ADMM For Consensus On Colored Networks

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

***Abstract*— We propose a novel distributed algorithm for one of the most fundamental problems in networks: the average consensus. We view the average consensus as an optimization problem, which allows us to use recent techniques and results from the optimization area. Based on the assumption that a coloring scheme of the network is available, we derive a decentralized, asynchronous, and communication-efficient algorithm that is based on the Alternating Direction Method of Multipliers (ADMM). Our simulations with other state-of-the-art consensus algorithms show that the proposed algorithm is the one exhibiting the most stable performance across several network models.**

I. Introduction

Over the last decade, many algorithms have been proposed for solving the averaging consensus problem: "given a network of nodes, where each node holds a number, compute the average of all the numbers and make it available in all nodes." The algorithms for this simple yet fundamental problem are usually provided together with a detailed theoretical analysis, for example, convergence guarantees and rates of convergence. Solving the average consensus problem in a decentralized way is important in many applications including sensor networks (e.g., clock synchronization and parameter estimation), coordination of mobile autonomous agents, and modeling social networks [1], [2], [3], [4].

One way to address consensus is to solve the following optimization problem in a distributed way [5], [6]:

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\sum_{p=1}^{P}(x-\theta_p)^2\,, \qquad (1)$$

where θ_p is the number held by node p, as shown in Fig. 1. The solution to (1) can be easily computed by differentiation as $x^\star = \theta^\star = (1/P)\sum_{p=1}^{P}\theta_p$, the average of all the nodes' numbers. The goal in consensus is to make this average available in all nodes, without aggregating data in any kind of central or special node. The work in [5] applied an incremental version of the subgradient method to (1), resulting in a distributed, asynchronous algorithm. In [6], (1) was solved by applying the Alternating Direction Method of Multipliers (ADMM) to two different reformulations of (1).

In this paper, we introduce a novel, asynchronous consensus algorithm that is also based on solving (1) in a distributed manner. Although we also apply ADMM, our approach fundamentally differs from [6], since we consider a different reformulation of (1) and use an extended version of ADMM. Our reformulation assumes that a node coloring scheme of the network is available beforehand. Based on this assumption, we derive an algorithm whose performance in terms of number of communications is comparable to that of [1] and [7], the state-of-the-art algorithms for fast consensus. The reformulation we apply here was originally proposed in [8] in the context of finding sparse solutions of linear systems and recently applied to other optimization problems in [9].

The amount of communication used by a distributed algorithm is an important measure of performance, since communicating in a network is usually the most energy-consuming operation, e.g., in sensor networks, or the slowest one, e.g., in supercomputing. Our assumption of having a node coloring available beforehand is realistic in many distributed scenarios. For example, some medium access (MAC) protocols such as TDMA [10] already rely on some kind of node coloring. Node coloring is a well studied problem and many distributed algorithms are available (see [9] and the references therein).

Related work. The existing algorithms for average consensus can be divided in two classes: synchronous and asynchronous algorithms. In a synchronous algorithm all nodes perform the same operations at the same time, including exchanging their solution estimates. If x_p^k represents the estimate (of $\theta^\star$) of node p at discrete time k, the canonical format of a synchronous consensus algorithm is [1], [4]

$$x_p^{k+1} = a_{pp}x_p^k + \sum_{j\in\mathcal{N}_p} a_{pj}x_j^k\,, \qquad (2)$$

where $\mathcal{N}_p$ is the set of neighbors of node p and the a_{ij}'s

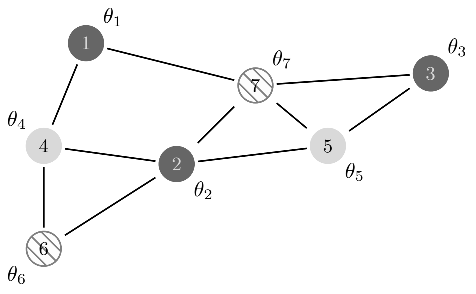


Fig. 1. Example network with $P = 7$ nodes and $E = 10$ edges. The network is colored with the following scheme: $\mathcal{C}_1 = \{1,2,3\}$, $\mathcal{C}_2 = \{4,5\}$, $\mathcal{C}_3 = \{6,7\}$.

*This work was supported by the following grants from Fundação para a Ciência e Tecnologia (FCT): CMU-PT/SIA/0026/2009, PTDC/EEA-ACR/73749/2006, PEst-OE/EEI/LA0009/2011, and SFRH/BD/33520/2008 (through the Carnegie Mellon/Portugal Program managed by ICTI).

