COLA: COMMUNICATION-CENSORED LINEARIZED ADMM FOR DECENTRALIZED CONSENSUS OPTIMIZATION

Weiyu Li[*] *Yaohua Liu*[**] *Zhi Tian*[†] *Qing Ling*[‡]

[*] School of Gifted Young, University of Science and Technology of China
[**] Department of Automation, University of Science and Technology of China
[†] Department of Electrical and Computer Engineering, George Mason University
[‡] School of Data and Computer Science, Sun Yat-Sen University

ABSTRACT

This paper proposes a communication- and computation-efficient algorithm to solve a convex consensus optimization problem defined over a decentralized network. A remarkable existing algorithm to solve this problem is the alternating direction method of multipliers (ADMM), in which at every iteration every node updates its local variable through combining neighboring variables and solving an optimization subproblem. The proposed algorithm, called as communication-censored linearized ADMM (COLA), leverages a linearization technique to reduce the iteration-wise computation cost of ADMM and uses a communication-censoring strategy to alleviate the communication cost. To be specific, COLA introduces successive linearization approximations to the local cost functions such that the resultant computation is first-order and light-weight. Since the linearization technique slows down the convergence speed, COLA further adopts the communication-censoring strategy to avoid transmissions of less informative messages. A node is allowed to transmit only if the distance between the current local variable and its previously transmitted one is larger than a censoring threshold. We establish convergence as well as sublinear and linear rates of convergence of COLA, and demonstrate its satisfactory communication-computation tradeoff with numerical experiments.

Index Terms— Decentralized consensus optimization, alternating direction method of multipliers, communication censoring.

1. INTRODUCTION

This paper considers solving a consensus optimization problem

$$\tilde{x}^* = \arg\min_{\tilde{x}} \sum_{i=1}^{n} f_i(\tilde{x}), \tag{1}$$

which is defined over a bidirectionally connected decentralized network of n nodes. All the nodes cooperate to find an optimal argument $\tilde{x}^*$ of the common optimization variable $\tilde{x} \in \mathcal{R}^p$, but the local cost function $f_i(\tilde{x}) : \mathcal{R}^p \to \mathcal{R}$ held by every node i is kept private. We focus on the scenario that the nodes cannot afford complicated computation, while communication resources are also limited. Our goal is to devise a communication-efficient decentralized algorithm, which relies on light-weight computation, to solve (1).

Decentralized consensus optimization has attracted extensive interest in recent years. Problems in the form of (1) are involved in a variety of research areas, including wireless sensor networks [1], communication networks [2], multi-robot networks [3], smart grids [4], machine learning systems [5, 6], to name a few. Popular algorithms to solve (1) span from the primal domain to the dual domain. The primal domain algorithms, such as sub-gradient descent [7, 8, 9], dual averaging [10, 11, 12] and network Newton [13], have to use diminishing step sizes to guarantee exact convergence to an optimal solution, and thus suffer from slow convergence. On the other hand, (1) can be reformulated as a constrained optimization problem and solved by the dual domain algorithms, among which the celebrated alternating direction method of multipliers (ADMM) is able to achieve fast and exact convergence [1, 14, 15, 16]. When ADMM is implemented in a synchronous manner, at every iteration, every node solves an optimization subproblem dependent on its local cost function, and then exchanges the calculated local variable with its neighbors. Therefore, if the local cost functions are not in simple forms, solving the subproblems is computationally demanding. To alleviate the computation cost, the decentralized linearized ADMM (DLM) replaces the local cost functions in ADMM by their linear approximations, and attains a dual domain method with light-weight computation [17, 18]. Similar techniques have also been applied to develop other first-order dual domain algorithms, such as EXTRA [19] and NEXT [20]. If computing the inverse of a Hessian matrix is affordable at a node, one can replace the local cost functions by their quadratic approximations. The resultant second-order algorithms, DQM and ESOM, have faster convergence than their first-order counterparts [21, 22].

There is an essential communication-computation tradeoff in all decentralized algorithms [23, 24, 25, 26, 27]. An algorithm with light-weight iteration-wise computation generally needs more number of iterations, and in consequence more communication cost, to reach a target accuracy. For example, comparing with ADMM, DLM enjoys simple first-order computation, but suffers from relatively slow convergence speed and high computation cost. In this paper, we aim at achieving a favorable communication-computation tradeoff in a decentralized network, where the nodes are only able to afford first-order computation. The limitation on the computation power may come from that the nodes are equipped with cheap computing units in a wireless sensor network, or from that using higher-order information is prohibitively time-consuming for finding a high-dimensional solution in a machine learning system.

