A RANDOMIZED DUAL CONSENSUS ADMM METHOD FOR MULTI-AGENT DISTRIBUTED OPTIMIZATION

Tsung-Hui Chang

Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology, Taipei, Taiwan 10607
E-mail: tsunghui.chang@ieee.org

ABSTRACT

Recently, the alternating direction method of multipliers (ADMM) has been used for distributed consensus optimization and is shown to converge faster than conventional approaches based on consensus subgradient. In this paper, we consider a convex optimization problem with a linearly coupled equality constraint and employ a dual consensus ADMM (DC-ADMM) method for solving the problem in a fully distributed fashion. In particular, by considering a non-ideal network where the agents can be ON and OFF randomly and the communications among agents can fail probabilistically, we propose a randomized DC-ADMM method that is robust against these non-ideal effects. Moreover, we show that the proposed randomized method is provably convergent to an optimal solution and has a worst-case $\mathcal{O}(1/k)$ convergence rate, where k is the iteration number. Simulation results are presented to examine the practical convergence behavior of the proposed method in the presence of randomly ON/OFF agents and non-ideal communication links.

Index Terms— Distributed consensus optimization, multi-agent network, ADMM, randomized optimization

1. INTRODUCTION

Multi-agent distributed optimization [1] has drawn significant attention in recent years due to the need for large-scale signal processing and machine learning tasks over data networks [2]. In particular, the distributed agents (e.g., data servers) may collaboratively solve a data regression or learning problem by using only locally collected data since moving these data over the network may not be always feasible, especially when the data size is extremely large (e.g., big data). Many of the signal processing and machine learning problems can be formulated as the following optimization problem

$$(\mathsf{P}) \quad \min_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\in\mathbb{R}^L} \ \sum_{i=1}^N \phi_i(\boldsymbol{x}_i) \ \text{ s.t. } \ \sum_{i=1}^N \mathbf{E}_i\boldsymbol{x}_i = \boldsymbol{q}, \qquad (1)$$

where $\phi_i : \mathbb{R}^L \to \mathbb{R}\cup\{\infty\}$ is a local cost function (possibly nonsmooth and with extended values) associated with agent i, $\boldsymbol{x}_i \in \mathbb{R}^L$ is the local regression/decision variable, $\mathbf{E}_i \in \mathbb{R}^{M\times L}$ contains the locally observed data and $\boldsymbol{q} \in \mathbb{R}^M$ is a parametric vector which is assumed to be known by all agents. Applications of (P) include the basis pursuit (BP), logistic regression and LASSO problems in machine learning [3], the network flow control problem [4], smart grid control problem [5] and interference management problem in communications [6], to name a few. In the context of multi-agent optimization, it is assumed that the agents can access the local information (i.e., ϕ_i, $\mathbf{E}_i$ and $\boldsymbol{q}$) and exchange messages with neighbors only, but they want to globally solve the coupled problem (P).

Various distributed optimization methods have been proposed in the literature for solving problems with the same form as (P). For example, the consensus-based primal-dual subgradient methods [7, 8] can be employed to handle (P). These types of methods are simple and can handle a wide range of application problems, but the convergence rate is slow in general. Recently, the alternating direction method of multipliers (ADMM) [9, 10] has been used for fast distributed consensus optimization [11–18]. Specifically, the work [11] proposed a consensus ADMM (C-ADMM) method for solving a distributed LASSO problem. The works in [12, 13] proposed several distributed ADMM (D-ADMM) methods for solving problems with the same form as (P). However, the D-ADMM methods require each agent either to update the variables sequentially (not in parallel) or to solve a min-max (saddle point) subproblem at each iteration. Besides, it is also assumed that certain coloring scheme is available to the network graph. To overcome these issues, in [14], the authors proposed a distributed optimization method, called dual consensus ADMM (DC-ADMM), which solves (P) in a fully parallel manner over arbitrary networks as long as the network graph is connected.

Recently, there is an increasing interest in asynchronous distributed ADMM methods; see [16–18]. While the methods in [17, 18] require certain restrictive assumption on the network topology, the work [16] proposed a randomized ADMM scheme that works for any connected network graph. In this paper, we adopt the randomized strategy in [16] and propose a randomized DC-ADMM method that can solve (P) globally even in the presence of randomly ON/OFF agents and imperfect communication links. In particular, only agents that are active at the iteration update the local variable and exchange messages with neighboring active agents. We show that the proposed randomized DC-ADMM method converges to an optimal solution of (P) in the mean and has a worst-case $\mathcal{O}(1/k)$ convergence rate, where k is the iteration number. The presented simulation results show that the proposed randomized method is robust and converges consistently in the considered non-ideal network scenarios.

