. We assume A_p and b_p are known by node p only, but the total number of nodes P and the parameter β in (3) are known by all nodes. Hence, (2) can be written as (1) by setting $f_p(x) = \frac{1}{P}\|x\|_1$ and $X_p = \{x : A_p x = b_p\}$. Similarly, (3) is written as (1) doing $f_p(x) = \frac{1}{2}\|A_p x - b_p\|^2 + \frac{\beta}{P}\|x\|_1$ and $X_p = \mathbb{R}^n$. In CS, solving (2) and (3) is a required step to reconstruct the signal of interest from the acquired/compressed signal b, respectively, in noiseless and noisy settings. And, although there is some literature about the properties of distributed CS (e.g. [2]), the problem of distributed reconstruction has not yet been studied much. There are several applications for distributed CS reconstruction including

This work was supported by the FCT grant CMU-PT/SIA/0026/2009, PTDC/EEA-ACR/73749/2006 and SFRH/BD/33520/2008 (through the Carnegie Mellon/Portugal Program by ICTI) from Fundação para a Ciência e Tecnologia and also by ISR/IST plurianual funding (PIDDAC Program).

geographical modeling and healthcare [2], or processing wideband signals captured by narrowband antennas [8].

Related work. Problem (1) is closely related to decomposition methods [4], where subgradient methods are very popular [3]. However, in spite of their nice convergence properties (e.g., noise robustness), they are rather slow. Other distributed algorithms solving (1) include the method of multipliers concatenated with the nonlinear Gauss-Seidel method [9] or with the diagonal quadratic approximation [10], a double-looped algorithm using Nesterov's method in both loops [8], and an algorithm called D-Lasso [5] that is, as ours, based on ADMM. In [11], we compared a particular case, namely BP, of the proposed algorithm with all these algorithms and concluded that D-Lasso is the only truly competitor. Hence, in section 3, we only compare our algorithm with D-Lasso. The distributed version of ADMM that has been proposed in [12] requires a central node; our algorithm is completely decentralized.

Contributions. We propose a new algorithm for solving separable optimization problems (1) in a distributed way. This algorithm extends our preliminary work [11], where we proposed an algorithm for solving (2) specifically. Extensive experiments for (2) and (3) indicate that our algorithm requires in almost all scenarios less communications than previous algorithms. We believe this fact can have a significant impact on applications.

2. PROPOSED METHOD

We assume a network (undirected graph) with P nodes and E edges and represent its graph with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, P\}$ is the set of nodes and $\mathcal{E} = \{\ldots, (i,j), \ldots\}$ is the set of edges. A link $(i,j) \in \mathcal{E}$ means that nodes i and j can communicate. The set $\mathcal{N}_p$ represents the set of neighbors of node p and $D_p := |\mathcal{N}_p|$ its degree. A prerequisite for our algorithm is a coloring scheme [1], such that every pair of adjacent nodes have different colors. The minimum number of colors graph $\mathcal{G}$ requires is represented by $\chi(\mathcal{G})$.

The algorithm we propose is only proven to converge for graphs such that $\chi(\mathcal{G}) = 2$, called bipartite graphs (e.g., a grid graph). Although we have no proof of convergence for graphs with $\chi(\mathcal{G}) > 2$, our algorithm never failed to converge in any of our simulations for that type of graphs.

Algorithm for bipartite graphs. We present the algorithm for bipartite graphs since it is simpler, and once understood, its generalization to graphs with $\chi(\mathcal{G}) > 2$ is straightforward. Assume that $\mathcal{G}$ has two colors, 1 and 2, and represent the nodes with color i in the set $\mathcal{C}_i$, $i = 1, 2$. Also assume, without loss of generality, that nodes numbered from 1 to $|\mathcal{C}_1|$ are in $\mathcal{C}_1$ and the remaining are in $\mathcal{C}_2$.