Given the constraint on the computation cost, we adopt the communication-censoring strategy to further save the communication cost. The basic idea of the communication-censoring strategy is to only allow transmissions of informative messages over the network. A simple yet powerful protocol is to prevent a node from transmitting a variable that is close to its previously transmitted one, where the "closeness" is determined by comparing the Euclidean distance with a predefined time-varying censoring threshold. The communication-censoring strategy is tightly related to event-triggered control of continuous-time networks [28, 29, 30], and finds successful applications in discrete-time decentralized optimiza-

tion. It has been combined with primal domain methods such as sub-gradient descent [31] and dual averaging [32], as well as dual domain methods such as dual decomposition [33] and ADMM [34]. However, similar to their uncensored counterparts, the primal domain methods have to use diminishing step sizes to guarantee exact convergence. On the other hand, the dual domain methods require the nodes to solve computationally demanding subproblems. Our proposed algorithm, called as communication-censored linearized ADMM (COLA), combines the communication-censoring strategy with the first-order dual domain method DLM. Particularly, we modify the standard communication-censoring strategy in [31,32,33,34] to fit for the special algorithmic structure of DLM so as to attain better performance. We rigorously establish the convergence as well as sublinear and linear convergence rates of COLA. To the best of our knowledge, COLA is the first communication-censored method that only uses gradient information but achieves linear convergence.

Notations. Throughout the paper, we consider a bidirectionally connected network $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$, where $\mathcal{V} = \{1, \ldots, n\}$ denotes the set of n nodes and $\mathcal{A} = \{1, \ldots, m\}$ is the set of m directed arcs. Note that m is even because the network is bidirectional. Nodes i and j are called as neighbors if $(i, j) \in \mathcal{A}$ and $(j, i) \in \mathcal{A}$. We denote the set of node i's neighbors as $\mathcal{N}_i$ with cardinality $d_{ii} = |\mathcal{N}_i|$. Further define the extended block arc source matrix $A_s \in \mathcal{R}^{mp\times np}$ containing $m \times n$ square blocks $(A_s)_{e,i} \in \mathcal{R}^{p\times p}$. The block $(A_s)_{e,i} = I_p$ if the arc $e = (i, j) \in \mathcal{A}$ and is null otherwise, where I_p is the p-dimensional identity matrix. Likewise, define the extended block arc destination matrix $A_d \in \mathcal{R}^{mp\times np}$, whose block $(A_d)_{e,j} \in \mathcal{R}^{p\times p}$ is not null but I_p if and only if the arc $e = (i, j) \in \mathcal{A}$ terminates at node j. Then, define the extended oriented incidence matrix as $G_o = A_s - A_d$ and the unoriented one as $G_u = A_s + A_d$.

2. ALGORITHM DEVELOPMENT

ADMM is a powerful tool to solve a structured optimization problem with two blocks of variables, which are separable in the cost function and subject to a linear equality constraint. To rewrite the (1) into the standard bivariate form, we introduce local variables $x_i \in \mathcal{R}^p$ as copies of $\tilde{x}$ at nodes i, and auxiliary variables $z_{ij} \in \mathcal{R}^p$ at arcs $(i, j) \in \mathcal{A}$. Since the network is connected, (1) is equivalent to

$$\min_{\{x_i\},\{z_{ij}\}} \sum_{i=1}^{n} f_i(x_i), \quad \text{s.t. } x_i = z_{ij},\ x_j = z_{ij},\ \forall (i, j) \in \mathcal{A}. \tag{2}$$

The optimal solution of (2) satisfies $x_i^* = \tilde{x}^*$ and $z_{ij}^* = \tilde{x}^*$, where $\tilde{x}^*$ is an optimal solution of (1).

Concatenating the variables as vectors $x = [x_1; \ldots; x_n] \in \mathcal{R}^{np}$ and $z = [z_1; \ldots; z_m] \in \mathcal{R}^{mp}$, and introducing the aggregate function $f(x) := \sum_{i=1}^{n} f_i(x_i)$, we rewrite (2) in the matrix form

$$\min_{x,z} f(x), \quad \text{s.t. } Ax + Bz = 0, \tag{3}$$

where $A := [A_s; A_d] \in \mathcal{R}^{2mp\times np}$ and $B := [-I_{mp}; -I_{mp}]$.