[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

are positive weights that satisfy $\sum_{j \in \mathcal{N}_p \cup \{p\}} a_{pj} = 1$, for all p. The estimation variable x_p^k is initialized with the value held by node p: $x_p^0 = \theta_p$. Algorithm (2) can be written in the matrix format $x^{k+1} = Ax^k$, where $x^k = (x_1^k, \ldots, x_P^k)$ is the vector of estimates, and the ijth entry of the matrix A contains a_{ij}. Note that when nodes i and j do not communicate directly a_{ij} is set to zero, which makes the nonzero pattern of A reflect the topology of the network. The convergence of (2) is usually studied through the spectral properties of A. While in synchronous algorithms all nodes exchange estimates in the same time slot, in asynchronous algorithms only a subset of nodes exchanges estimates per time slot. Asynchronous algorithms include randomized gossiping [11], [2] and deterministic gossiping [12], [2]. At each iteration of a canonical gossiping algorithm, two neighboring nodes i and j are selected to exchange their estimates and, after their exchange occurs, each estimate is updated as $x_i^{k+1} = x_j^{k+1} = (x_i^k + x_j^k)/2$. Among all the existing synchronous and asynchronous consensus algorithms, the algorithms with faster convergences rates are [1], [7] (see also [6]). The algorithm we propose here is considered asynchronous (deterministic) and uses as few communications as [1], [7], for many networks. This will be shown in our simulations in section III.

Regarding methods for distributed optimization, we point out the work of [13], which couples consensus algorithms with subgradient algorithms to solve unconstrained optimization problems. The subgradient algorithm, however, makes the resulting method too slow. Several instances of optimization problems were solved with the algorithms in [14] and [15], which are also ADMM-based. In particular, both algorithms were applied to the consensus problem in [6]. An overview and comparison of distributed optimization algorithms for several problems can be found in [8], [9].

II. Proposed Algorithm

In this section, we derive and analyze the proposed algorithm. We start by introducing some notation.

Notation. We assume a network with P nodes and E edges and represent it with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are respectively the set of nodes and the set of edges. See Fig. 1 for an example. Each edge is represented with $\{i,j\} = \{j,i\}$, and $\{i,j\} \in \mathcal{E}$ means that nodes i and j communicate directly and thus can exchange their estimates. We represent the set of neighbors of node p with $\mathcal{N}_p$ and its degree with $D_p = |\mathcal{N}_p|$.

As stated before, we assume a (node) coloring scheme is available before the execution of the algorithm. A coloring scheme is an assignment of numbers, called colors, to each node such that no neighboring nodes have the same color. Fig. 1 shows a coloring scheme that uses 3 colors. If a network is colored with C colors, we represent the nodes that have color c with $\mathcal{C}_c$, $c = 1, \ldots, C$. The number of nodes in $\mathcal{C}_c$ is denoted with $C_c = |\mathcal{C}_c|$. Without loss of generality, we assume the nodes are numbered such that the first C_1 nodes are in $\mathcal{C}_1$, the next C_2 nodes are in $\mathcal{C}_2$, and so on. Fig. 1 illustrates this numbering scheme.

Problem manipulation. For simplicity, let $f_p(x)$ represent the term in the cost function of (1) that node p knows, i.e., $f_p(x) = \frac{1}{2}(x - \theta_p)^2$. The goal is to solve

$$\underset{x}{\text{minimize}} \ \sum_{p=1}^{P} f_p(x)\,, \tag{3}$$

in a distributed way. Since node p only knows $f_p(x)$, all nodes have to cooperate in order to solve (3). Note that x is a global variable whose optimal value $\theta^\star$ every node wants to know. This motivates us to replicate copies of x throughout the entire network: node p will hold the pth copy, x_p, which will be updated iteratively during the algorithm. We have, however, to guarantee that all the copies are equal. Among the several options for doing that, we choose to make it edge-wise, i.e., if $\{i,j\} \in \mathcal{E}$, we constrain $x_i = x_j$, with the convention that $i < j$. This yields the following problem

$$\begin{array}{ll} \underset{\bar{x} = (x_1, \ldots, x_P)}{\text{minimize}} & \sum_{p=1}^{P} f_p(x_p) \\ \text{subject to} & x_i = x_j\,, \quad \{i,j\} \in \mathcal{E}\,, \end{array} \tag{4}$$

where the new variable is the collection of all copies: $\bar{x} = (x_1, \ldots, x_P) \in \mathbb{R}^P$. Since we assume the network is connected (it has only one component), all copies are equal and consequently (3) and (4) are equivalent problems.

