Distributed Dual Coordinate Ascent in General Tree Networks and Communication Network Effect on Synchronous Machine Learning

Myung Cho, Lifeng Lai, and Weiyu Xu

Abstract

Due to the big size of data and limited data storage volume of a single computer or a single server, data are often stored in a distributed manner. Thus, performing large-scale machine learning operations with the distributed datasets through communication networks is often required. In this paper, we investigate the impact of network communication constraints on the convergence speed of the communication-efficient distributed machine learning algorithm. Firstly, we study the convergence rate of the distributed dual coordinate ascent algorithm in a general tree-structured network. Since a tree network model can be understood as the generalization of a star network model, our algorithm can be thought of as the generalization of the distributed dual coordinate ascent in a star network model. Secondly, by considering network communication delays, we optimize the network-constrained distributed dual coordinate ascent algorithm to maximize its convergence speed. In numerical experiments, we consider machine learning scenarios over communication networks, where local workers cannot directly reach to a central node due to constraints in communication, and demonstrate that the usability of our distributed dual coordinate ascent algorithm in tree networks.

Index Terms

distributed machine learning, distributed dataset, machine learning over communication networks

The conference paper of this topic was presented in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2019 [1]. In this journal paper, we provide the full proof of our theorem, and additional numerical experiments and analyses.

M. Cho is with the Department of ECE, Penn State Behrend, Erie, PA 16563, USA (E-mail: mxc6077@psu.edu).

L. Lai is with the Department of ECE, University of California, Davis, CA 95616, USA (E-mail: lflai@ucdavis.edu).

W. Xu is with the Department of ECE, University of Iowa, Iowa City, IA 52242, USA (E-mail: weiyu-xu@uiowa.edu).

I. Introduction

In the past decade, machine learning has been driven by huge amount of data, simply called *big data*. In various fields including education, finance, transportation, healthcare, engineering, and management, etc., big data is fundamentally changing our lives and societies [2], e.g., recommender services [3], disease diagnosis and analysis [4], or even signal recovery [5]. However, due to limited storage volumes in storage server and constraints in communication, we face challenges of processing big data. Especially, big data are very often collected and stored from different locations at different times. Also, it is very expensive, inefficient, and insecure to aggregate distributed data in one central place. Machine learning over wireless communication networks can be a good example having these challenges, where machine learning process is performed through multiple decentralized devices having local data over wireless communication networks without sharing their raw data with others. Therefore, it is quite natural to consider solving large-scale machine learning problems with distributed data over communication networks in order to obtain valuable information from the distributed data.

Solving large-scale machine learning problems dealing with distributed data over communication networks is a challenging problem, due to the limited resources and obstacles including limited communication bandwidth, limited storage volume, limited energy consumption or even privacy and security issues. In order to handle the challenges of distributed data with limited resources, researchers have developed and studied various algorithms in [6–16] and the references therein. More specially, synchronous Stochastic Gradient Decent (SGD) [6, 7], synchronous Stochastic Dual Coordinate Ascent (SDCA) [8–10, 12, 13], asynchronous SGD [11, 14], and asynchronous SDCA [15, 16] for distributed data have been intensively investigated in the literature. Among them, [12] reports that even though the convergence of SGD does not depend on the size of data, SDCA can outperform SGD when we need relatively high solution accuracy. Moreover, asynchronous updating scheme in SGD and SDCA can suffer from the conflicts between intermediate results.

Motivated by these facts, [8–10] consider using synchronous SDCA to solve regularized loss minimization problems in a star network. In the scenario, data are distributed over a few local workers in the star network, and each local worker communicates with a central station. The authors in [8–10] analyze the convergence rate of the distributed SDCA in terms of communication rounds. Espeically, the strong aspects of the proposed distributed optimization

framework in [9, 10] include free-of-tuning parameters or learning rates compared with SGD-based methods, and the readily computable duality gap for fair stopping criterion and efficient accuracy certificates.

However, in practice, the local workers may be organized in various types of network topologies such as a tree, a mesh, or a ring. Especially, in wireless communication networks, due to limited communication power and energy consumption, local workers sometimes cannot directly communicate with a central node. In this situation, the distributed dual coordinate ascent in a star network cannot be used for distributed machine learning. And if intermediate nodes are added for the communication from local workers to a central node, the distributed dual coordinate ascent for a star network will easily suffer from the increased latency and delay in communication. Therefore, considering communication network topologies in distributed machine learning problems is an important problem, and taking advantage of the network topologies may play a significant role in finding efficient solutions for the problems. Then, it is natural to ask how to design and analyze the distributed dual coordinate ascent over a network with general topologies beyond a star network. Additionally, since delay and latency in communication can affect the convergence speed of a distributed machine learning algorithm, it is essential to investigate how network communication delays will affect the design and convergence rate of distributed dual coordinate ascent algorithms previsouly introduced in [8–10] in terms of overall computational time instead of the number of communication rounds. The authors in [17] analyzed the convergence bound in terms of time by considering communication delays in a network for a consensus optimization problem. Additionally, the research [18–22] studied separable consensus problems to each worker by using ADMM techniques. We remark that the regularized loss minimization problem considered in [8–10] is a different problem from the consensus problems considered in [18–22] in the aspect of separability.

The contribution of this paper is three-fold. Firstly, we design the distributed dual coordinate ascent for a regularized loss minimization problem in a general *tree-structured* communication network and analyze the convergence rate of the algorithm over the general tree network. Since a star network is a special case of a general tree network, our distributed dual coordinate ascent algorithm can be thought of as a generalized version of the distributed dual coordinate ascent in a star network. Secondly, we study the influence of the communication constraints in a network on the convergence rate of the distributed dual coordinate ascent. By considering delays in communication, we optimize the network-constrained dual coordinate ascent to maximize its

convergence speed in terms of time, and provide an analytical solution for the optimal number of local iterations depending on the communication delay severity. The analytical solution, which is a function of the delay severity rate between the communication delay and the local processing time, can be used to achieve the fastest convergence speed of the distributed dual coordinate ascent in time. Finally, we demonstrate the usability of our proposing algorithm in machine learning over communication networks, where local workers cannot directly reach to a central node.