According to [16], we can solve (3) by ADMM and eliminate the update of z under mild initial conditions. For every node i, the resultant algorithm includes primal and dual updates

$$x_i^{k+1} = \arg\min_{x_i} f_i(x_i) + \langle \mu_i^k - c \sum_{j\in\mathcal{N}_i} (x_i^k + x_j^k), x_i\rangle + c d_{ii} x_i^2, \tag{4}$$

$$\mu_i^{k+1} = \mu_i^k + c \sum_{j\in\mathcal{N}_i} (x_i^{k+1} - x_j^{k+1}), \tag{5}$$

where $\mu_i^k \in \mathcal{R}^p$ is the local dual variable of node i.

The decentralized ADMM algorithm in (4) and (5) to solve (1) is implemented in a synchronous manner. At time k, every node i updates its local primal variable x_i^{k+1} using its x_i^k and μ_i^k, as well as x_j^k from all neighbors j. Then node i broadcasts its x_i^{k+1} to all neighbors. Finally, node i updates its local dual variable μ_i^{k+1} using its x_i^{k+1} and μ_i^k, as well as x_j^{k+1} from all neighbors j. The costs of implementing ADMM are two-fold. The first is in computing the local primal and dual variables x_i^k and μ_i^k, in which the update of x_i^k in (4) is demanding when the local cost function $f_i(x_i)$ is complicated. The second is in transmitting the local primal variables x_i^{k+1}, which is expensive when the bandwidth resource is limited.

COLA adopts two strategies to improve the computation and communication efficiency of ADMM: linearization and communication censoring. The linearization technique has been used in [17,18] to devise DLM, a gradient-based variant of ADMM. DLM effectively reduces the computational cost of solving subproblems in ADMM, but sacrifices on the convergence speed and thus results in high communication cost. Therefore, we use the communication-censoring strategy to prevent transmissions of less informative messages. Note that though the communication-censoring strategy has been applied to improve the communication efficiency of sub-gradient descent, dual averaging, dual decomposition and ADMM [31, 32, 33, 34], we customize it in COLA so as to achieve a satisfactory balance between communication and computation, as we shall explain below.

Linearization. Notice that the update of the primal variables x_i^{k+1} in (4) dominates the computation cost of ADMM. When $f_i(x_i)$ is not in a simple form such as linear or quadratic, (4) has no explicit solution. Therefore, a computationally demanding inner loop should be used to solve x_i^{k+1}. To address this issue, the DLM algorithm proposed in [17, 18] linearizes the local cost functions at every iteration. To be specific, at time k, the function $f_i(x_i)$ in (4) is replaced by its quadratic approximation $f_i(x_i^k) + \langle \nabla f_i(x_i^k), x_i - x_i^k\rangle + \frac{\rho}{2}\|x_i - x_i^k\|^2$ at $x_i = x_i^k$, where $\rho > 0$ is a positive linearization parameter, such that the update of x_i^{k+1} is replaced by

$$x_i^{k+1} = x_i^k - \frac{1}{2cd_{ii} + \rho}\left(\nabla f_i(x_i^k) + c \sum_{j\in\mathcal{N}_i} (x_i^k - x_j^k) + \mu_i^k\right). \tag{6}$$

Note that the main computation cost of (6) is in calculating the gradient $\nabla f_i(x_i^k)$, which is light-weight. The update of dual variable keeps the same as (5) in ADMM.

Communication censoring. The linearization technique significantly reduces the computation cost of ADMM, but slows down the convergence speed, and hence results in high communication cost. Hence, we introduce the communication-censoring strategy to further reduce the communication cost. Intuitively, when x_i^{k+1} is close to x_i^k, it is not necessary for node i to transmit both of them to neighbors. Motivated by this fact, the communication-censoring strategy prevents transmissions of less informative messages so as to reduce the communication cost. To rigorously explain the communication-censoring strategy, define a state variable $\hat{x}_i^k \in \mathcal{R}^p$ as the latest value that node i has transmitted to neighbors before time k. At time k, after calculating x_i^{k+1}, node i evaluates the difference between $\hat{x}_i^k$ and x_i^{k+1} by their Euclidean distance $\xi_i^{k+1} = \|\hat{x}_i^k - x_i^{k+1}\|$, and then compares the difference with a predefined censoring threshold $\tau^{k+1} \geq 0$. Node i is allowed to transmit x_i^{k+1} to neighbors and update $\hat{x}_i^{k+1} = x_i^{k+1}$, if and only if $\xi_i^{k+1} \geq \tau^{k+1}$. Otherwise, the transmission is censored and $\hat{x}_i^{k+1} = \hat{x}_i^k$. With the state variable $\hat{x}_i^k$,