2. NETWORK MODEL AND ASSUMPTIONS

Assume that there are N agents in the network and that the network is modeled as a undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Here $\mathcal{V} = \{1, \ldots, N\}$ is the set of agents and $\mathcal{E}$ is the set of edges, i.e., $(i, j) \in \mathcal{E}$ if and only if agent i and agent j are neighbors and can communicate with each other. For each agent i, we define $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ as the index subset of neighbors.

This work is supported by Ministry of Science and Technology, Taiwan (R.O.C.), under grants NSC 102-2221-E-011-005-MY3.

To develop the proposed methods, we make the following assumptions on the network and problem (P).

Assumption 1 (a) *The undirected graph $\mathcal{G}$ is connected;* **(b)** *ϕ_i's are proper closed convex functions; there is no duality gap between* (P) *and its Lagrange dual (i.e., Slater condition); moreover, the minimum of* (P) *is attained and so is its optimal dual value.*

Assumption 1(a) implies that neighbor-wise variable consensus can infer global consensus in the network. Hence, consensus optimization can be achieved via neighbor-wise message exchange only. Assumption 1(b) implies that (P) is a convex optimization problem and both the primal and dual optimal solutions are achievable.

3. REVIEW OF DC-ADMM

DC-ADMM is recently proposed in [14] for solving (P) in a fully distributed manner. The idea of DC-ADMM is based on the Lagrange dual optimization and the consensus ADMM method [11]. To illustrate this method, let us consider the following problem which is equivalent to the Lagrange dual problem of (P)

$$\min_{\boldsymbol{y}\in\mathbb{R}^M} \sum_{i=1}^N \left(\varphi_i(\boldsymbol{y}) + \frac{1}{N}\boldsymbol{y}^T\boldsymbol{q}\right), \tag{2}$$

where $\boldsymbol{y}\in\mathbb{R}^M$ denotes the Lagrange dual variable associated with the linear equality constraint $\sum_{i=1}^N \mathbf{E}_i\boldsymbol{x}_i = \boldsymbol{q}$ in (1). In (2), functions $\varphi_i(\boldsymbol{y})$ are given by

$$\varphi_i(\boldsymbol{y}) \triangleq \max_{\boldsymbol{x}_i}\left\{ -\phi_i(\boldsymbol{x}_i) - \boldsymbol{y}^T\mathbf{E}_i\boldsymbol{x}_i\right\} \forall\, i\in\mathcal{V}. \tag{3}$$

In the standard dual decomposition method [19], the subproblems in (3) can be solved by the agents individually provided that $\boldsymbol{y}$ is given. However, the dual variable $\boldsymbol{y}$ is not distributed.

To achieve a fully distributed algorithm, one allows each agent i to have a local copy of the variable $\boldsymbol{y}$, denoted by $\boldsymbol{y}_i$, while enforcing the distributed $\boldsymbol{y}_i$'s to be the same across the network through neighbor-wise consensus constraints. This is equivalent to reformulating (2) as the following problem

$$\min_{\substack{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N\\ \{\boldsymbol{t}_{ij}\}}} \sum_{i=1}^N \left(\varphi_i(\boldsymbol{y}_i) + \frac{1}{N}\boldsymbol{y}_i^T\boldsymbol{q}\right) \tag{4a}$$

$$\text{s.t. } \boldsymbol{y}_i = \boldsymbol{t}_{ij}\ \forall j\in\mathcal{N}_i,\ i\in\mathcal{V}, \tag{4b}$$

$$\boldsymbol{y}_j = \boldsymbol{t}_{ij}\ \forall j\in\mathcal{N}_i,\ i\in V, \tag{4c}$$

where $\{\boldsymbol{t}_{ij}\}$ are slack variables. Constraints (4a) and (4b) ensure the neighbor-wise consensus, i.e., $\boldsymbol{y}_i = \boldsymbol{y}_j\ \forall j\in\mathcal{N}_i,\ i\in\mathcal{V}$. Under Assumption 1(a), neighbor-wise consensus is equivalent to the global consensus; as a result, (4) is equivalent to (2). Next, let us employ ADMM [9] to solve (4). It can be shown that [14] the ADMM steps for solving (4) are given by: for iteration $k = 1, 2, \cdots$

$$\boldsymbol{y}_i^k = \arg\min_{\boldsymbol{y}_i}\left\{\varphi_i(\boldsymbol{y}_i) + \frac{1}{N}\boldsymbol{y}_i^T\boldsymbol{q} + \sum_{j\in\mathcal{N}_i}(\boldsymbol{u}_{ij}^{k-1} + \boldsymbol{v}_{ji}^{k-1})^T\boldsymbol{y}_i + c\sum_{j\in\mathcal{N}_i}\|\boldsymbol{y}_i - \boldsymbol{t}_{ij}^{k-1}\|_2^2\right\}\forall i\in\mathcal{V}, \tag{5}$$