The rest of the paper is organized as follows. In Section II, we introduce the regularized loss minimization problem with distributed data. Section III describes a review of existing works on the synchronous distributed dual coordinate ascent in a star network. In Section IV, we propose the generalized distributed dual coordinate ascent in tree-structured networks. Section V describes the convergence analysis of the generalized distributed dual coordinate ascent. In Section VI, we study the communication delay factor in the convergence speed of the distributed dual coordinate ascent. In Section VII, we demonstrate the performance of the generalized distributed dual coordinate ascent and the optimal iteration numbers for the fast convergence speed.

Notations: We denote the set of real numbers as $\mathbb{R}$. We use $[k]$ to denote the index set of the coordinates in the k-th coordinate block. For an index set Q, $\overline{Q}$ and $|Q|$ are used to represent the complement and the cardinality of Q respectively. We use bold letters to represent vectors and matrices. If we use an index set as a subscript of a vector (resp. matrix), we refer to the partial vector (resp. partial matrix) over the index set (resp. with columns over the index set). The superscript (t) is used to denote the t-th iteration. For example, $\boldsymbol{\alpha}_{[k]}^{(t)}$ represents a partial vector $\boldsymbol{\alpha}$ over the k-th block coordinate set at the t-th iteration. We reserve the superscript $\star$ to denote the optimal solution to an optimization problem.

II. Problem Formulation

We consider the following regularized loss minimization problem [8–10, 13, 15, 16]:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\ P(\boldsymbol{w}) \triangleq \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i), \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$, $i = 1, 2, ..., m$, are data points, $\ell_i(\cdot)$, $i = 1, 2, ..., m$, are loss functions, and λ is a tuning parameter for a regularization term. Note that due to the regularization term for $\boldsymbol{w}$, which is a global variable, this minimization problem is not separable for each distributed node unlike the consensus problems introduced in [18–22], where the regularization term is defined like

$\sum_{k=1}^{K} r(\boldsymbol{w}_k)$. Here $r(\cdot)$ is a regularization function and K is the number of distributed nodes. By considering different loss functions, (1) can be interpreted as various machine learning problems including regression and classification. For instance, for linear classification, by choosing the loss function $\ell_i(\cdot)$ to the hinge loss, i.e., $\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) = \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i))$, (1) with labeled dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, where $y_i \in \mathbb{R}$ is label information, can be understood as the linear Support Vector Machine (SVM) classification problem. For regression, we can set $\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) = (\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2$ with some measurement data y_i, $i = 1, 2, ..., m$. Throughout the paper, we assume that the data points $\boldsymbol{x}_i$, $i = 1, 2, ..., m$, are normalized in ℓ_2 norm, i.e., $\|\boldsymbol{x}_i\| \leq 1$, $i = 1, 2, ..., m$, and the dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ is divided and distributed over a network.

From the primal problem (1), we have the following dual problem by considering the conjugate function, i.e., $\ell_i(a) = \sup_b ab - \ell_i^*(b)$, where $a, b \in \mathbb{R}$ and $\ell_i(\cdot)$ is convex:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}}\ D(\boldsymbol{\alpha}) \triangleq -\frac{\lambda}{2}\|\boldsymbol{A}\boldsymbol{\alpha}\|^2 - \frac{1}{m}\sum_{i=1}^{m}\ell_i^*(-\alpha_i), \tag{2}$$

where α_i is the i-th element of the dual vector $\boldsymbol{\alpha} \in \mathbb{R}^m$, and the data matrix $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ whose i-th column is $\frac{1}{\lambda m}\boldsymbol{x}_i$, i.e., $\boldsymbol{A}_i = \frac{1}{\lambda m}\boldsymbol{x}_i$, is introduced for notation convenience. By defining $\boldsymbol{w}(\boldsymbol{\alpha}) \triangleq \boldsymbol{A}\boldsymbol{\alpha}$ shown in [9, 13], we have the duality gap as $P(\boldsymbol{w}(\boldsymbol{\alpha}))-D(\boldsymbol{\alpha})$ for a useful and readily computable stopping criteria. It is noteworthy that from the duality principle [23], we have $P(\boldsymbol{w}) \geq D(\boldsymbol{\alpha})$ for all $\boldsymbol{w}$ and $\boldsymbol{\alpha}$, and thus, $P(\boldsymbol{w}(\boldsymbol{\alpha})) \geq D(\boldsymbol{\alpha})$ for all $\boldsymbol{\alpha}$. If $\boldsymbol{\alpha} = \boldsymbol{\alpha}^\star$, which is the optimal solution to the dual problem (2), and the loss function $\ell(\cdot)$ is convex, we have $P(\boldsymbol{w}(\boldsymbol{\alpha}^\star)) = D(\boldsymbol{\alpha}^\star)$ from strong duality condition. Thus, $\boldsymbol{w}(\boldsymbol{\alpha}^\star)$ becomes $\boldsymbol{w}^\star$, which is the optimal solution to the primal problem (1).

In the following sections, we consider a distributed dual coordinate ascent for the regularized loss minimization problem over distributed data. We firstly review the previous research on the distributed dual coordinate ascent in a star network.

III. Review of the distributed dual coordinate ascent in a star network

The distributed dual coordinate ascent for the regularized loss minimization problem over distributed data in a network has been studied in [8–10, 15], where a star network topology for the network is considered as shown in Figure 1. In particular, the authors in [9] introduced a distributed dual coordinate ascent framework, called the Communication-Efficient Distributed Dual Coordinate Ascent (CoCoA), and later proposed CoCoA+ [10], which is an enhanced version of CoCoA by adjusting the parameter value in the accumulation of intermediate results for faster convergence speed than CoCoA. Since we are interested in the distributed dual coordinate

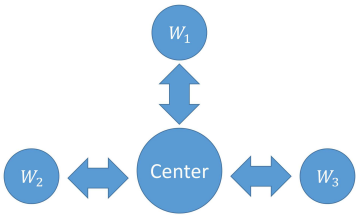


Fig. 1. Illustration of a star network having one central station and three local workers W_1, W_2 and W_3.

ascent for various structural network topologies and their influences to the performance of the distributed algorithm, we provide a high level review of CoCoA proposed in [9].