Algorithm 1 COLA Run by Node i

Input: Initialize local variables to $x_i^0 = 0$, $\mu_i^0 = 0$, $\hat{x}_i^0 = 0$ and $\hat{x}_j^0 = 0$ for all $j \in \mathcal{N}_i$.

while stopping criterion not met **do**
  Compute local primal variable x_i^{k+1} by (7).
  Compute $\xi_i^{k+1} = \|\hat{x}_i^k - x_i^{k+1}\|$.
  If $\xi_i^{k+1} \geq \tau^{k+1}$, transmit x_i^{k+1} to neighbors and let $\hat{x}_i^{k+1}$ $= x_i^{k+1}$; else do not transmit and let $\hat{x}_i^{k+1} = \hat{x}_i^k$.
  If receive x_j^{k+1} from any neighbor j, let $\hat{x}_j^{k+1} = x_j^{k+1}$; else let $\hat{x}_j^{k+1} = \hat{x}_j^k$.
  Update local dual variable μ_i^{k+1} by (8).
  $k = k + 1$.
end While

COLA changes the DLM updates in (6) and (5) to

$$x_i^{k+1} = x_i^k - \frac{1}{2cd_{ii} + \rho}\left(\nabla f_i(x_i^k) + c\sum_{j\in\mathcal{N}_i}(\hat{x}_i^k - \hat{x}_j^k) + \mu_i^k\right), \quad (7)$$

$$\mu_i^{k+1} = \mu_i^k + c\sum_{j\in\mathcal{N}_i}(\hat{x}_i^{k+1} - \hat{x}_j^{k+1}). \quad (8)$$

COLA run by node i is outlined in Algorithm 1.

The communication censoring strategy of COLA is not the standard one used in the existing communication-censored algorithms [31, 32, 33, 34], where all the local primal variables x_i are replaced by the state variables $\hat{x}_i$. We customize the communication censoring strategy for COLA and keep the local primal variable x_i^k in $x_i^k - \frac{1}{2cd_{ii}+\rho}\nabla f_i(x_i^k)$ as it is, because x_i^k has already been available for node i, and is more up-to-date than $\hat{x}_i^k$. Recall that the term $x_i^k - \frac{1}{2cd_{ii}+\rho}\nabla f_i(x_i^k)$ comes from the linearization of $f_i(x_i)$. Intuitively, linearization around $x_i = x_i^k$ leads to faster convergence than linearization around $x_i = \hat{x}_i^k$, which has been validated in our preliminary numerical experiments. On the other hand, we do not change the state variables $\hat{x}_i^k$ and $\hat{x}_i^{k+1}$ by the corresponding local primal variables x_i^k and x_i^{k+1} in the terms $c\sum_{j\in\mathcal{N}_i}(\hat{x}_i^k - \hat{x}_j^k)$ and $c\sum_{j\in\mathcal{N}_i}(\hat{x}_i^{k+1} - \hat{x}_j^{k+1})$. Note that x_i^k and $\hat{x}_i^k$ are not equal when communication censoring happens and the error between them is determined by the censoring threshold τ^k, while the dual update (8) accumulates all the previous differences between the neighboring state variables $\hat{x}_i^k$ and $\hat{x}_j^k$. Thus, replacing the state variables $\hat{x}_i^k$ therein by the corresponding local primal variables x_i^k shall accumulate the errors, and result in instability or even divergence of the recursion.

3. CONVERGENCE AND RATES OF CONVERGENCE

This section proves that COLA converges to an optimal solution of (1) under Assumption 1, which is standard in analysis of network optimization algorithms. Further with Assumption 2, COLA converges to the unique optimal solution of (1) at a linear or sublinear rate, depending on the choice of the censoring threshold. The proofs are given in a longer version of this paper [35].