We now note that the constraints in (4) can be written more compactly as $B^\top \bar{x} = 0$, where $B \in \mathbb{R}^{P \times E}$ is the node-arc incidence matrix of the given network. In B, each column is associated to an edge of the network: the column associated to $\{i,j\} \in \mathcal{E}$ has 1 in the ith entry and -1 in the jth entry; the remaining entries have zeros. Given our assumption on the ordering of the nodes and the coloring scheme, we can write $B^\top \bar{x} = B_1^\top \bar{x}_1 + B_2^\top \bar{x}_2 + \cdots + B_C^\top \bar{x}_C$, where $\bar{x}_c$ collects the copies of the nodes in $\mathcal{C}_c$, i.e.,

$$\bar{x} = (\underbrace{x_1, \ldots, x_{C_1}}_{\bar{x}_1}, \ldots, \underbrace{x_{P-C_p+1}, \ldots, x_P}_{\bar{x}_C})\,,$$

and B is partitioned by rows accordingly. Therefore, (4) is written equivalently as

$$\begin{array}{ll} \underset{\bar{x} = (\bar{x}_1, \ldots, \bar{x}_C)}{\text{minimize}} & \sum_{c=1}^{C} \sum_{p \in \mathcal{C}_c} f_p(x_p) \\ \text{subject to} & B_1^\top \bar{x}_1 + \cdots + B_C^\top \bar{x}_C = 0\,. \end{array} \tag{5}$$

Note that in (5) we aggregated the functions of the nodes in $\mathcal{C}_c$ in the sum $\sum_{p \in \mathcal{C}_c} f_p(x_p)$. This is the format to which the Extended ADMM applies, as explained next.

Extended ADMM. The Alternating Direction Method of Multipliers (ADMM) solves the following problem:

$$\begin{array}{ll} \underset{x_1, x_2}{\text{minimize}} & g_1(x_1) + g_2(x_2) \\ \text{subject to} & A_1 x_1 + A_2 x_2 = 0\,, \end{array} \tag{6}$$

where g_1 and g_2 are closed, convex functions, and the matrices A_1 and A_2 have full column rank. ADMM consists of iterating the following equations

$$x_1^{k+1} = \arg\min_{x_1} L_\rho(x_1, x_2^k; \lambda^k) \tag{7}$$

$$x_2^{k+1} = \arg\min_{x_2} L_\rho(x_1^{k+1}, x_2; \lambda^k) \tag{8}$$

$$\lambda^{k+1} = \lambda^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1})\,, \tag{9}$$

where

$$L_\rho(x_1, x_2; \lambda) = g_1(x_1) + g_2(x_2) + \lambda^\top(A_1 x_1 + A_2 x_2) + \frac{\rho}{2}\|A_1 x_1 + A_2 x_2\|^2 \tag{10}$$

is the augmented Lagrangian of (6) with parameter $\rho > 0$. First, the augmented Lagrangian is minimized with respect to (w.r.t.) the first variable x_1 (see (7)); next, using the new value for x_1, L_ρ is minimized w.r.t. x_2. Finally, the dual variable λ is updated in a gradient ascent way as (9). The literature about ADMM is vast: see for example [16], [17] and the references therein. It is known that the scheme (7)-(9) converges to a solution of (6) under very mild assumptions. Moreover, if one is interested only in the optimal value of (6), not the variables x_1 and x_2 solving it, the assumption of the full column rankness of A_1 and A_2 can be dropped. The same assumption can be dropped in case g_1 and g_2 are strictly convex, since this is enough to guarantee that the sequences produced by (7)-(9) converge [18].

There is a natural generalization of (7)-(9) for solving an extension of (6) for the sum of $C \geq 2$ functions in the objective and for the sum of C terms in the constraints, i.e.,

$$\begin{array}{ll} \underset{x_1,\ldots,x_C}{\text{minimize}} & \sum_{c=1}^C g_c(x_c) \\ \text{subject to} & \sum_{c=1}^C A_c x_c = 0\,, \end{array} \tag{11}$$

where each function g_c is closed and convex and each matrix A_c has full column rank. Iterations (7)-(9) can be generalized to solve this problem as follows:

$$x_1^{k+1} = \arg\min_{x_1} L_\rho(x_1, x_2^k, \ldots, x_C^k; \lambda^k) \tag{12}$$

$$x_2^{k+1} = \arg\min_{x_2} L_\rho(x_1^{k+1}, x_2, x_3^k, \ldots, x_C^k; \lambda^k)$$