Suppose a star network has K local workers and each local worker has disjoint parts of dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$. Specifically, the k-th local worker has training data $\{(\boldsymbol{x}_i, y_i)\}$, $i \in [k]$, where $[k]$ represents the index set for the training data of the k-th local worker. Hence, we have $|\cup_{k=1}^K [k]| = m$. With this problem setting, the authors in [9] introduced the distributed dual coordinate ascent for a star network. Due to the nature of the distributed algorithm, the algorithm updates the global variable in the outer iteration, and locally each worker has inner iterations. Particularly, at the t-th outer iteration of the algorithm, each worker solves a local dual problem for given dataset via LocalDualMethod(·), which represents any dual method to solve (2), e.g. Stochastic Dual Coordinate Ascent (SDCA), simply denoted by LocalSDCA(·), through inner iterations. And then, each local worker sends the intermediate solution to the center node. The center node collects and accumulates all the results from the local workers, and then updates and shares the global solution $\boldsymbol{w}^{(t)}$ at the t-th outer iteration back to the workers. Algorithm 1 describes the detail steps of the distributed coordinate ascent in a star network. The following theorem characterizes the convergence rate of the algorithm in [9].

Theorem 1 ([9, Theorem 2])**.** *Suppose that Algorithm 1 is run for T outer iterations of K local computers with the procedure LocalSDCA(·) having local geometric improvement Θ. Further, assume that the loss functions $\ell_i(\cdot)$ are $1/\gamma$-smooth. Then, the following geometric convergence rate holds for the global (dual) objective:*

$$\mathbb{E}[D(\boldsymbol{\alpha}^\star) - D(\boldsymbol{\alpha}^{(T)})] \leq \Big(1 - (1-\Theta)\frac{1}{K}\frac{\lambda m\gamma}{\rho + \lambda m\gamma}\Big)^T \big(D(\boldsymbol{\alpha}^\star) - D(\boldsymbol{\alpha}^{(0)})\big), \tag{3}$$

where m is the size of the whole dataset and ρ is any real number satisfying

$$\rho \geq \rho_{min} \triangleq \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\textit{maximize}} \ \lambda^2 m^2 \frac{\sum_{k=1}^K \|\boldsymbol{A}_{[k]}\boldsymbol{\alpha}_{[k]}\|^2 - \|\boldsymbol{A}\boldsymbol{\alpha}\|^2}{\|\boldsymbol{\alpha}\|^2} \geq 0.$$

Algorithm 1: Communication-efficient Distributed Dual Coordinate Ascent (CoCoA) [9]

Input: $T \geq 1$

Output: $\boldsymbol{w}$, $\boldsymbol{\alpha}$

Data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ distributed over K local workers

Initialization: $\boldsymbol{\alpha}_{[k]}^{(0)} \leftarrow \mathbf{0}$ for all local workers, and $\boldsymbol{w}^{(0)} \leftarrow \mathbf{0}$

for $t = 1$ **to** T **do**

  for *all local workers* $k = 1, 2, ..., K$ *in parallel* **do**

    $(\triangle\boldsymbol{\alpha}_{[k]}, \triangle\boldsymbol{w}_k) \leftarrow \text{LocalDualMethod}(\boldsymbol{\alpha}_{[k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$

    $\boldsymbol{\alpha}_{[k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[k]}^{(t-1)} + \frac{1}{K} \triangle \boldsymbol{\alpha}_{[k]}$

  end

  send $\triangle\boldsymbol{w}_k$, $k = 1, ..., K$, to the central station

  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{1}{K} \sum_{k=1}^{K} \triangle\boldsymbol{w}_k$

  distribute $\boldsymbol{w}^{(t)}$ to local workers

end

With LocalSDCA(·), which uses the SDCA to solve the dual problem for given dataset at each worker, the local geometric improvement Θ can be set to

$$\Theta = (1 - s/\tilde{m})^H, \tag{4}$$

where $\tilde{m} \triangleq \max_{k=1,...,K} m_k$ is the size of the largest block of coordinates among K local workers, H is the number of local (or inner) iterations in LocalSDCA(·), and $s \in [0, 1]$ is a step size of the gradient ascent which determines how far the next solution will be taken from the current solution at each iteration. Additionally, by choosing different parameter values instead of $\frac{1}{K}$ in the summation of $\triangle\boldsymbol{w}_k$'s in Algorithm 1, the authors in [10] proposed CoCoA+, which has the same framework as CoCoA introduced in Algorithm 1, for faster convergence speed than CoCoA.

CoCoA has been shown to work well for distributed machine learning problems with distributed data in a star network, which is a simple network model. However, the topology of a network may not necessarily be a star network. In the next section, we study the distributed dual coordinate ascent in a general network, which is a tree-structured network model.

IV. Generalized distributed dual coordinate ascent in tree networks

One may think of a connected communication network, e.g., a spanning tree network, as a virtual star network by considering the long relays of links from a central node to each leaf node

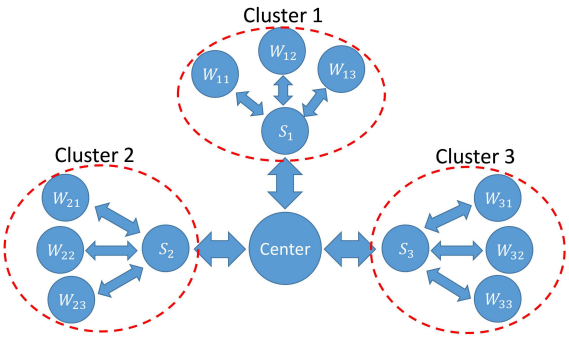


Fig. 2. Illustration of a tree-structured network, which has two layers. In the network, a central station (root node) has three direct child nodes S_1, S_2 and S_3. Each node S_i has three direct child nodes W_{ij}, $j = 1, 2, 3$.

as a direct virtual-link to the central node from a leaf node. Since communication delays are normally exist in a network and the communication is a big burden of distributed algorithms, the distributed algorithms in the virtual star network can easily suffer from the long delays in communication by significantly slowing down the convergence of the distributed algorithms. Therefore, in a connected communication network, it is efficient to perform distributed optimization among local workers close to each other, and then, communicate the intermediate results to a central or sub-central nodes. Based on this idea, we investigate how to design the distributed dual coordinate ascent over a general tree-structured network instead of a simple star network, and provide its convergence analysis. Since every connected communication network has a spanning tree, we choose to investigate the distributed algorithm over a tree-structured network, which is also a generalization of a star network.