Assumption 1. *The communication graph $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$ is bidirectionally connected. The local cost functions f_i are proper, closed, convex, and differentiable with Lipschitz continuous gradients, i.e., there exists a constant $M > 0$, such that $\|\nabla f_i(\tilde{x}) - \nabla f_i(\tilde{y})\| \leq M\|\tilde{x} - \tilde{y}\|$ for any i and $\tilde{x}, \tilde{y} \in \mathcal{R}^p$. The dual variable μ of COLA is initialized in the column space of G_o^T, i.e., there exists a vector $\phi^0 \in \mathcal{R}^{mp}$ such that $\mu^0 = G_o^T\phi^0$.*

Assumption 2. *The local cost functions f_i are strongly convex. That is, there exists a constant $m > 0$, such that $\langle\nabla f_i(\tilde{x}) - \nabla f_i(\tilde{y}), \tilde{x} - \tilde{y}\rangle \geq m\|\tilde{x} - \tilde{y}\|^2$ for any i and $\tilde{x}, \tilde{y} \in \mathcal{R}^p$.*

Note that the initial condition in Assumption 1 can be easily satisfied, with the simplest choice being $\mu^0 = 0$.

Theorem 1. *Under Assumption 1, in COLA we choose $c > 0$ and $\rho > 0$ such that $c\lambda_{\min}(L_u) + \rho > \frac{M}{2}$, and set the censoring threshold $\{\tau^k\}$ as a non-increasing non-negative summable sequence such that $\sum_{k=0}^{\infty}\tau^k < \infty$. Then the primal variable x^k converges to an optimal solution x^* of (3).*

Theorem 1 asserts that COLA converges to an optimal solution of (1) under mild conditions and provides guidelines for setting parameters. Fixing ρ, a network with better connectedness (namely, larger $\lambda_{\min}(L_u)$) allows us to choose a smaller penalty constant c. Fixing c and $\lambda_{\min}(L_u)$, the linearization parameter ρ must be large enough to guarantee convergence. Note that ρI_p approximates the Hessians of $f_i(x_i)$. A large ρ over-approximates the curvature and forces x_i^{k+1} to be close to x_i^k, which stabilizes the recursion. On the contrary, a small ρ under-approximates the curvature and allows the local variables to change quickly, at the cost of possible divergence. Regarding the censoring threshold τ^k, we require it to be summable. Intuitively, τ^k determines the maximal error that we introduce to the primal update. When this error is controllable, the convergence of COLA is guaranteed.

Theorem 2. *Under Assumptions 1–2, in COLA we choose $c > 0$ and $\rho > \frac{M^2}{2m}$, and set the censoring threshold $\tau^k = \alpha(\beta)^k$ with $\alpha > 0$ and $\beta \in (0, 1)$. Then there exists a positive constant $\delta > 0$ such that the primal variable x^k converges to the unique optimal solution x^* of (3) at a global linear rate $\mathcal{O}((1+\delta)^{-\frac{k}{2}})$.*

Theorem 3. *Under Assumptions 1 and 2, in COLA we choose $c > 0$ and $\rho > \frac{M^2}{2m}$, and set the censoring threshold $\tau^k = \alpha(k)^{-r}$ with $\alpha > 0$ and $r > 1$. Then there exists a finite k_0 such that the distance between the primal variable x^k and the unique optimal solution x^* of (3) is upper-bounded by a sequence decaying sublinearly to 0 at a rate of $\mathcal{O}((k)^{-\frac{q}{2}})$, where $q \in (0, 2r-1)$, when $k \geq k_0$.*

Theorems 2 and 3 indicate that, to achieve linear (or sublinear) convergence, we have to impose stronger requirements on the parameters. The sequence of censoring threshold should be not only summable, but also linearly (or sublinearly and faster than $\mathcal{O}((k)^{-1})$) decaying. In addition, the parameters c and ρ should be larger. Note that due to $M \geq m$, $\rho > \frac{M^2}{2m} \geq \frac{M}{2}$ and consequently $c\lambda_{\min}(L_u) + \rho > \frac{M}{2}$, which is required in Theorem 1.