$$\vdots$$

$$x_C^{k+1} = \arg\min_{x_C} L_\rho(x_1^{k+1}, x_2^{k+1}, \ldots, x_{C-1}^{k+1}, x_C; \lambda^k) \tag{13}$$

$$\lambda^{k+1} = \lambda^k + \rho \sum_{c=1}^C A_c x_c^{k+1}\,, \tag{14}$$

where the augmented Lagrangian is now

$$L_\rho(x_1, \ldots, x_C; \lambda) = \sum_{c=1}^C g_c(x_c) + \lambda^\top\Big(\sum_{c=1}^C A_c x_c\Big) + \frac{\rho}{2}\Big\|\sum_{c=1}^P A_c x_c\Big\|^2. \tag{15}$$

Although practical evidence suggests that the convergence results for (7)-(9) may still hold for (12)-(14), that still remains an open question. Very recently, though, it was proved that the sequence produced by (12)-(14) solves (11) under the assumption that all the functions are strongly convex [18]. This result will guarantee the convergence of the method we propose, since in our case the objective functions are strongly convex. The convergence proof in [18] does not require full column rankness of each matrix A_c: strong convexity of all the functions is enough for proving the result.

Algorithm 1 Consensus On Colored Networks

```
Initialization: for all $p \in \mathcal{V}$, set $\gamma_p^1 = x_p^1 = 0$ and $k = 1$
 1: repeat
 2:     for $c = 1, \ldots, C$ do
 3:         for all $p \in \mathcal{C}_c$ [in parallel] do
 4:             $v_p^k = \gamma_p^k - \rho \sum_{j \in \mathcal{N}_p, j<p} x_j^{k+1} - \rho \sum_{j \in \mathcal{N}_p, j>p} x_j^k$
 5:             $x_p^{k+1} = (\theta_p - v_p^k)/(1 + \rho D_p)$
 6:             Send $x_p^{k+1}$ to $\mathcal{N}_p$
 7:         end for
 8:     end for
 9:     for all $p \in \mathcal{V}$ [in parallel] do
            $\gamma_p^{k+1} = \gamma_p^k + \rho \sum_{j \in \mathcal{N}_p}(x_p^{k+1} - x_j^{k+1})$
10:     end for
11:     $k \leftarrow k + 1$
12: until some stopping criterion is met
```

Applying ADMM. Formulating the consensus problem as (5) enables us to apply ADMM, since (5) and (11) have the same format: make the association $g_c(\bar{x}_c) = \sum_{p \in \mathcal{C}_c} g_p(x_p)$ and $A_c = B_c^\top$, for $c = 1, \ldots, C$. If we apply iterations (12)-(14) directly to (5), we see that each update of $\bar{x}_c$ yields C_c problems that can be solved in parallel. For example, the first block variable $\bar{x}_1$ is updated as

$$\bar{x}_1^{k+1} = \underset{\bar{x}_1 = (x_1, \ldots, x_{C_1})}{\arg\min} \sum_{p \in \mathcal{C}_1} f_p(x_p) + {\lambda^k}^\top B_1^\top \bar{x}_1 + \frac{\rho}{2}\Big\|B_1^\top \bar{x}_1 + \sum_{c=2}^C B_c^\top \bar{x}_c^k\Big\|^2, \tag{16}$$

where we dropped the functions and the linear terms not depending on $\bar{x}_1$. Developing the squared term in (16),

$$\bar{x}_1^\top B_1 B_1^\top \bar{x}_1 + 2\bar{x}_1^\top \sum_{c=2}^C B_1 B_c^\top \bar{x}_c^k + \Big\|\sum_{c=2}^C B_c^\top \bar{x}_c^k\Big\|^2. \tag{17}$$

In the first term of (17), $B_1 B_1^\top$ is the first diagonal block (of size $C_1 \times C_1$) of the network Laplacian, which is a diagonal matrix because the first C_1 nodes have the same color and hence cannot be neighbors; each diagonal element is the degree of node p, D_p, and $\bar{x}_1^\top B_1 B_1^\top \bar{x}_1 = \sum_{p \in \mathcal{C}_1} D_p x_p^2$. Regarding the second term, $B_1 B_c^\top$ is an off-diagonal block of the Laplacian and contains -1 in the ijth entry if nodes i and j are neighbors. Therefore, $\bar{x}_1^\top \sum_{c=2}^C B_1 B_c^\top \bar{x}_c^k = -\sum_{p \in \mathcal{C}_1} \sum_{j \in \mathcal{N}_p} x_p x_j^k$. Finally, the last term of (17) does not depend on $\bar{x}_1$ and can dropped from the optimization problem. Problem (16) is then simplified to