In Figure 2, we show a two-layer tree network as an example of a general tree-structured network, where the number of layers represents the depth of the tree network. The root node of the tree network represents the central station of the network. Any other tree nodes correspond to local workers. Each tree node may have several direct child nodes. For example, the root node has three direct child nodes S_1, S_2, and S_3 in Figure 2. Without loss of generality, we assume that only the local workers corresponding to the leaf nodes have the distributed data, which is the disjoint segmented blocks of the data matrix $\boldsymbol{A}$ in column-wise. Note that $\boldsymbol{A}_i = \frac{1}{\lambda m}\boldsymbol{x}_i$, where $\boldsymbol{A}_i$ is the i-th column of $\boldsymbol{A}$ and $\boldsymbol{x}_i \in \mathbb{R}^d$ is the i-th data point. If a non-leaf node Q stores data, we can always create a virtual leaf node L attached to Q, and "stores" the data in L. Thus, without loss of generality, we can assume that the dataset $\{\boldsymbol{x}_i, y\}_{i=1}^m$ are distributed only to leaf nodes.

For a tree node Q, we also denote the set of indices of data points stored in the subtree with Q,

Algorithm 2: TreeDualMethod: Distributed Dual Coordinate Ascent for the Root Node Q

Input: $R \geq 1$

Initialization: $\boldsymbol{\alpha}_{[Q;k]}^{(0)} \leftarrow 0$ for all direct child nodes k of node Q, $\boldsymbol{w}^{(0)} \leftarrow 0$

for $t = 1$ **to** R **do**

  for *all direct child nodes* $k = 1, 2, ..., K$ *in parallel* **do**

    $(\triangle \boldsymbol{\alpha}_{[Q;k]}, \triangle \boldsymbol{w}_k) \leftarrow \text{TreeDualMethod}(\boldsymbol{\alpha}_{[Q;k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$

    $\boldsymbol{\alpha}_{[Q;k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[Q;k]}^{(t-1)} + \frac{1}{K} \triangle \boldsymbol{\alpha}_{[Q;k]}$

  end

  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{1}{K} \sum_{k=1}^{K} \triangle \boldsymbol{w}_k$

end

Output: $\boldsymbol{\alpha}^{(R)}$, and $\boldsymbol{w}^{(R)}$

where Q is considered as the root node of the subtree, by Q. Hence, the subtree includes Q and its indirect and direct child nodes. We denote the set of indices of data points stored in the subtree whose root node is the k-th direct child node of Q as $[Q;k]$. If Q is a leaf node, we denote the number of data points stored in Q as m_Q. In a tree network, we additionally assume that a node can only communicate with its direct child nodes or its direct parent node. We then introduce the generalized distributed dual coordinate ascent, which we call TreeDualMethod, to solve the dual problem (2) with distributed data stored in a general tree-structured network. Algorithm 2, Algorithm 3 and Procedure P describe the computational steps of TreeDualMethod for the root node, a general tree node (not root or leaf), and a leaf node respectively. It is noteworthy that like the distributed algorithm in a star network case, in the distributed networks, the output $\triangle \boldsymbol{w}_Q$ in Procedure P and Algorithm 3 or the output $\boldsymbol{w}$ in Algorithm 2 are transmitted between nodes, while the outputs $\boldsymbol{\alpha}$ and $\triangle \boldsymbol{\alpha}_Q$ are not transmitted through communication networks. Each node provides $\boldsymbol{\alpha}$ or $\Delta \boldsymbol{\alpha}_Q$ as an output of each node, but those outputs are used in each node at the next iteration without transmission to other nodes. Therefore, even though we have a large dataset, the communication cost is not affected by the size of the dataset. Also, when the dimension of $\boldsymbol{\alpha} \in \mathbb{R}^m$ is large, i.e., large amount of data, transmitting $\triangle \boldsymbol{w}_Q$ or $\boldsymbol{w}$ whose dimension is much smaller than m, is good for communication efficiency in distributed algorithms.

V. Convergence analysis of TreeDualMethod for a tree network

We analyze the convergence rate of the distributed dual coordinate ascent in a general tree-structured network model in this section. In a nutshell, we show a recursive relation between the convergence rate of the algorithm at a tree node Q and that at the node Q's direct child

Algorithm 3: TreeDualMethod: Distributed Dual Coordinate Ascent for a General Tree Node Q (not root or leaf)

Input: $T \geq 1$, $\boldsymbol{\alpha}_Q$, $\boldsymbol{w}$

Initialization: $\boldsymbol{\alpha}_{[Q;k]}^{(0)} \leftarrow \boldsymbol{\alpha}_{[Q;k]}$ for all direct child nodes k of node Q , $\boldsymbol{w}^{(0)} \leftarrow \boldsymbol{w}$

for $t = 1$ **to** T **do**

  for *all direct child nodes* $k = 1, 2, ..., K$ *of* Q *in parallel* **do**

    $(\triangle \boldsymbol{\alpha}_{[Q;k]}, \triangle \boldsymbol{w}_k) \leftarrow$ TreeDualMethod$(\boldsymbol{\alpha}_{[Q;k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$

    $\boldsymbol{\alpha}_{[Q;k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[Q;k]}^{(t-1)} + \frac{1}{K} \triangle \boldsymbol{\alpha}_{[Q;k]}$

  end

  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{1}{K} \sum_{k=1}^{K} \triangle \boldsymbol{w}_k$

end

Output: $\triangle \boldsymbol{\alpha}_Q \triangleq \boldsymbol{\alpha}_Q^{(T)} - \boldsymbol{\alpha}_Q^{(0)}$, and $\triangle \boldsymbol{w}_Q \triangleq \boldsymbol{A}_Q \triangle \boldsymbol{\alpha}_Q$