$$\boldsymbol{t}_{ij}^k = \boldsymbol{t}_{ji}^k = \frac{\boldsymbol{y}_i^k + \boldsymbol{y}_j^k}{2}\ \forall j\in\mathcal{N}_i,\ i\in\mathcal{V}, \tag{6}$$

$$\boldsymbol{u}_{ij}^k = \boldsymbol{u}_{ij}^{k-1} + c(\boldsymbol{y}_i^k - \boldsymbol{t}_{ij}^k)\ \forall j\in\mathcal{N}_i,\ i\in\mathcal{V}, \tag{7}$$

$$\boldsymbol{v}_{ji}^k = \boldsymbol{v}_{ji}^{k-1} + c(\boldsymbol{y}_i^k - \boldsymbol{t}_{ji}^k)\ \forall j\in\mathcal{N}_i,\ i\in\mathcal{V}, \tag{8}$$

where $\boldsymbol{u}_{ij}\in\mathbb{R}^M$, $\boldsymbol{v}_{ij}\in\mathbb{R}^M$ are the Lagrange dual variables associated with each of the constraints in (4b) and (4c), respectively, and $c>0$ is a penalty parameter. Equations (5) and (6) involve updating the primal variables of (4) in a one-round Gauss-Seidel fashion; while (7) and (8) update the dual variables by subgradient ascent.

Notice that subproblem (5) is a strongly convex problem. However, it is not easy to handle because subproblem (5) is in fact a min-max (saddle point) problem (see the definition of φ_i in (3)). Fortunately, by applying the minimax theorem [20, Proposition 2.6.2] and exploiting the strong convexity of (5) with respect to $\boldsymbol{y}_i$, one can avoid solving the min-max problem (5) directly. As shown in [14], subproblem (5) can be obtained in closed-form as follows

$$\boldsymbol{y}_i^k = \tfrac{1}{2|\mathcal{N}_i|}\big(2\textstyle\sum_{j\in\mathcal{N}_i}\boldsymbol{t}_{ij}^{k-1} - \frac{1}{c}\sum_{j\in\mathcal{N}_i}(\boldsymbol{u}_{ij}^{k-1} + \boldsymbol{v}_{ji}^{k-1}) + \tfrac{1}{c}(\mathbf{E}_i\boldsymbol{x}_i^k - \tfrac{1}{N}\boldsymbol{q})\big), \tag{9}$$

where $\boldsymbol{x}_i^k$ is given by an solution to the following quadratic program

$$\boldsymbol{x}_i^k = \arg\min_{\boldsymbol{x}_i}\left\{\phi_i(\boldsymbol{x}_i) + \frac{c}{4|\mathcal{N}_i|}\big\|\tfrac{1}{c}(\mathbf{E}_i\boldsymbol{x}_i - \tfrac{1}{N}\boldsymbol{q}) - \tfrac{1}{c}\textstyle\sum_{j\in\mathcal{N}_i}(\boldsymbol{u}_{ij}^{k-1} + \boldsymbol{v}_{ji}^{k-1}) + 2\sum_{j\in\mathcal{N}_i}\boldsymbol{t}_{ij}^{k-1}\big\|_2^2\right\}. \tag{10}$$

Note that subproblem (10) is a convex problem under Assumption 1(b). Moreover, subproblem (10) is easier to handle than (5) as standard convex solvers or simple algorithms can be directly applied.

Finally, by substituting (6) into (7)-(10) and by letting

$$\boldsymbol{p}_i^k \triangleq \textstyle\sum_{j\in\mathcal{N}_i}(\boldsymbol{u}_{ij}^k + \boldsymbol{v}_{ji}^k)\ \forall i\in\mathcal{V}, \tag{11}$$

for all k, the steps from (7) to (10) can be equivalently written as (12), (13) and (14) in Algorithm 1.

It is worthwhile to note that, in step (14), each agent i has to exchange $\boldsymbol{y}_i^k$ with its neighbors at each iteration k. Given $\{\boldsymbol{y}_j^{k-1}\}_{j\in\mathcal{N}_i}$ at each iteration k, the updates of $\boldsymbol{x}_i$ in (12) and $\boldsymbol{y}_i$ in (13) are independent of other agents. Therefore, unlike the D-ADMM methods in [12, 13], the DC-ADMM method in Algorithm 1 is fully parallel. It has been shown in [14] that, under Assumption 1, the iterates $\{\boldsymbol{x}_i^k\}_{i=1}^N$ in DC-ADMM are guaranteed convergent to an optimal solution of (P).