$$\bar{x}_1^{k+1} = \underset{\bar{x}_1 = (x_1, \ldots, x_{C_1})}{\arg\min} \sum_{p \in \mathcal{C}_1} \Big(f_p(x_p) + \big(\gamma_p^k - \rho \sum_{j \in \mathcal{N}_p} x_j^k\big)x_p + \frac{\rho D_p}{2} x_p^2\Big)\,, \tag{18}$$

where $\gamma_p^k := \sum_{j \in \mathcal{N}_p} \lambda_{\{p,j\}}^k$ was obtained from the second term in (16): $(B_1\lambda^k)^\top \bar{x}_1 = \sum_{p\in\mathcal{C}_1}\sum_{j\in\mathcal{N}_p}\lambda_{\{p,j\}}x_p$. We decomposed the dual variable λ as $(\ldots,\lambda_{\{i,j\}},\ldots)$, where $\lambda_{\{i,j\}}$ is associated to the constraint $x_i = x_j$, i.e., to the edge between node i and j. Clearly, there are not interdependencies in the cost function of (18) and, therefore, it decomposes into C_1 problems that can be solved in parallel. The same applies to the other block variables; the only difference is that there must be paid attention to the nodes' relative numbering in the sum defining γ_p^k. Its general definition is $\gamma_p^k := \sum_{j\in\mathcal{N}_p}\text{sign}(j-p)\lambda_{\{p,j\}}^k$, where sign$(a)$ gives 1 (resp. -1) if $a \geq 0$ (resp. < 0). Algorithm 1 shows the resulting algorithm for the consensus problem, i.e., with $f_p(x) = \frac{1}{2}(x-\theta_p)^2$ for all p.

Note that step 5 of Algorithm 1 contains the (closed-form) solution of the problem that each node has to solve (see (18)):

$$x_p^{k+1} = \arg\min_{x_p}\frac{1}{2}(x_p-\theta_p)^2 + v_p^k x_p + \frac{\rho D_p}{2}x_p^2\,.$$

Also, the update of the dual variables corresponding to step 9 was simplified. The reason is because node p only requires the sum of the $\lambda_{\{i,j\}}$'s associated to its edges; see the definition of γ_p^k. Therefore, if we replace the update $\lambda_{\{i,j\}}^{k+1} = \lambda_{\{i,j\}}^k + \rho\,\text{sign}(j-i)(x_i^{k+1} - x_j^{k+1})$ into the definition of γ_p^k we obtain the expression of step 9.

Algorithm 1 is asynchronous in the sense that nodes operate in a color-based order. First, the nodes with color 1 perform steps 4 and 5 and obtain new estimates x_p^{k+1}, which are immediately sent to their neighbors. Next, nodes with color 2 repeat the same tasks, and so on. Although in Algorithm 1 all nodes with the same color operate at the same time, in practice they do not need to, assuming each node knows the colors of its neighbors. In that case, at a given iteration, once a node has received the estimates from all the neighbors with lower color, it has all the information for computing a new estimate. According to the same reasoning, step 9 also need not be carried out in parallel: if a given node has received the estimates x_j^{k+1} from all its neighbors, it can update γ_p independently of the other nodes. The conclusion is that, provided the nodes know the colors of their neighbors, a color-based operation can be carried out without any global coordination.

Convergence guarantees. Algorithm 1 is guaranteed to converge due to the recent result in [18], which states that if all the functions in (11) are strongly convex, then the generalized ADMM iterations (12)-(14) solve (11). In our case, each g_c in (11) is given by $\sum_{p\in\mathcal{C}_c} g_p(x_p) = \frac{1}{2}\sum_{p\in\mathcal{C}_p}(x_p - \theta_p)^2$, which is strongly convex. The convergence is then assured by the equivalence between (1) and (5).

III. Simulation Results

We now provide experimental comparison of Algorithm 1 with [1], [7], which are known to be state-of-the-art for fast consensus. We leave [6] out because it performs slower than [7] in the scenarios considered here (noise-free).