Procedure P. TreeDualMethod: Distributed Dual Coordinate Ascent for a Leaf Node Q

Input: $H \geq 1$, $\boldsymbol{\alpha}_Q \in \mathbb{R}^{m_Q}$, and $\boldsymbol{w} \in \mathbb{R}^d$ consistent with other coordinate blocks of $\boldsymbol{\alpha}$ s.t. $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{\alpha}$

Data: $\{(\boldsymbol{x}_i, y_i)\}_{i \in Q}$, where $|Q| = m_Q$

Initialization: $\triangle \boldsymbol{\alpha}_Q \leftarrow 0 \in \mathbb{R}^{m_Q}$, and $\boldsymbol{w}^{(0)} \leftarrow \boldsymbol{w}$

for $h = 1$ **to** H **do**

  choose $i \in Q$ uniformly at random

  find $\triangle \alpha$ maximizing $-\frac{\lambda m}{2} \|\boldsymbol{w}^{(h-1)} + \frac{1}{\lambda m} \triangle \alpha \boldsymbol{x}_i\|^2 - \ell_i^*(-(\alpha_i^{(h-1)} + \triangle \alpha))$

  $\alpha_i^{(h)} \leftarrow \alpha_i^{(h-1)} + \triangle \alpha$

  $(\triangle \boldsymbol{\alpha}_Q)_i \leftarrow (\triangle \boldsymbol{\alpha}_Q)_i + \triangle \alpha$

  $\boldsymbol{w}^{(h)} \leftarrow \boldsymbol{w}^{(h-1)} + \frac{1}{\lambda m} \triangle \alpha \boldsymbol{x}_i$

end

Output: $\triangle \boldsymbol{\alpha}_Q$ and $\triangle \boldsymbol{w}_Q \triangleq \boldsymbol{A}_Q \triangle \boldsymbol{\alpha}_Q$

nodes. Hence, the overall convergence rate of the distributed dual coordinate ascent in a general tree-structured network can be understood in a recursive manner, where the number of recursions is dependent on the number of layers of the tree network.

Suppose Q has K direct child nodes. We use $\boldsymbol{\alpha}_{[Q;k]}$ to denote the partial dual variable vector corresponding to its k-th direct child node, where $1 \leq k \leq K$. Then, let us define the local suboptimality gap for the k-th direct child node of Q as

$$\epsilon_{Q,k}(\boldsymbol{\alpha}) \triangleq \underset{\hat{\boldsymbol{\alpha}}_{[Q;k]}}{\text{maximize}} \; D(\boldsymbol{\alpha}_{[Q;1]}, ..., \hat{\boldsymbol{\alpha}}_{[Q;k]}, ..., \boldsymbol{\alpha}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[Q;1]}, ..., \boldsymbol{\alpha}_{[Q;k]}, ..., \boldsymbol{\alpha}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}}). \tag{5}$$

Remark that the local suboptimality gap for the k-th child node is defined with fixing $\boldsymbol{\alpha}_{\overline{Q}}$ and

$\boldsymbol{\alpha}_{[Q;i]}$'s, where $i \neq k$, and only updating $\boldsymbol{\alpha}_{[Q;k]}$. Thus, the local suboptimality gap for the k-th direct child node of Q represents the maximum objective value gap that the k-th direct child node of Q can achieve from the current $\boldsymbol{\alpha}^{(t)}$ value with fixing other α_i, $i \notin [Q;k]$, variables. Then, we introduce the following assumption about the local geometric improvement of TreeDualMethod at the k-th direct child node of Q.

Assumption 1 (Geometric improvement of TreeDualMethod at a direct child node)**.** *For a tree node Q, we assume that there exists $\Theta \in [0,1)$ such that for any given $\boldsymbol{\alpha}$, TreeDualMethod at the k-th direct child node of Q returns an update $\triangle\boldsymbol{\alpha}_{[Q;k]}$ satisfying*

$$\mathbb{E}[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[Q;1]}, ..., \boldsymbol{\alpha}_{[Q;k-1]}, \boldsymbol{\alpha}_{[Q;k]} + \triangle\boldsymbol{\alpha}_{[Q;k]}, ..., \boldsymbol{\alpha}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}})] \leq \Theta \cdot \epsilon_{Q,k}(\boldsymbol{\alpha}). \tag{6}$$

Note that Assumption 1 here is introduced for a tree node in a general tree network, while Assumption 1 of [9] is introduced for an abstract function in the distributed algorithm framework.

For a leaf node, we use LocalSDCA for TreeDualMethod described in Procedure P as in [9]. We remark that this geometric improvement condition holds true with LocalSDCA if the k-th direct child node of Q is a leaf child node with Θ introduced in (4). We provide the following proposition about the convergence bound for a leaf node B even with the input $\boldsymbol{w}$ also determined by $\boldsymbol{\alpha}_{\overline{Q}}$ and $\boldsymbol{\alpha}_{Q \setminus B}$ in Procedure P.

Proposition 1 ([9, Proposition 1])**.** *Let us consider a tree node Q whose direct child node B is a leaf node. Assume that loss functions $\ell_i(\cdot)$ are $1/\gamma$-smooth. Then for the leaf node B, Assumption 1 holds with*

$$\Theta = \left(1 - \frac{\lambda m \gamma}{1 + \lambda m \gamma} \frac{1}{m_B}\right)^H. \tag{7}$$

where m_B is the size of data stored at node B, H is the number of iterations in Procedure P.

Additionally, Theorem 2, which is our main result, shows that if the geometric improvement condition holds true for direct child nodes of Q, then the geometric improvement condition also holds true for Q; thus it leads to a recursive calculation of the convergence rate for the whole tree network.