4. PROPOSED RANDOMIZED DC-ADMM

The DC-ADMM method in Algorithm 1 has assumed that all agents in the network are active and communications between the agents are errorless at every iteration k. In this section, we develop an randomized DC-ADMM method which is applicable to networks with randomly ON/OFF agents and non-ideal communication links. Before presenting the proposed method, we should remark here that this randomized approach is motivated by the recent work [16] which considers a randomized ADMM method. However, the problem considered therein is different from our problem (P) and their analysis results are not applicable to our randomized DC-ADMM method to be presented shortly.

We consider a random network as follows. At each iteration (e.g., time epoch), each agent has a probability, say $\alpha_i\in(0,1]$, to be ON (active), and probability $(1-\alpha_i)$ to be OFF. In addition, for each link $(i,j)\in\mathcal{E}$, there is a probability $p_{ij}\in(0,1]$ that agent i and agent j cannot successfully exchange messages due to,

Algorithm 1 DC-ADMM for solving (P) [14]

1: **Given** initial variables $\boldsymbol{x}_i^{(0)} \in \mathbb{R}^L$, $\boldsymbol{y}_i^{(0)} \in \mathbb{R}^L$ and $\boldsymbol{p}_i^{(0)} = \boldsymbol{0}$ for each agent $i \in \mathcal{V}$. Set $k = 1$.
2: **repeat**
3: For all $i \in \mathcal{V}$ (in parallel),

$$\boldsymbol{x}_i^k = \arg\min_{\boldsymbol{x}_i} \Big\{ \phi_i(\boldsymbol{x}_i) + \frac{c}{4|\mathcal{N}_i|} \big\| \tfrac{1}{c}(\mathbf{E}_i \boldsymbol{x}_i - \tfrac{1}{N}\boldsymbol{q}) - \tfrac{1}{c}\boldsymbol{p}_i^{k-1} + \textstyle\sum_{j\in\mathcal{N}_i}(\boldsymbol{y}_i^{k-1} + \boldsymbol{y}_j^{k-1}) \big\|_2^2 \Big\}, \quad (12)$$

$$\boldsymbol{y}_i^k = \tfrac{1}{2|\mathcal{N}_i|}\big(\textstyle\sum_{j\in\mathcal{N}_i}(\boldsymbol{y}_i^{k-1} + \boldsymbol{y}_j^{k-1}) - \tfrac{1}{c}\boldsymbol{p}_i^{k-1} + \tfrac{1}{c}(\mathbf{E}_i \boldsymbol{x}_i^k - \tfrac{1}{N}\boldsymbol{q})\big), \quad (13)$$

$$\boldsymbol{p}_i^k = \boldsymbol{p}_i^{k-1} + c \textstyle\sum_{j\in\mathcal{N}_i}(\boldsymbol{y}_i^k - \boldsymbol{y}_j^k). \quad (14)$$

4: **Set** $k \leftarrow k + 1$.
5: **until** a predefined stopping criterion is satisfied.

e.g., deep fading. So, the probability that agent i and agent j are both active and able to successfully exchange messages is given by $\beta_{ij} \triangleq \alpha_i\alpha_j(1 - p_{ij})$. If this happens, we say that link $(i, j) \in \mathcal{E}$ is active at the iteration. For each iteration k, we let $\Omega^k \subseteq \mathcal{V}$ be the set of active agents and let $\Psi^k \subseteq \{(i, j) \in \mathcal{E} \,|\, i, j \in \Omega^k\}$ be the set of active edges.

Now let us recall the genuine DC-ADMM steps from (5) to (8). Then, at each iteration k in the random network, only the active agents perform local variable update. In particular, active agents $i \in \Omega^k$ update the variable $\boldsymbol{y}_i$ following (5) (i.e., (9) and (10)) whereas others keep $\boldsymbol{y}_i$ unchanged, i.e., $\boldsymbol{y}_i^k = \boldsymbol{y}_i^{k-1}\ \forall i \notin \Omega^k$. Since each active agent $i \in \Omega^k$ can only receive message $\boldsymbol{y}_j^k$ from neighbors in the set $\{j \in \mathcal{N}_i \mid (i, j) \in \Psi^k\}$, agent i would update $\{\boldsymbol{t}_{ij}\}_{j\in\mathcal{N}_i}$ and $\{\boldsymbol{u}_{ij}, \boldsymbol{v}_{ji}\}_{j\in\mathcal{N}_i}$ as follows