In problem (1), all nodes have to agree on a solution $x^\star$. Due to the distributed nature of the problem, we replicate a copy of the variable x throughout all the nodes. The copy in node p is denoted with $x_p \in \mathbb{R}^n$. To guarantee equivalence to (1), we have to constrain all these copies to be equal, for example requiring $x_i = x_j$ for all $(i,j) \in \mathcal{E}$. Let $B \in \mathbb{R}^{P \times E}$ be the node-arc incidence matrix of the graph, i.e., a matrix where each column represents the (i,j)th edge and has -1 and 1 in its ith and jth entries, respectively. The remaining entries have zeros. Then, the constraint $x_i = x_j$, for all $(i,j) \in \mathcal{E}$ is written more compactly as $(B^\top \otimes I_n)\bar{x} = 0$, where $\otimes$ is the Kronecker product, I_n is the identity matrix in $\mathbb{R}^n$, and $\bar{x} = (x_1, \ldots, x_P) \in (\mathbb{R}^n)^P$. Therefore, (1) is equivalent to

$$\begin{array}{ll} \text{minimize} & \sum_{p=1}^P f_p(x_p) \\ \text{subject to} & x_p \in X_p\,, \quad p = 1, \ldots, P \\ & (B^\top \otimes I_n)\bar{x} = 0\,, \end{array} \tag{4}$$

with variable $\bar{x}$. Due to our numbering scheme, we can partition B into $\begin{bmatrix} B_1^\top & B_2^\top \end{bmatrix}^\top$ and rewrite (4) as

$$\begin{array}{ll} \text{minimize} & \sum_{p \in \mathcal{C}_1} f_p(x_p) + \sum_{p \in \mathcal{C}_2} f_p(x_p) \\ \text{subject to} & x_p \in X_p\,, \quad p = 1, \ldots, P \\ & (B_1^\top \otimes I_n)\bar{x}_1 + (B_2^\top \otimes I_n)\bar{x}_2 = 0\,, \end{array} \tag{5}$$

where $\bar{x} = (\bar{x}_1, \bar{x}_2) \in \mathbb{R}^{c_1} \times \mathbb{R}^{c_2}$, and $c_i = |\mathcal{C}_i|$, $i = 1, 2$. Now we apply the alternating direction method of multipliers [1] (ADMM) to problem (5). Consider the augmented Lagrangian

$$\begin{aligned} L(\bar{x}_1, \bar{x}_2; \lambda) = \sum_{p \in \mathcal{C}_1} f_p(x_p) + \sum_{p \in \mathcal{C}_2} f_p(x_p) + \phi(\bar{x}_1, B_1; \lambda) \\ + \phi(\bar{x}_2, B_2; \lambda) + 2\bar{x}_1^\top (B_1 B_2^\top \otimes I_n)\bar{x}_2\,, \end{aligned} \tag{6}$$

where $\phi(\bar{x}_i, B_i; \lambda) = \lambda^\top (B_i^\top \otimes I_n)\bar{x}_i + \frac{\rho}{2}\|(B_i^\top \otimes I_n)\bar{x}_i\|^2$, for $i = 1, 2$, and $\rho > 0$ is a predefined parameter. ADMM is an iterative algorithm which, at iteration k, minimizes $L(\bar{x}_1, \bar{x}_2^{(k)}; \lambda^{(k)})$ w.r.t. $\bar{x}_1$, finding $\bar{x}_1^{(k+1)}$; then, minimizes $L(\bar{x}_1^{(k+1)}, \bar{x}_2; \lambda^{(k)})$ w.r.t. $\bar{x}_2$, finding $\bar{x}_2^{(k+1)}$; and finally, updates the dual variable $\lambda = (\ldots, \lambda_{ij}, \ldots) \in (\mathbb{R}^n)^E$ as $\lambda^{(k+1)} = \lambda^{(k)} + \rho((B_1^\top \otimes I_n)\bar{x}_1^{(k+1)} + (B_2^\top \otimes I_n)\bar{x}_2^{(k+1)})$. It can be shown that each minimization step decomposes into several optimization problems, one for each node in $\mathcal{C}_1$ or $\mathcal{C}_2$, whether the minimization is w.r.t. $\bar{x}_1$ or $\bar{x}_2$, respectively. After each minimization step, the nodes transmit their estimate of $x^\star$ to their neighbors (which, of course, have a different color). It can also be shown that the dual variable λ does not need to be updated directly: node p can update the auxiliary variable $\gamma_p^{(k)} = \sum_{j \in \mathcal{N}_p} \text{sign}(j - p)\lambda_{pj}^{(k)}$ instead. The result is the following algorithm, which we name D-ADMM, where the 'D' comes from "distributed."