Theorem 2. *Let us consider a tree node Q which has K direct child nodes satisfying the local geometric improvement requirement introduced in Assumption 1, with parameters Θ_1, Θ_2, ..., and Θ_K. We assume that Algorithm 3 (or Algorithm 2) has an input $\boldsymbol{w}$ and is run for T iterations. We further assume that loss functions $\ell_i(\cdot)$'s are $1/\gamma$-smooth.*

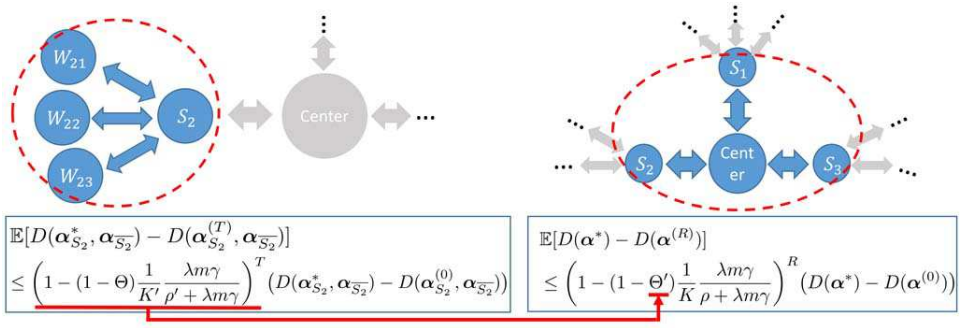


Fig. 3. Illustration of the structure of the tree network factor in convergence analysis.

Then, for any input $\boldsymbol{w}$ to Algorithm 3 (or Algorithm 2), the following geometric convergence rate holds for Q:

$$\mathbb{E}[D(\boldsymbol{\alpha}^*_Q, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(T)}_Q, \boldsymbol{\alpha}_{\overline{Q}})] \leq \left(1-(1-\Theta)\frac{1}{K}\frac{\lambda m\gamma}{\rho+\lambda m\gamma}\right)^T \left(D(\boldsymbol{\alpha}^*_Q, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(0)}_Q, \boldsymbol{\alpha}_{\overline{Q}})\right), \tag{8}$$

where $\Theta = \max_k \Theta_k$, and ρ is any real number satisfying

$$\rho \geq \rho_{min} \triangleq \underset{\boldsymbol{\alpha}_Q \in \mathbb{R}^{|Q|}}{\textit{maximize}} \ \lambda^2 m^2 \frac{\sum_{k=1}^K \|\boldsymbol{A}_{[Q;k]}\boldsymbol{\alpha}_{[Q;k]}\|^2 - \|\boldsymbol{A}_Q\boldsymbol{\alpha}_Q\|^2}{\|\boldsymbol{\alpha}_Q\|^2} \geq 0.$$

Proposition 1 is for the local geometric improvement of TreeDualMethod at a leaf node. Theorem 2 is for the local geometric improvement of TreeDualMethod at any non-leaf tree node. Note that $\left(1-(1-\Theta)\frac{1}{K}\frac{\lambda m\gamma}{\rho+\lambda m\gamma}\right)^T$ in (8) becomes the "Θ" for a tree node Q, and then, (8) is interpreted as the local geometric improvement of TreeDualMethod at the direct child node by the direct parent node of Q. Therefore, by combining Theorem 2 with Proposition 1, we can recursively obtain the convergence rate of the generalized distributed dual coordinate ascent algorithm for the whole tree network. Figure 3 illustrates the structure of the tree network factor in convergence rate, which is shown through Θ and Θ'. Specifically, depending on the number of layers in the tree network, the depth of recursion in computing Θ in (8) is determined.

Remark that Theorem 2 is different from Theorem 2 of [9] in three aspects. Firstly, Theorem 2 is applicable to any tree node in a general tree network, beyond a star network discussed in [9]. Secondly, even when the input $\boldsymbol{w}$ of Algorithm 3 is determined by not only $\boldsymbol{\alpha}_Q$ but also $\boldsymbol{\alpha}_{\overline{Q}}$, Theorem 2 holds. Note that $\boldsymbol{w} = \boldsymbol{A}(\boldsymbol{\alpha}_Q, \boldsymbol{\alpha}_{\overline{Q}}) = \boldsymbol{A}_Q\boldsymbol{\alpha}_Q + \boldsymbol{A}_{\overline{Q}}\boldsymbol{\alpha}_{\overline{Q}}$. Unlike our Theorem, in Theorem 2 of [9], due to the star network topology, a local worker has $\boldsymbol{w}$ as an input from the root node which is updated with intermediate results obtained from all the local workers. Hence, $\boldsymbol{\alpha}_{\overline{Q}}$ is not considered in Theorem 2 of [9] and its proof. Our proof of Theorem 2 addresses this challenge that the input $\boldsymbol{w}$ is also affected by $\boldsymbol{\alpha}_{\overline{Q}}$. Therefore, we have to deal with both updating coordinates $\boldsymbol{\alpha}_Q \in \mathbb{R}^{|Q|}$ and un-updating coordinates $\boldsymbol{\alpha}_{\overline{Q}} \in \mathbb{R}^{|\overline{Q}|}$, where $|Q| + |\overline{Q}| = m$, while in

the proof of Theorem 2 of [9], all the coordinates are updating coordinates, i.e., $\boldsymbol{\alpha} \in \mathbb{R}^m$. For the readability, we place the proof of Theorem 2 in Appendix A. Finally, unlike [9], we do not consider different local-dual problem introduced in Eqn. (8) of [9] for local workers, but deal with the original dual problem introduced in (2) with fixed $\overline{\boldsymbol{w}} \triangleq \boldsymbol{A}_{\overline{Q}}\boldsymbol{\alpha}_{\overline{Q}}$ for a general tree node Q. Therefore, our theorem works for any tree node in a general tree network rather than just for one central node, which allows the recursive convergence analysis of the distributed dual coordinate ascent in a general tree network as shown in Figure 3.

We have discussed how the network topology can affect the convergence rate of the distributed dual coordinate ascent, which is expressed in terms of the number of iterations. However, for distributed algorithms, communications in a network can be a bottleneck of the convergence of the distributed algorithms. Therefore, it is quite natural to consider communication delay, which is normally expressed in time, in order to predict or estimate the convergence speed of the distributed algorithms. In the next section, we study how communication delay, which is one of major network constraints, impacts the convergence of distributed dual coordinate ascent algorithms. By taking communication delays into account, we optimize the number of local iterations H in Procedure P and T in Algorithm 3 for maximum convergence speed.