$$\boldsymbol{t}_{ij}^k = \begin{cases} \frac{\boldsymbol{y}_i^k + \boldsymbol{y}_j^k}{2}, & \text{if } (i, j) \in \Psi^k, \\ \boldsymbol{t}_{ij}^{k-1}, & \text{otherwise,} \end{cases} \quad (15)$$

$$\boldsymbol{u}_{ij}^k = \begin{cases} \boldsymbol{u}_{ij}^{k-1} + c(\boldsymbol{y}_i^k - \boldsymbol{t}_{ij}^k) & \text{if } (i, j) \in \Psi^k, \\ \boldsymbol{u}_{ij}^{k-1}, & \text{otherwise,} \end{cases} \quad (16)$$

$$\boldsymbol{v}_{ji}^k = \begin{cases} \boldsymbol{v}_{ji}^{k-1} + c(\boldsymbol{y}_i^k - \boldsymbol{t}_{ji}^k) & \text{if } (i, j) \in \Psi^k, \\ \boldsymbol{v}_{ji}^{k-1}, & \text{otherwise.} \end{cases} \quad (17)$$

As seen, only the variables corresponding to the active edges are updated accordingly, otherwise, the variables remain unchanged[1]. Note that, from (15), we still have

$$\boldsymbol{t}_{ij}^k = \boldsymbol{t}_{ji}^k\ \forall i, j, k. \quad (18)$$

By (18) and by letting $\boldsymbol{p}_i^k \triangleq \sum_{j\in\mathcal{N}_i}(\boldsymbol{u}_{ij}^k + \boldsymbol{v}_{ji}^k)\ \forall i \in \mathcal{V}$, (16) and (17) can be written as

$$\boldsymbol{p}_i^k = \boldsymbol{p}_i^{k-1} + 2c \textstyle\sum_{j|(i,j)\in\Psi^k}(\boldsymbol{y}_i^k - \boldsymbol{t}_{ij}^k)\ \forall i \in \mathcal{V}. \quad (19)$$

Finally, we summarize the steps of the proposed randomized DC-ADMM method in Algorithm 2.

[1]We have assumed that the agents are capable of detecting communication errors, e.g., via error correcting coding techniques. Moreover, if agent $i \in \Omega^k$ detects that an error occurs in the link from agent $j \in \Omega^k$, agent i would treat $(i, j) \notin \Psi^k$ and further sends a signaling to agent j. Agent j will also take $(j, i) \notin \Psi^k$ as it either receives the signaling from agent i or detects the link error by itself.

Algorithm 2 Randomized DC-ADMM for solving (P)

1: **Given** initial variables $\boldsymbol{x}_i^0 \in \mathbb{R}^L$, $\boldsymbol{y}_i^0 \in \mathbb{R}^L$, $\boldsymbol{p}_i^0 = \boldsymbol{0}$ and

$$\boldsymbol{t}_{ij}^0 = \tfrac{\boldsymbol{y}_i^0 + \boldsymbol{y}_j^0}{2}\ \forall j \in \mathcal{N}_i,$$

for each agent $i \in \mathcal{V}$. Set $k = 1$.
2: **repeat**
3: For all $i \in \Omega^k$ (in parallel),

$$\boldsymbol{x}_i^k = \arg\min_{\boldsymbol{x}_i} \Big\{ \phi_i(\boldsymbol{x}_i) + \frac{c}{4|\mathcal{N}_i|} \big\| \tfrac{1}{c}(\mathbf{E}_i \boldsymbol{x}_i - \tfrac{1}{N}\boldsymbol{q}) - \tfrac{1}{c}\boldsymbol{p}_i^{k-1} + 2\textstyle\sum_{j\in\mathcal{N}_i} \boldsymbol{t}_{ij}^{k-1} \big\|_2^2 \Big\}, \quad (20)$$

$$\boldsymbol{y}_i^k = \tfrac{1}{2|\mathcal{N}_i|}\big(2\textstyle\sum_{j\in\mathcal{N}_i} \boldsymbol{t}_{ij}^{k-1} - \tfrac{1}{c}\boldsymbol{p}_i^{k-1} + \tfrac{1}{c}(\mathbf{E}_i \boldsymbol{x}_i^k - \tfrac{1}{N}\boldsymbol{q})\big), \quad (21)$$

$$\boldsymbol{t}_{ij}^k = \begin{cases} \frac{\boldsymbol{y}_i^k + \boldsymbol{y}_j^k}{2}, & \text{if } (i, j) \in \Psi^k, \\ \boldsymbol{t}_{ij}^{k-1}, & \text{otherwise,} \end{cases} \quad (22)$$