Algorithm 1 D-ADMM-I for bipartite networks

```
Initialization: for all $p \in \mathcal{V}$, set $\gamma_p^{(1)} = x_p^{(1)} = 0$ and $k = 1$
1: repeat
2:     for all $p \in \mathcal{C}_1$ [in parallel] do
3:         Set $v_p^{(k)} = \gamma_p^{(k)} - \rho \sum_{j \in \mathcal{N}_p} x_j^{(k)}$ and find
               $x_p^{(k+1)} = \begin{array}{ll} \text{argmin} & f_p(x_p) + {v_p^{(k)}}^\top x_p + \frac{D_p \rho}{2}\|x_p\|^2 \\ \text{s.t.} & x_p \in X_p \end{array}$
4:         Send $x_p^{(k+1)}$ to $\mathcal{N}_p$
5:     end for
6:     Repeat 2-5 for all $p \in \mathcal{C}_2$, replacing $x_j^{(k)}$ by $x_j^{(k+1)}$ in $v_p^{(k)}$
7:     for all $p \in \mathcal{V}$ [in parallel] do
           $\gamma_p^{(k+1)} = \gamma_p^{(k)} + \rho \sum_{j \in \mathcal{N}_p} (x_p^{(k+1)} - x_j^{(k+1)})$
8:     end for
9:     $k \leftarrow k + 1$
10: until some stopping criterion is met
```

It can be seen that nodes with different colors cannot operate in parallel, but all the nodes with the same color operate in parallel. At each iteration, each node solves the optimization problem in step 3, which depends only on local data (function f_p and the set X_p), but requires knowledge of the neighbors' estimates. Those estimates are communicated in step 4. After step 6 is executed, all nodes have performed one minimization step and have sent the respective solution to all their neighbors. Next, in step 7, they update the auxiliary variable γ_p in parallel. Regarding the convergence of Algorithm 1, the following theorem holds.

Theorem 1 ([11]). *For each $p = 1, \dots, P$, the sequence $\{x_p^{(k)}\}$ produced by Algorithm 1 converges to a solution of* (1)*.*

The proof consists of showing that (5) satisfies the conditions of the theorem that establishes the convergence of ADMM [1]. In particular, the matrices $(B_i^\top \otimes I_n)$, $i = 1, 2$ have to be full column-rank, but this follows from well known properties of the matrix B.

Algorithm for general graphs. The generalization of Algorithm 1 to graphs colored with more than two colors (in particular, $\chi(\mathcal{G}) > 2$) is now straightforward: in each iteration, steps 2–5 are repeated for all colors. Unfortunately, Theorem 1 no longer applies because the proof of convergence of ADMM [1] cannot be easily generalized to the case when the variable is partitioned into more than two blocks. However, empirical evidence shown in next section suggests that the conclusions of the theorem may still hold.

3. EXPERIMENTAL RESULTS

We now present results from the simulations of Algorithm 1 solving (2) and (3). As mentioned, we only compare our algorithm with D-Lasso, since it is the only competitive one. In the case of (2), a comparison with other algorithms can found in [11]. In particular, Algorithm 1 applied to (2) always outperformed any other algorithm for all the considered networks and data types.