4. NUMERICAL EXPERIMENTS

This section provides numerical experiments to demonstrate the satisfactory communication-computation tradeoff of COLA. We shall show that COLA inherits the advantage of cheap computation from its uncensored counterpart DLM [17, 18], but significantly reduces the overall communication cost. Beyond DLM, we compare COLA with the classical ADMM [16] and its censored version COCA [34], both of which do not use the linearization technique and are not computation-efficient. We also compare with the event-triggered sub-gradient descent (ETSD) algorithm, which is a primal domain first-order method. For fair comparisons, the parameters c and ρ are tuned to be the best for the uncensored algorithms DLM and

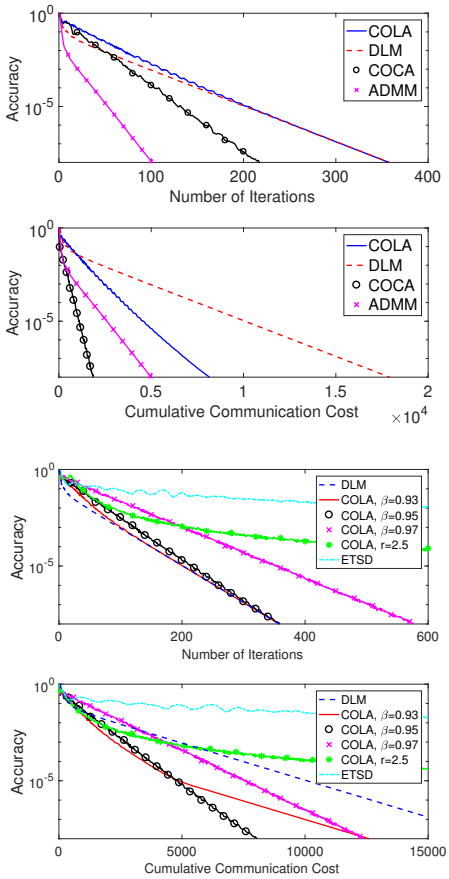

Fig. 1. Random network for decentralized least squares.

ADMM, and kept the same in their censored counterparts, respectively. We use the accuracy of the primal variable as the performance metric, defined by $\|x^k - x^*\|^2/\|x^0 - x^*\|^2$. The computation cost is evaluated by the time spent to reach a target accuracy, and the communication cost is defined as the accumulated number of broadcast messages. The numerical experiments are carried out on a laptop with an Intel I5 processor and 4GB memory. The programming environment is Matlab R2017a in macOS Sierra.

Decentralized least squares. The decentralized least squares problem aims at minimizing (1), where $f_i(\tilde{x}) = (1/2)\|A_{(i)}\tilde{x} - y_{(i)}\|_2^2$, with $A_{(i)} \in \mathcal{R}^{p\times p}$ and $y_{(i)} \in \mathcal{R}^p$ being private for node i. In the experiments, $y_{(i)} = A_{(i)}b_{(i)}$, and entries of $A_{(i)}$ and $b_{(i)}$ follow the i.i.d. uniform distribution within $[0, 1]$. The size of the network is $n = 50$ and the dimension of the local variables is $p = 3$.

First, we compare four algorithms, COLA, DLM, COCA and ADMM, over a random network, as shown in the TOP of Fig. 1. In the random network, 10% of all possible bidirectional edges are randomly chosen to be connected. We use the linear censoring threshold in the form of $\tau^k = \alpha(\beta)^k$, where α and β are hand-tuned in COLA and COCA so as to achieve the best communication efficiency. COLA requires slightly more iterations to reach the target accuracy than DLM. Among the two uncensored algorithms, ADMM also converges faster than COCA. This observation verifies the intuitive idea that linearizing the local cost functions leads to slower convergence. On the other hand, COLA saves the overall communication cost compared with DLM. Given a target accuracy of 10^{-8} and compared with DLM, COLA saves $\sim 1/2$ communication costs.

Next, we compare the choice of the censoring threshold in COLA over the same network. We compare four censoring thresholds, the linear sequences $\tau^k = \alpha(\beta)^k$ with $\alpha = 0.7$ while $\beta = 0.93$, 0.95 and 0.97, as well as the sublinear sequence $\tau^k = \alpha(k)^{-r}$ with $\alpha = 1000$ and $r = 2.5$. The parameters c and ρ remain the same. As shown in the BOTTOM of Fig. 1, using a linear censoring threshold outperforms that using a sublinear censoring threshold, in terms of both communication and computation. The reason is that the sublinear rate of the censoring threshold limits the convergence rate of COLA, as we have theoretically analyzed in Section 3. Regarding the different choices of the linear rate, we observe that a smaller β needs less number of iterations to reach a target accuracy, since it leads to faster decay of the censoring threshold, and thus less communication censoring per iteration. In contrast, with a larger β, we need more number of iterations and less communication cost per iteration. Therefore, a moderate β, such as $\beta = 0.95$ in this case, is preferred.