$$\boldsymbol{p}_i^k = \boldsymbol{p}_i^{k-1} + 2c \textstyle\sum_{j|(i,j)\in\Psi^k}(\boldsymbol{y}_i^k - \boldsymbol{t}_{ij}^k); \quad (23)$$

whereas for all $i \notin \Omega^k$ (in parallel)

$$\boldsymbol{x}_i^k = \boldsymbol{x}_i^{k-1},\ \boldsymbol{y}_i^k = \boldsymbol{y}_i^{k-1}, \boldsymbol{p}_i^k = \boldsymbol{p}_i^{k-1}, \quad \boldsymbol{t}_{ij}^k = \boldsymbol{t}_{ij}^{k-1}\ \forall j \in \mathcal{N}_i. \quad (24)$$

4: **Set** $k \leftarrow k + 1$.
5: **until** a predefined stopping criterion is satisfied.

There are two key differences between the proposed randomized DC-ADMM method and the original DC-ADMM method in Algorithm 1. Firstly, in addition to $(\boldsymbol{x}_i^k, \boldsymbol{y}_i^k, \boldsymbol{p}_i^k)$, each agent i in randomized DC-ADMM also requires to keep variables $\{\boldsymbol{t}_{ij}, j \in \mathcal{N}_i\}$ explicitly. Secondly, randomized DC-ADMM takes into account possibly inactive agents and inactive communication links because variables $(\boldsymbol{x}_i^k, \boldsymbol{y}_i^k, \boldsymbol{p}_i^k)$ are updated only if $i \in \Omega^k$ and variables $(\boldsymbol{t}_{ij}^k, \{\boldsymbol{u}_{ij}^k, \boldsymbol{v}_{ji}^k\})$ are updated only if $(i, j) \in \Psi^k$. Since the randomized DC-ADMM method coincides with the DC-ADMM method in Algorithm 1 when $\Omega^k = \mathcal{V}$ and $\Psi^k = \mathcal{E}$ for all k, the former is a generalization of the latter to the random networks.

Interestingly, the randomized DC-ADMM method in Algorithm 2 is still provably convergent, as stated in the following theorem.

Theorem 1 *Suppose that Assumption 1 holds. Besides, assume that each agent i has an active probability $\alpha_i \in (0, 1]$ and each $(i, j) \in \mathcal{E}$ has a link failure probability $p_{ij} \in (0, 1]$. Let $(\{\boldsymbol{x}_i^\star\}_{i=1}^N, \boldsymbol{y}^\star)$ be a pair of optimal primal-dual solution of* (P) *, and let $\{\boldsymbol{u}_{ij}^\star\}$ be the optimal dual variables of problem* (4)*. Moreover, let*

$$\bar{\boldsymbol{x}}_i^k \triangleq \frac{1}{k}\sum_{\ell=0}^{k-1} \boldsymbol{x}_i^\ell\ \ \forall i \in \mathcal{V}, \quad (25)$$

where $\{\boldsymbol{x}_i^k\}_{i=1}^N$ are generated by Algorithm 2. Then, it holds that

$$\left| \mathbb{E}\Big[\sum_{i=1}^N \phi_i(\bar{\boldsymbol{x}}_i^k) - \sum_{i=1}^N \phi_i(\boldsymbol{x}_i^\star) \Big] \right| + \left\| \mathbb{E}\Big[\sum_{i=1}^N \mathbf{E}_i \bar{\boldsymbol{x}}_i^k - \boldsymbol{q} \Big] \right\|_2 \le \frac{(1 + \|\boldsymbol{y}^\star\|_2)C_1 + C_2}{k},$$

where C_1 and C_2 are some positive constants that depend on $\{\alpha_i\}$, $\{\beta_{ij}\}$, $\boldsymbol{y}^\star$ and $\{\boldsymbol{u}_{ij}^\star\}$

Due to space limitation, the proof is omitted here. Theorem 1 implies that the proposed randomized DC-ADMM method can converge to an optimal solution of (P) in the mean, with a $\mathcal{O}(1/k)$ worst-case convergence rate.