Performance measure. We will use the number of communication steps to assess the performance of an algorithm. Communicating is the most energy-consuming task in sensor networks and the runtime bottleneck in computer clusters. Therefore, the less communications an algorithm uses, the more energy-efficient or faster that algorithm can be. D-ADMM and D-Lasso have a very similar structure: both consist of a single loop where in each iteration each node solves the optimization problem in step 3 of Algorithm 1, and communicates its solution to all its neighbors. Thus, one iteration of both algorithms is comparable since they perform the same computations and transmit the same amount of information across the nodes. To emphasize that we are interested in the number of communications, we will use the term *communication step* to denote an iteration of either D-ADMM or D-Lasso. Note that, given a network with E edges, $2E$ multiplied by the number of communication steps gives the total number of communications in the network. Also, note that while D-ADMM operates asynchronously, D-Lasso can perform synchronously. Thus, in environments where synchronism is allowed, D-Lasso can have smaller execution times in spite of using more communications. We note, however, that such environments are rare in practice; for example, in wireless networks the packet collision problem prevents synchronism.

Experimental setup. The data we used for simulating the algorithms comes from CS. In one case, we generate our own data: each entry in the matrix $A \in \mathbb{R}^{500 \times 2000}$ was drawn from an i.i.d. Gaussian distribution with zero mean and variance $1/500$. The vector b was generated from multiplying A by a vector with 80 nonzero random entries, located at random places. In the other case, we used data provided by the Sparco toolbox [13], namely problem 7 ($A \in \mathbb{R}^{600 \times 2560}$). Regarding the networks, we generated 7 different networks with $P = 10$ nodes according to a random model, shown in Table 1. A description of the parameters of these models can be found, e.g., in [11]. We mention that the only network that was not generated by a random model was the Lattice one: it is just a 2-dimensional grid with dimensions 2×5, thus it is bipartite. Hence, this is the only network for which D-ADMM is proven to converge. For the other networks, although there is not a guarantee of convergence, D-ADMM always converged.

Table 1. Network models for the experiments.

Network number	Model	Parameters
1	Erdős-Rényi	$p = 0.25$
2	Erdős-Rényi	$p = 0.75$
3	Watts-Strogatz	$(n, p) = (4, 0.6)$
4	Watts-Strogatz	$(n, p) = (2, 0.8)$
5	Barabasi-Albert	———
6	Geometric	$d = 0.75$
7	Lattice	———