VI. Impacts of communication delay on the convergence rate of distributed dual coordinate ascent

Earlier works [8–10] bounded the convergence of distributed dual coordinate ascent algorithms with respect to the number of inner and outer iterations. However, in distributed algorithms, there may be significant communication delays in a distributed network. Thus, the convergence speed of distributed algorithms depends on not only how many iterations of these algorithms have been run, but also the communication delays in performing these iterations. Intuitively, if the communication delay is close to zero, local workers may be better to perform a small number of local iterations, and communicate with the central station at a higher frequency; on the other hand, if the communication delay is large, namely, there is a large communication cost, then local workers may want to perform more local iterations before communicating with the central station in order to speed up convergence. Therefore, our goal here is to investigate the convergence speed of distributed dual coordinate ascent with respect to total execution time including computational time and communication delays, and to optimize the number of local iterations by considering communication delays to achieve the maximum convergence speed of the distributed

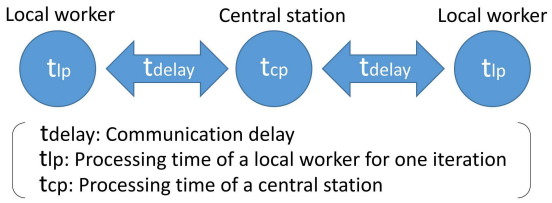


Fig. 4. Definition of delay and computational time.

dual coordinate ascent. The research [17, 24–26] studied the impact of the communication delays on the convergence rate of algorithms in various distributed optimization problems including distributed consensus problems. However, for the regularized loss minimization problem dealing with in this paper, to the best of our knowledge, our paper is the first one to analytically study the communication delay impact on the convergence rate.

For simplicity, let us first consider a star network as shown in Figure 1 and the corresponding Algorithm 1. Since the communication delay is normally given in time, we need to consider both time and the number of iterations in the convergence analysis in order to obtain the optimal number of iterations in practical applications having communication delay and computational time. We assume that the round-trip communication delay between a local worker and the central station is t_{delay}. We use t_{lp} to denote the computational time for one local iteration at a worker, and use t_{cp} to denote the computational time for parameter update at the central station. Figure 4 illustrates the communication delay, and the processing time of each local and central station.

Suppose that each local worker performs H local iterations before communicating with the central station, and there are T outer iterations in total. Then, the total experienced time is

$$t_{total} = (t_{lp}H + t_{delay} + t_{cp}) \cdot T. \tag{9}$$

Hence, the number of outer iterations T is given by

$$T = t_{total}/(t_{lp}H + t_{delay} + t_{cp}). \tag{10}$$

From (8), for T outer iterations, the expected gap between the optimal objective value and the current objective value with Algorithm 1 is expressed as

$$\left(1 - \left(1 - [1-\delta]^H\right)^{C/K}\right)^T, \tag{11}$$

where $\delta = \frac{s}{\tilde{m}}$, $C = \lambda m\gamma/(\rho + \lambda m\gamma)$, and K is the number of local workers. In order to minimize the gap in objective value (11) for a given total time t_{total}, we introduce the following optimization problem over the number of local iterations H by plugging (10) into (11):

$$\underset{H\geq 0}{\text{minimize}}\; F(H) \triangleq \left(1-\left(1-\left[1-\delta\right]^{H}\right)^{C/K}\right)^{\frac{t_{total}}{t_{lp}H+t_{delay}+t_{cp}}}. \tag{12}$$

In order to figure out the optimal number of local iterations, let us find the critical point of the objective function $F(H)$. By applying logarithm to $F(H)$, we have

$$\ln F(H) = \underbrace{\frac{t_{total}/t_{lp}}{H+(t_{delay}+t_{cp})/t_{lp}}}_{(A)} \underbrace{\ln\left(\frac{K-C}{K}+\frac{C}{K}\left[1-\delta\right]^{H}\right)}_{(B)}. \tag{13}$$

(13) can be interpreted as the multiplication of two parts: the fraction part (A) and the logarithm part (B). Note that the fraction part (A) is a decreasing function over H. And for the logarithm part (B), as H increases, (B) goes to $\ln((K-C)/K)$, which is less than zero, due to the condition $0 \leq 1-\delta < 1$. At $H = 0$, $\ln F(H)$ is 0 due to $(B) = 0$. As H goes to infinity, $\ln F(H)$ will go to 0 due to $(A) = 0$. Therefore, we can expect at least a critical point at some H. In order to figure out the critical point of (13), which is the same critical point of $F(H)$, we calculate the first order condition as follows:

$$\frac{d\ln F(H)}{dH} = \frac{\left(\frac{K-C}{K}\right)\left(\frac{t_{total}}{t_{lp}}\right)(1-\delta)^{H}\ln(1-\delta)}{\left(\frac{K-C}{K}+\frac{C}{K}\left[1-\delta\right]^{H}\right)\left(H+\frac{t_{delay}+t_{cp}}{t_{lp}}\right)} - \frac{\left(\frac{t_{total}}{t_{lp}}\right)\ln\left(\frac{K-C}{K}+\frac{C}{K}\left[1-\delta\right]^{H}\right)}{\left(H+\frac{t_{delay}+t_{cp}}{t_{lp}}\right)^{2}} = 0. \tag{14}$$

By simplifying (14) and denoting $\frac{t_{delay}+t_{cp}}{t_{lp}}$ to r, we have the first order condition over H as

$$\underbrace{{}^{(K-C)}/_{K}\left(H+r\right)\left[1-\delta\right]^{H}\ln(1-\delta)}_{(C)} - \underbrace{\left({}^{(K-C)}/_{K}+{}^{C}/_{K}\left[1-\delta\right]^{H}\right)\ln\left({}^{(K-C)}/_{K}+{}^{C}/_{K}\left[1-\delta\right]^{H}\right)}_{(D)} = 0. \tag{15}$$