Performance measure: communication steps. We will compare the algorithms using the measure *communications steps*. We say that a communication step occurred after all the nodes in the network have updated their estimates and exchanged them with their neighbors. Since all the algorithms we compare consist of a single loop containing these operations, the number of communication steps will be equal to the number of iterations. However, we use the term "communication steps" instead of "number of iterations" because one iteration of Algorithm 1 may take longer than one iteration of either [1] or [7]. The reason is because Algorithm 1 is asynchronous, while [1] and [7] are synchronous, i.e., all nodes perform at the same time. Hence, in networks that allow all nodes to communicate at the same time, an iteration of [1], [7] is faster than an iteration of Algorithm 1. We do not take into account the cost of coloring the network for two reasons. First, the coloring can be done offline before any data arrives, and it needs to be done only once. Second, due to packet collisions, no algorithm can be implemented in a wireless network without using a MAC protocol [19, Ch.6]. A schedule-based MAC protocol, for example TDMA or FDMA, requires computing an interference scheme of the network beforehand; an interference scheme is a valid coloring scheme and thus can be used in Algorithm 1. Contention-based MAC protocols, on the other hand, make any synchronous algorithm become asynchronous, besides being less energy-efficient.

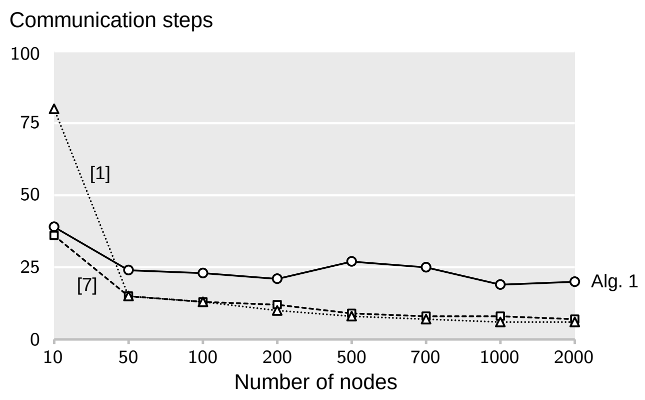
TABLE I
Network parameters

Number	Model	Parameters
1	Erdős-Rényi	0.25
2	Erdős-Rényi	0.75
3	Watts-Strogatz	$(2, 0.8)$
4	Watts-Strogatz	$(4, 0.6)$
5	Barabasi-Albert	——
6	Geometric	0.2
7	Lattice	——

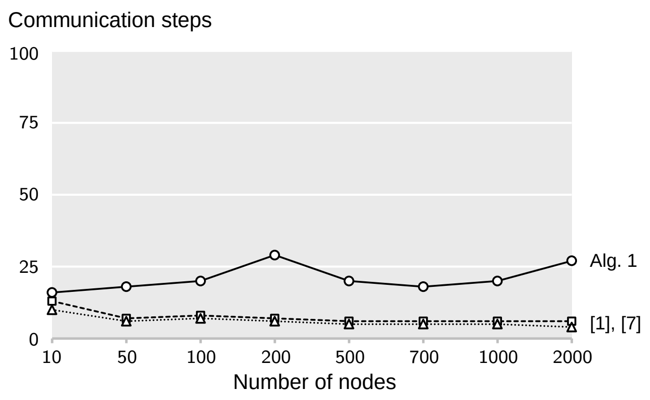
Network models. To compare all algorithms, we generated networks according to 7 different models, shown in Table I. The used models and the role of its parameters are explained in Table II. For each network in Table I, we generated 8 networks of different sizes, ranging from 10 to 2000 nodes. The values for the parameters shown in Table I represent just an average, since every time we created a non-connected network, we would create another with the parameters slightly changed in the direction to make the new network connected with larger probability.

Experimental setting. For each network realization, we drew θ_p randomly and independently from a Gaussian distribution with mean 10 and standard deviation 100. We chose ρ in Algorithm 1 from the set $\{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 0.1, 0.5, 1, 5, 10, 50, 10^2\}$, since there is not a simple way of choosing the best ρ for an ADMM-based algorithm.[1] Therefore, for each network configuration

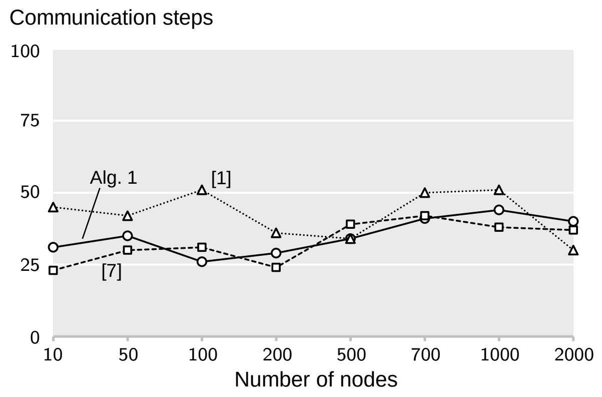
[1]An exception is [6]. In fact, analyzing Algorithm 1 within the same framework as [6] seems to be a promising direction of research.