5. NUMERICAL RESULTS

In this section, we examine the numerical performances of the proposed randomized DC-ADMM method (Algorithm 2), by considering the following sparse logistic regression (LR) problem [14, 21]

$$\min_{\substack{\boldsymbol{x}_i\in\mathcal{X},\\ i=1,\ldots,N}} \left\{ \sum_{m=1}^{M} \log\left(1+\exp(-b_m \sum_{i=1}^{N} \boldsymbol{e}_{iq}^T \boldsymbol{x}_i)\right) + \lambda \sum_{i=1}^{N} \|\boldsymbol{x}_i\|_1 \right\}, \tag{26}$$

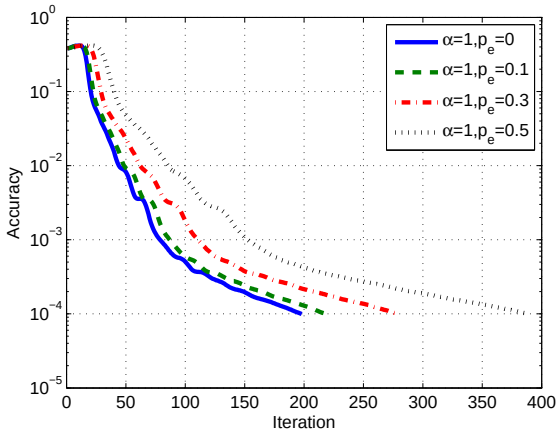
where $b_1, \ldots, b_M$ are the binary labels of the M training data, $\boldsymbol{x}_i \in \mathbb{R}^L$ is a local regression variable and $\mathbf{E}_i = [\boldsymbol{e}_{i1}, \ldots, \boldsymbol{e}_{iM}]^T \in \mathbb{R}^{M\times L}$ is a column-partitioned training data matrix owned by each agent i, for all $i \in \mathcal{V}$. By introducing a slack variable $\boldsymbol{z} = [z_1, \ldots, z_M]^T$ $\sum_{i=1}^N \mathbf{E}_i \boldsymbol{x}_i$, the LR problem can be reformulated as

$$\begin{aligned} \min_{\substack{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\in\mathcal{X},\\ \boldsymbol{z}\in\mathbb{R}^M}} & \left\{ \sum_{m=1}^{M} \log\left(1+\exp(-b_m z_m)\right) + \lambda \sum_{i=1}^{N} \|\boldsymbol{x}_i\|_1 \right\} \\ \text{s.t. } & \textstyle\sum_{i=1}^N \mathbf{E}_i \boldsymbol{x}_i - \boldsymbol{z} = \mathbf{0}, \end{aligned} \tag{27}$$

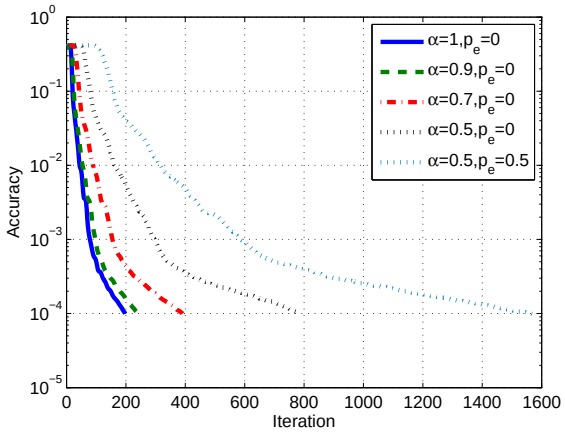
which is an instance of (P) and Algorithm 2 can be applied.

The training data $\mathbf{E}_i$'s and labels b_m's were generated following the same approach as in [14] based on the images D24 and D68 of the Brodatz data set (`http://www.ux.uis.no/~tranden/brodatz.html`). The feasible set $\mathcal{X}$ in (27) was set to $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^L \mid |x_i| \le 10\ \forall i\}$. To implement Algorithm 2, we employed the fast iterative shrinkage-thresholding algorithm (FISTA) [22] to solve subproblem (20). The stopping condition of FISTA is based on the proximal gradient residue which was set to 10^{-5}. The stopping criterion of Algorithms 2 is based on measuring the solution accuracy $\mathsf{acc} = (\mathsf{obj}(\boldsymbol{x}^k) - \mathsf{obj}^\star)/\mathsf{obj}^\star$, where $\mathsf{obj}(\boldsymbol{x}^k)$ denotes the objective value of (26) at point $\boldsymbol{x}^k = (\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_N^k)$, and $\mathsf{obj}^\star$ is the optimal value of (26) which was obtained by using FISTA. The connected graph $\mathcal{G}$ was randomly generated following the same method as in [23]. For simplicity, the active probabilities of agents are set the same $\alpha \triangleq \alpha_1 = \cdots = \alpha_N$ and the link failure probabilities of all edges are also set the same $p_e \triangleq p_{ij}\ \forall (i,j) \in \mathcal{E}$.