n	Accuracy	COLA	DLM	COCA	ADMM
50	10^{-4}	1.076s	0.971s	30.119s	9.629s
50	10^{-5}	1.126s	1.055s	37.198s	12.467s
100	10^{-4}	1.466s	1.320s	37.488s	11.175s
100	10^{-5}	1.879s	1.721s	45.302s	15.797s

Table 1. The time spent of the four algorithms in two networks with different numbers of nodes n and target accuracies.

In the BOTTOM of Fig. 1, we also compare COLA with ETSD, a communication-censored primal domain first-order method. ETSD uses Metropolis-Hastings mixing matrix, a sublinear step size $O((k)^{-\frac{2}{3}})$ and a linear censoring threshold $\alpha(\beta)^k$, where α and β are hand-tuned to achieve the best communication efficiency. ETSD requires much more number of iterations and communication cost to reach a target accuracy comparing to COLA. The main reason of the unsatisfactory performance of ETSD is the diminishing step size, which is used to guarantee exact convergence.

Decentralized logistic regression. In the decentralized logistic regression problem, the local cost function of node i is $f_i(\tilde{x}) = \frac{1}{l_i}\sum_{l=1}^{l_i} \ln\left(1+\exp(-y_{(i)l}q_{(i)l}^T\tilde{x})\right)$, where $q_{(i)l} \in \mathcal{R}^p$ is the lth column of a matrix $Q_{(i)} \in \mathcal{R}^{p\times l_i}$, $y_{(i)l} \in \{-1, +1\}$ is the lth element of a binary vector $y_{(i)} \in \mathcal{R}^{l_i}$, and l_i is the number of samples held by node i. In our simulation, the dimension of local variables is $p = 3$. Each l_i is i.i.d. and uniformly chosen from integers within $[1, 10]$. Entries of the first two columns of $Q_{(i)}$ follow the i.i.d. discrete uniform distribution on the set $\{0.1w\}$, $w = 1, 2, \cdots, 10$, while entries of the last column are all set as 1. Entries of $y_{(i)}$ are i.i.d. and follow the uniform distribution on $\{-1, 1\}$. Note that the primal updates of ADMM and COCA have no explicit solutions. Therefore, we solve the subproblems therein by a gradient descent inner loop, which terminates when the ℓ_2 norm of the gradient is less than 10^{-8}.

We conduct the numerical experiments over two random networks with $n = 50$ and $n = 100$ nodes, in both of which 10% of all possible bidirectional edges are randomly chosen to be connected. The censoring threshold in both COCA and COLA is set as $\tau^k = \alpha(\beta)^k$, and the parameters α and β are hand-tuned to obtain the best communication efficiency. As shown in Table 1, we compare the CPU time for COLA, DLM, COCA and ADMM to reach target accuracies 10^{-4} and 10^{-5}. Notice that the two linearized algorithms, COLA and DLM, compute much faster than COCA and ADMM. The time spent by COLA in both networks is a bit more than that of DLM, since COLA requires a few more iterations to reach a target accuracy.

Acknowledgement. Qing Ling is supported by NSF China grant 61573331 and NSF Anhui grant 1608085QF130. Zhi Tian is supported by NSF grant 1741338.

5. REFERENCES

[1] I. Schizas, A. Ribeiro, and G. Giannakis, "Consensus in ad hoc WSNs with noisy links - Part I: Distributed estimation of deterministic signals," IEEE Transactions on Signal Processing, vol. 56, no. 1, pp. 350–364, 2008

[2] G. Giannakis, Q. Ling, G. Mateos, I. Schizas, and H. Zhu, "Decentralized learning for wireless communications and networking," In: Splitting Methods in Communication and Imaging, Science and Engineering, Springer, 2016

[3] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 427–438, 2013

[4] H. Liu, W. Shi, and H. Zhu, "Distributed voltage control in distribution networks: Online and robust implementations," IEEE Transactions on Smart Grid, vol. 9, no. 6, pp. 6106–6117, 2018