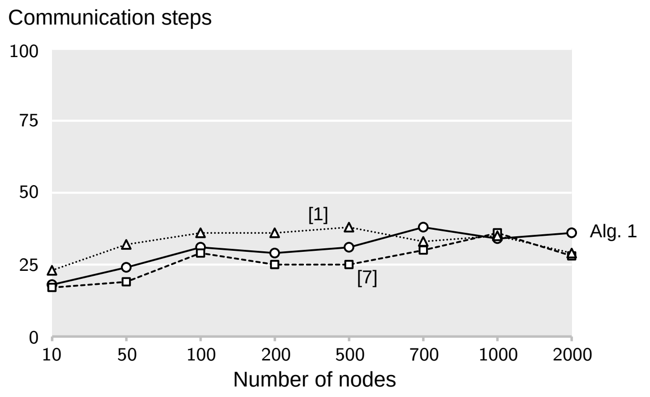
(a) Network 1: Erdős-Rényi, $p = 0.25$



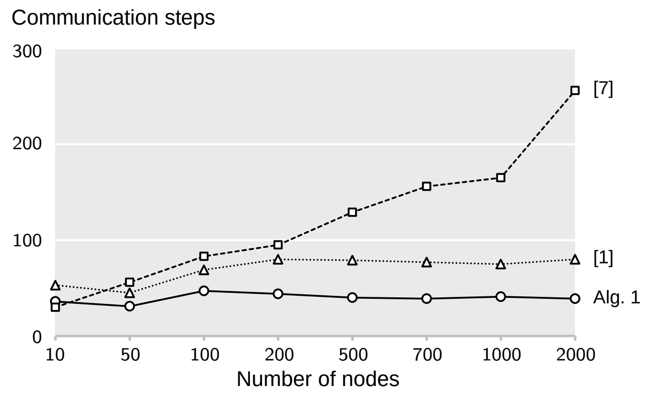
(b) Network 2: Erdős-Rényi, $p = 0.75$



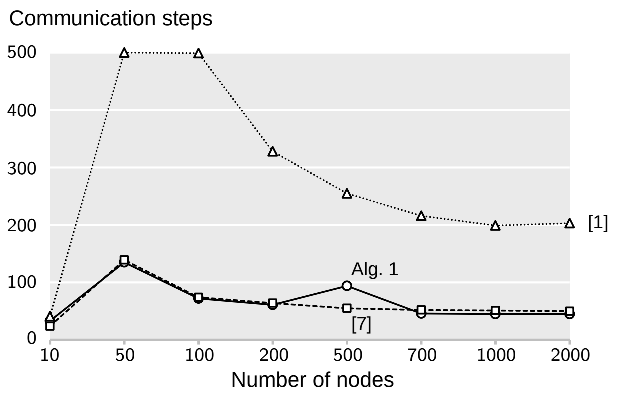
(c) Network 3: Watts-Strogatz, $(n, p) = (2, 0.8)$



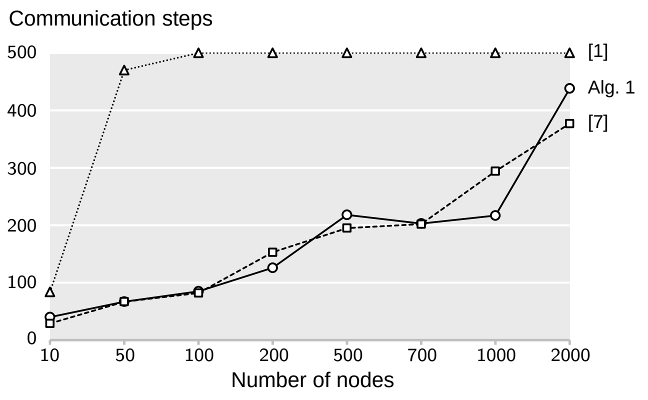
(d) Network 4: Watts-Strogatz, $(n, p) = (4, 0.6)$



(e) Network 5: Barabasi



(f) Network 6: Geometric, $d = 0.2$



(g) Network 7: Lattice

Fig. 2. Number of communication steps to achieve a $10^{-2}\%$ precision as a function of the number of nodes. The networks were generated according to Table I. Note that (e), (f) and (g) have a scale for the vertical axis different from the other plots.