In Fig. 1, we present the convergence curves of the proposed randomized DC-ADMM method, for an example with $N = 50$, $L = 200$, $M = 100$ and $\lambda = 0.05$. The penalty parameter c was set to 0.05, and the stopping condition was set to $\mathsf{acc} < 10^{-4}$. In particular, Fig. 1(a) displays the convergence curves of randomized DC-ADMM for $\alpha = 1$ and for various values of p_e. Note that the curve of $\alpha = 1$ and $p_e = 0$ corresponds to the original DC-ADMM method (Algorithm 1). One can see from this figure that randomized DC-ADMM converges consistently, although the random link failure does slow down the convergence speed. Specifically, we observe that the number of iterations required to reach $\mathsf{acc} < 10^{-4}$ with $p_e = 0.5$ is roughly doubled compared to that with $p_e = 0$. This is reasonable as, with $p_e = 0.5$, the probability for two agents exchanging messages successfully (i.e., $\beta_{ij} = 1 - p_e$) is reduced half. So the network requires a doubled number of iterations to reach the same solution accuracy. Fig. 1(b) shows the results of randomized DC-ADMM for $p_e = 0$ and for various values of α. Similarly, one can observe that randomized DC-ADMM still converges consistently and the convergence speed is slowed due to randomly inactive agents. One can observe that, with $\alpha = 0.5$, the number of iterations required to reach $\mathsf{acc} < 10^{-4}$ is around four times of that for $\alpha = 1$. This is because the probability for two agents being ON simultaneously and exchange message successfully (i.e., $\beta_{ij} = \alpha^2$) is decreased to one fourth. Fig. 1(b) also shows the convergence curve of randomized DC-ADMM for $\alpha = 0.5$ and $p_e = 0.5$. One can see that randomized DC-ADMM still converges even in such a severe scenario.



(a)



(b)

Fig. 1. Convergence curves of randomized DC-ADMM.

6. CONCLUSIONS

We have proposed in this paper the randomized DC-ADMM method (Algorithm 2) for solving (P) over non-ideal networks with randomly ON/OFF agents and imperfect communication links. We have shown that the proposed randomized method converges to an optimal solution of (P) in the mean with a worst-case $\mathcal{O}(1/k)$ convergence rate. The simulation results have shown that the proposed randomized DC-ADMM method is robust against these non-ideal effects and always converges.

7. REFERENCES

[1] B. Yang and M. Johansson, "Distributed optimization and games: A tutorial overview," Chapter 4 of *Networked Control Systems*, A. Bemporad, M. Heemels and M. Johansson (eds.), LNCIS 406, Springer-Verlag, 2010.

[2] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up Machine Learning- Parallel and Distributed Approaches*. Cambridge University Press, 2012.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer-Verlag, 2001.

[4] D. P. Bertsekas, *Network Optimization : Contribuous and Discrete Models*. Athena Scientific, 1998.

[5] T.-H. Chang, M. Alizadeh, and A. Scaglione, "Coordinated home energy management for real-time power balancing," in *Proc. IEEE PES General Meeting*, San Diego, CA, July 22-26, 2012, pp. 1–8.

[6] C. Shen, T.-H. Chang, K.-Y. Wang, Z. Qiu, and C.-Y. Chi, "Distributed robust multicell coordianted beamforming with imperfect CSI: An ADMM approach," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2988–3003, 2012.

[7] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Trans. Auto. Control*, vol. 57, no. 1, pp. 151–164, Jan. 2012.

[8] T.-H. Chang, A. Nedić, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE. Trans. Auto. Control.*, vol. 59, no. 6, pp. 1524–1538, June 2014.

[9] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[11] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Dec. 2010.

[12] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE. Trans. Signal Process.*, vol. 60, no. 4, pp. 1942–1956, April 2012.

[13] ——, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE. Trans. Signal Process.*, vol. 60, no. 10, pp. 2718–2723, May 2013.

[14] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 482–497, Jan. 2015.

[15] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, April 2014.

[16] E. Wei and A. Ozdaglar, "On the $O(1/K)$ convergence of asynchronous distributed alternating direction method of multipliers," available on arxiv.org.

[17] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. 31th ICML, , 2014.*, Beijing, China, June 21-26, 2014, pp. 1–9.

[18] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *Proc. IEEE CDC*, Florence, Italy, Dec. 10-13, 2013, pp. 3671–3676.

[19] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley, "Notes on decomposition methods," avaliable at http://see.stanford.edu/materials/lsocoee364b/08-decomposition_notes.pdf.

[20] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex analysis and optimization*. Cambridge, Massachusetts: Athena Scientific, 2003.

[21] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Large-scale sparse logistic regression," in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, June 28 - July 1, 2009, pp. 547–556.

[22] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[23] M. E. Yildiz and A. Scaglione, "Coding with side information for rate-constrained consensus," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3753–3764, 2008.