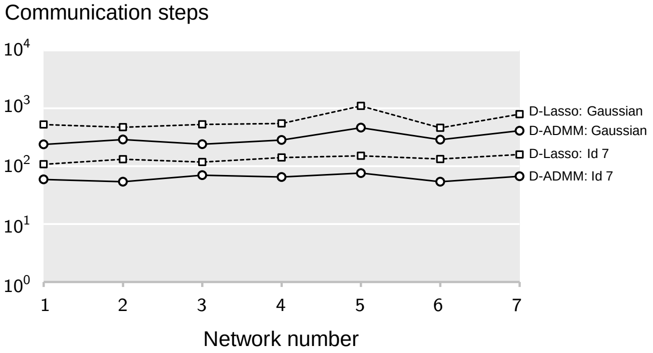
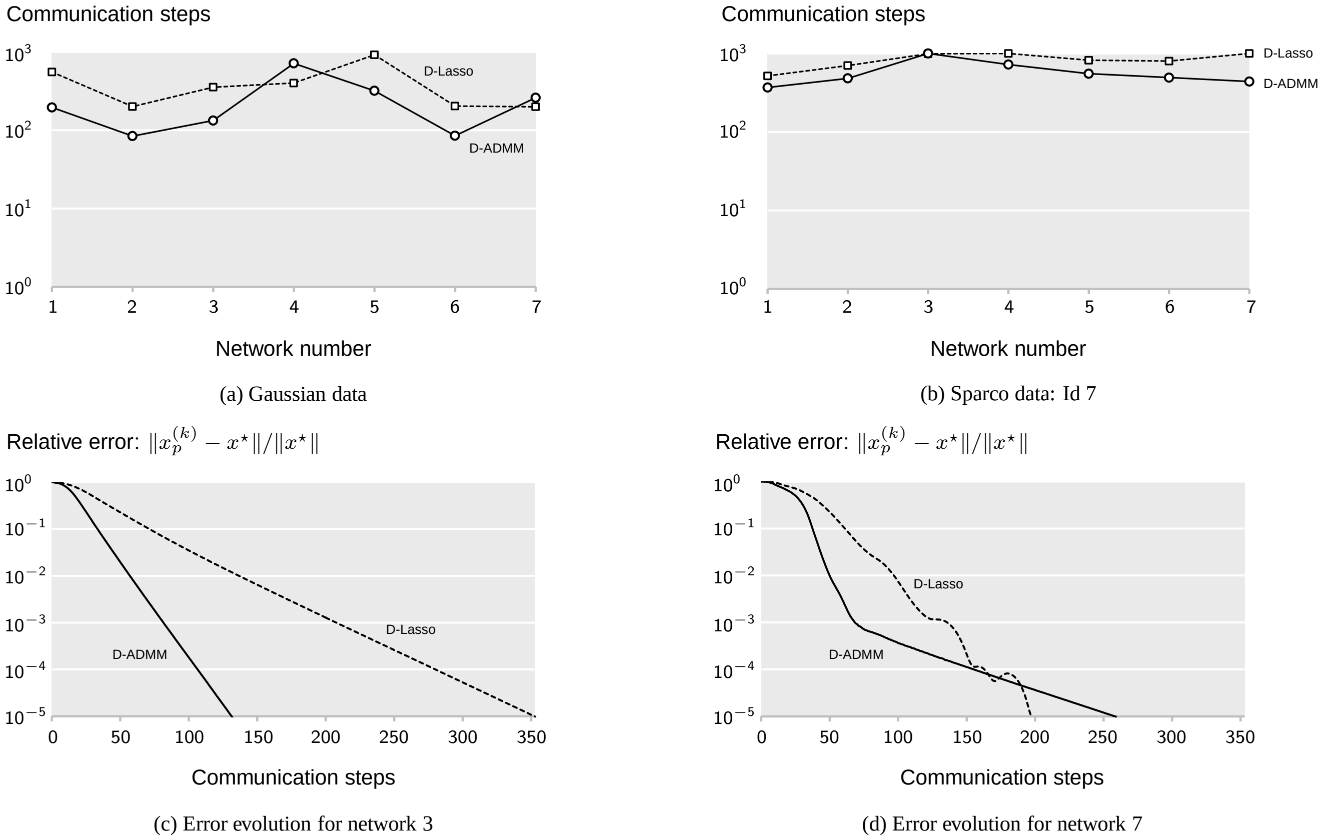


Fig. 1. BP results for Gaussian and Sparco (Id 7) data.

Both D-ADMM and D-Lasso are augmented Lagrangian-based algorithms. Therefore, they depend on a parameter ρ (see, e.g., (6)), which is chosen beforehand. It is known that the performance of augmented Lagrangian-based algorithms depends strongly on the choice of ρ. To mitigate that dependence, we designed our experiments the following way: given a network and a dataset, we run both D-ADMM and D-Lasso 5 times, where each run has a different value for ρ, chosen from the set $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$. And we always choose the best result, i.e., the one that leads to the least number of communication steps. In each run, the algorithms stop whenever they achieve a $10^{-3}\%$ accuracy at an arbitrary node, i.e., $\|x_p^{(k)} - x^\star\|/\|x^\star\| \leq 10^{-5}$ for an arbitrary node p, or when the maximum number of iterations 10^3 is reached. The solution $x^\star$ for BP or BPDN is computed beforehand using centralized algorithms.