TABLE II

NETWORK MODELS

Name	Parameters	Description
Erdős-Rényi	p	Every pair of nodes $\{i, j\}$ is connected or not with probability p
Watts-Strogatz	(n, p)	First, it creates a lattice where every node is connected to n nodes; then, it rewires every link with probability p. If link $\{i, j\}$ is to be rewired, it removes the link, and connects node i or node j (chosen with equal probability) to another node in the network, chosen uniformly.
Barabasi-Albert	——	It starts with one node. At each step, one node is added to the network by connecting it to 2 existing nodes: the probability to connect it to node p is proportional to D_p.
Geometric	d	It drops P points, corresponding to the nodes of the network, randomly in a $[0, 1]^2$ square; then, it connects nodes whose (Euclidean) distance is less than d.
Lattice	——	Creates a lattice of dimensions $m \times n$; m and n are chosen to make the lattice as square as possible.

we ran Algorithm 1 for all the ρ's and chose the best result.

Regarding algorithm [1], we ran its fastest version, under the equal neighbor model, which assumes the sum of all nodes' degrees, $\sum_{p=1}^{P} D_p$, is known by all the nodes. For algorithm [7], we chose the optimal value for a parameter on which it depends, which is a function of the second largest eigenvalue of the weighing matrix; as in the simulations of [7], we used a Metropolis-Hastings weighing matrix.

All algorithms stopped after achieving either a $10^{-2}\%$ solution accuracy, i.e., $\|x^k - 1_P\theta^\star\|/|\sqrt{P}\theta^\star| \leq 10^{-4}$, or the maximum number of 500 communication steps.

Results. Fig. 2 shows the results of our experiments. Each plot depicts the number of communication steps as a function of the number of nodes in the network. It can be seen that all algorithms require about the same number of iterations in the networks of Figs. 2(a), 2(b), 2(c), and 2(d) and, curiously, that number seems to be independent of the size of the network. While our Algorithm 1 performed worse than the other algorithms for Erdős-Rényi networks (Figs. 2(a) and 2(b)), it performed best in Barabasi networks (Fig. 2(e)). For the remaining networks, the behavior of the proposed algorithm is similar to the best of either [1] or [7]. In particular, for Watts-Strogatz networks with parameters $(2, 0.8)$, Fig. 2(c), Algorithm 1 required the least number of communication steps in 40% of the cases, the same percentage as [7]. For the other type of Watts-Strogatz networks, i.e., Fig. 2(d), Algorithm 1 was the best only once, while [7] was the best for the remaining scenarios. This was exactly the opposite case of Fig. 2(e), where Algorithm 1 required more communications than [7] only for the smallest network. Regarding Figs. 2(f) and 2(g), Algorithm 1 was the best, respectively, in 75% and 29% of the cases.

Overall, we can say that Algorithm 1 has a performance very similar to [7], and slightly better than [1]. Moreover, from the plots of Fig. 2 we can conclude that our Algorithm 1, among all, exhibits a near-optimal (or optimal) performance for all tested network types.

IV. CONCLUSIONS

We proposed a new algorithm for the average consensus problem. The algorithm resulted from applying ADMM to an optimization problem formulation of consensus, and its convergence is guaranteed by recent results coming from the optimization area. The proposed algorithm assumes a network coloring is available, which is realistic in many networks. We compared our algorithm with the fastest consensus algorithms and concluded that, while having a performance similar to those algorithms, the proposed algorithm appears to be the stablest across different network models.

REFERENCES

[1] A. Olshevsky and J. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Review*, vol. 53, no. 4, 2011.

[2] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, 2010.

[3] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, 2007.

[4] M. DeGroot, "Reaching a consensus," *J. American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[5] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proc. IPSN'04*, 2004, pp. 20–27.

[6] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," *IEEE Trans. Sig. Proc.*, vol. 59, no. 11, 2011.

[7] 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, 2010.

[8] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Distributed basis pursuit," *IEEE Trans. Sig. Proc.*, vol. 60, no. 4, 2012.

[9] ——, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," 2012, submitted, [Online] http://arxiv.org/abs/1202.2805.

[10] S. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Netw.*, vol. 16, pp. 985–997, 2010.

[11] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Info. Th.*, vol. 52, no. 6, 2006.

[12] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *IEEE Conf. Dec. Control*, vol. 5, 2003.

[13] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Tran. Aut. Contr.*, vol. 54, no. 1, 2009.

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

[15] 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.

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

[17] 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. Learning*, vol. 3, no. 1, 2011.

[18] D. Han and X. Yuan, "A note on the alternating direction method of multipliers," *J. Optim. Theory Appl.*, vol. 152, 2012.

[19] B. Krishnamachari, *Networking Wireless Sensors*. Cambridge University Press, 2005.