[5] X. Zhao and A. Sayed, "Distributed clustering and learning over networks," IEEE Transactions on Signal Processing, vol. 63, no. 13, pp. 3285–3300, 2015

[6] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," In: Proceedings of NIPS, 2017

[7] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multiagent optimization," IEEE Transactions on Automatic Control, vol. 54, no. 1, pp. 48–61, 2009

[8] D. Jakovetic, J. Xavier, and J. Moura, "Fast distributed gradient methods," IEEE Transactions on Automatic Control, vol. 59, no. 5, pp. 1131–1146, 2014

[9] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," SIAM Journal on Optimization, vol. 30, no. 5, pp. 1835–1854, 2016

[10] J. Duchi, A. Agarwal, and M. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," IEEE Transactions on Automatic Control, vol. 57, no. 3, pp. 592–606, 2012

[11] K. Tsianos and M. Rabbat, "Distributed dual averaging for convex optimization under communication delays," In: Proceedings of ACC, 2012

[12] S. Lee, A. Nedic, and M. Raginsky, "Stochastic dual averaging for decentralized online optimization on time-varying communication graphs," IEEE Transactions on Automatic Control, vol. 62, no. 12, pp. 6407–6414, 2017

[13] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton distributed optimization methods," IEEE Transactions on Signal Processing, vol. 65, no. 1, pp. 146–161, 2017

[14] 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, pp. 1–122, 2010

[15] G. Mateos, J. Bazerque, and G. Giannakis, "Distributed sparse linear regression," IEEE Transactions on Signal Processing, vol. 58, no. 10, pp. 5262–5276, 2010

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

[17] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," IEEE Transactions on Signal Processing, vol. 63, pp. 4051–4064, 2015

[18] T. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," IEEE Transactions on Signal Processing, vol. 63, no.2, pp. 482–497, 2015

[19] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," SIAM Journal on Optimization, vol. 25, no. 2, pp. 944–966, 2015

[20] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 2, pp. 120–136, 2016

[21] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "DQM: Decentralized quadratically approximated alternating direction method of multipliers," IEEE Transactions on Signal Processing, vol. 64, no. 19, pp. 5158–5173, 2016

[22] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 4, pp. 507–522, 2016

[23] K. Tsianos, S. Lawlor, M. Rabbat, "Communication/computation tradeoffs in consensus-based distributed optimization," In: Proceedings of NIPS, 2012

[24] A. Berahas, R. Bollapragada, N. Keskar, and E. Wei, "Balancing communication and computation in distributed optimization," arxiv: 1709.02999, 2017

[25] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," arxiv: 1701.03961, 2017

[26] A. Nedic, A. Olshevsky, and M. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," arxiv: 1709.08765, 2017

[27] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," arxiv: 1803.06443, 2018

[28] D. Dimarogonas, E. Frazzoli, and K. Johansson, "Distributed event-triggered control for multi-agent systems," IEEE Transactions on Automatic Control, vol. 57, no. 5, pp. 1291–1297, 2012

[29] E. Garcia, Y. Cao, H. Yu, P. Antsaklis, and D. Casbeer, "Decentralised event-triggered cooperative control with limited communication," International Journal of Control, vol. 86, no. 9, pp. 1479–1488, 2013

[30] C. Nowzari and J. Cortes, "Distributed event-triggered coordination for average consensus on weight-balanced digraphs," Automatica, vol. 68, pp. 237–244, 2016

[31] Q. Lu and H. Li, "Event-triggered discrete-time distributed consensus optimization over time-varying graphs," Complexity, 5385708, 2017

[32] K. Tsianos, S. Lawlor, J. Yu, and M. Rabbat, "Networked optimization with adaptive communication," In: Proceedings of GlobalSIP, 2013

[33] W. Chen and W. Ren, "Event-triggered zero-gradient-sum distributed consensus optimization over directed networks," Automatica, vol. 65, pp. 90–97, 2016

[34] Y. Liu, W. Xu, G. Wu, Z. Tian, and Q. Ling, "COCA: Communication-censored ADMM for decentralized consensus optimization," In: Proceedings of ASILOMAR, 2018

[35] W. Li, Y. Liu, Z. Tian, and Q. Ling, "Communication-censored linearized ADMM for decentralized consensus optimization," Available Online: http://home.ustc.edu.cn/~liweiyu/documents/COLA.pdf