Results. Figs. 1 and 2 show the results of our simulations for problems (2) (BP) and (3) (BPDN), respectively. In Fig. 1 we represent the number of communication steps as a function of network number (c.f. Table 1). There, we see that D-ADMM always required less communication steps than D-Lasso to achieve $10^{-3}\%$ of accuracy, for both types of data. The optimal values for ρ were always 10^{-2}, 10^{-1}, or 1, for both algorithms.

In Figs. 2(a) and 2(b) we present the same type of plots but for BPDN, for Gaussian and for Sparco data, respectively. In contrast with what happened for BP, here there are two cases where D-Lasso required less communication steps than D-ADMM: for networks 4 and 7 under Gaussian data. For illustration, we represent in Figs. 2(c) and 2(d) how the error evolved along the iterations for two particular cases: one for which D-ADMM required less communication steps than D-Lasso (Gaussian data, network 3), and another for which it



(a) Gaussian data

(b) Sparco data: Id 7

(c) Error evolution for network 3

(d) Error evolution for network 7

Fig. 2. BPDN results for Gaussian (a) and Sparco (b) data. Relative error along the iterations for Gaussian data: networks 3 (c) and 7 (d).

required more communication steps than D-Lasso (Gaussian data, network 7). In Fig. 2(c) we see that D-ADMM required uniformly less communication steps than D-Lasso to achieve any accuraccy between 10^{-1} and 10^{-5}. For network 7, however, this only happened until an accuracy of 10^{-4}, value after which D-Lasso started requiring less communication steps. This explains Fig. 2(a).

4. CONCLUSIONS

We proposed a distributed algorithm for solving separable optimization problems over a network of nodes. Each node has a private function that wants to minimize, and a private constraint set. All nodes reach a solution together by exchanging solution estimates in each iteration. By assuming that a coloring scheme is available beforehand, we are able to apply ADMM to our problem formulation. The resulting algorithm requires, in general, less communications than the state-of-the-art algorithms, as shown by numerical simulations for compressed sensing reconstruction problems.

5. REFERENCES

[1] D. Bertesekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.

[2] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, “Compressed sensing for networked data,” *IEEE Sig. Proc. Magazine*, vol. 25, no. 2, 2008.

[3] A. Nedić, A. Ozdaglar, and P. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Trans. Aut. Contr.*, vol. 55, no. 4, pp. 922–938, 2010.

[4] D. Palomar and M. Chiang, “A tutorial on decomposition methods for network utility maximization,” *IEEE J. Sel. Areas in Communications*, vol. 4, no. 8, pp. 1439–1451, 2006.

[5] J. Bazerque and G. Giannakis, “Distributed spectrum sensing for cognitive radio networks by exploiting sparsity,” *IEEE Trans. Sig. Proc.*, vol. 58, no. 3, 2010.

[6] O. Mangasarian and E. Wild, “Privacy-preserving classification of horizontally partitioned data via random kernels,” Tech. Rep., Data Mining Institute, 07-03, 2007.

[7] M. Figueiredo, R. Nowak, and S. Wright, “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems,” *IEEE J. Sel. Topics Sig. Proc.*, vol. 1, no. 4, 2007.

[8] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, “Basis pursuit in sensor networks,” in *IEEE Proc. ICASSP*, 2011.

[9] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, “Distributed algorithms for basis pursuit,” in *Workshop Sparse’09, Saint-Malo, France*, 2009.

[10] A. Ruszczynski, “Augmented lagrangian decomposition for sparse convex optimization,” *Inter. Inst. Applied Systems Analysis*, 1992.

[11] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, “Distributed basis pursuit,” *IEEE Trans. Sig. Proc.*, to appear, 2012.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating method of multipliers,” *Found. Trends Machine Learning*, vol. 3, no. 1, 2011.

[13] E. Berg, M. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and Ö. Yilmaz, “Sparco: a testing framework for sparse reconstruction,” Tech. Rep., Dept. Computer Science, Univ. British Columbia, 2007.