When H is large enough, (D) is approximated to $\left(\frac{K-C}{K}\right)\ln\left(\frac{K-C}{K}\right)$. And then, we have

$${}^{(K-C)}/_{K}\left(H+r\right)\left[1-\delta\right]^{H}\ln(1-\delta) = {}^{(K-C)}/_{K}\ln\left({}^{(K-C)}/_{K}\right). \tag{16}$$

Note that (16) has Lambert W-function [27], which is defined as when $xe^{x} = a$, the solution x is $W(a)$, where $W(\cdot)$ is the Lambert W-function. By using the definition of the Lamber W-function, we have the following optimal local iteration H from (16):

$$H = \frac{1}{\ln(1-\delta)}W\left(\left[1-\delta\right]^{r}\ln\left({}^{(K-C)}/_{K}\right)\right) - r. \tag{17}$$

From the recursive manner of the convergence analysis in a tree network as introduced in Section V, the optimal number of iterations T in Algorithm 3 for a node Q can also be obtained by using aforementioned equation (12) with slightly different interpretation. In the tree network, the number of local iterations H in (12) is understood as the number of local iteration T in Algorithm 3 for the node Q. The computational time for the local iteration at a worker, denoted by t_{lp}, is interpreted as the computational time for one-time receiving the updating intermediate results from Q's child nodes. And t_{delay} and t_{cp} represent the communication delay time and

the processing time at Q's direct parent node respectively. Thus, with the same equation as (12) with different interpretation, the optimal number of local iterations for a general tree node Q can be obtained as (17).

In the numerical experiments section, we will further investigate the impact of the communication delay severity r and other parameters including C, K, and δ in (17) on the optimal number of local iterations H.

VII. Numerical experiments

In wireless communication networks, it can often occur that the local workers are located out of communication range from the central node due to communication constraints such as limited communication power, long distance, limited bandwidth, and limited latency, etc. By reflecting the communication constraints, in the numerical experiments, we consider machine learning scenarios over communication networks, where local workers cannot directly communicate with a central node. Thus, in the distributed dual coordinate ascent for a star network, local workers can only share their local solutions with a central node through multiples of intermediate nodes, which can possibly cause heavy communication delay and latency. For comparison, we solve machine learning problems including regression and classification over different communication networks having different delays with the following datasets: KDD Cup 1998 dataset[1], covertype dataset[2] [28], and wine quality dataset[3] [29]. In addition, we numerically check that the optimal number of local iterations and demonstrate the impact of communication delay on the convergence speed of the distributed dual coordinate ascent by varying the communication delay in networks.

We compare the convergence of the generalized distributed dual coordinate ascent in tree networks against that in star networks with intermediate nodes. Since the authors in [9, 10] compared the distributed dual coordinate ascent in a star network, so-called CoCoA, with other well known methods including mini-batch SDCA [30], local SGD and mini-batch-SGD [31], we focus to compare our generalized distributed dual coordinate ascent in tree networks with that in star networks having same local workers by considering network constraints, especially, communication delay and latency. Additionally, since we are interested in the communication network effects in the convergence speed of the synchronous distributed dual coordinate ascent,

[1]KDD Cup 1998 dataset: https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1998+Data

[2]Binary Covertype dataset: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#covtype.binary

[3]Wine quality dataset: https://archive.ics.uci.edu/ml/datasets/wine+quality

considering the CoCoA+ [10], which is the updated version of CoCoA, or an asynchronous updating method is out of scope of this paper.

A. Machine learning over communication networks

We consider both regression and classification problems with KDD Cup 1998 dataset and the covtype dataset over communication networks. In the communication networks, we assume that local workers cannot directly reach to a central node, and huge communication delay exists due to the long relay of communication path. In order to reflect this scenario, we deal with various communication delays between the central node and its direct child nodes.

1) ***KDD Cup 1998 regression problem****:* In this numerical experiment, we test our algorithm and analysis for a ridge regression problem with KDD Cup 1998 dataset having 481 attributions including a label and $95,412$ instances. We consider the following specific optimization problem by setting $\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) = (\frac{1}{\lambda m}\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2$:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\ \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\|\boldsymbol{A}^T\boldsymbol{w} - \boldsymbol{y}\|^2, \tag{18}$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ is the feature data matrix whose i-th column is $\frac{1}{\lambda m}\boldsymbol{x}_i$ and $\boldsymbol{y} \in \mathbb{R}^m$ is a label vector. Then, the following dual problem is obtained from (18):

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}}\ -\frac{\lambda}{2}\|\boldsymbol{A}\boldsymbol{\alpha}\|^2 - \lambda^2 m\sum_{i=1}^{m}\left(\frac{\alpha_i^2}{4} - \frac{y_i\alpha_i}{\lambda m}\right). \tag{19}$$

Hence, in a local worker, $\triangle\alpha$ in Procedure P is simply calculated as follows:

$$\triangle\alpha = -\left(\frac{\|\boldsymbol{x}_i\|^2}{\lambda m} + \frac{\lambda^2 m^2}{2}\right)^{-1}\left(\boldsymbol{w}^{(h-1)^T}\boldsymbol{x}_i + \frac{\lambda^2 m^2}{2}\alpha_i^{(h-1)} - \lambda m y_i\right), \tag{20}$$

where $(\boldsymbol{x}_i, y_i)$ is a randomly chosen data point and $\alpha_i^{(h-1)}$ is α_i value at $(h-1)$-th iteration.

For the dataset, we take first 95,410 instances and 404 numerical-type attributions for our numerical experiments. And then, we normalize each attribution with ℓ_2 norm of it for the performance of regression operation, and then normalize each instance with ℓ_2 norm in order to make each instance $\boldsymbol{x}_i$ hold the condition $\|\boldsymbol{x}_i\| \le 1$. We set the tuning parameter λ to 1. For the communication networks, we consider a tree network model having ten local workers, two sub-central nodes (each having five local workers), and one central node. The simulated star network has ten local workers and one center node. In both cases, we evenly split the data to ten local workers; namely, 9,541 instances without overlap are assigned to each local worker.

We set up a scenario where communication delay, t_{delay}, exists between the center node and its direct child node. Therefore, in a star network, the communication delay exists between the

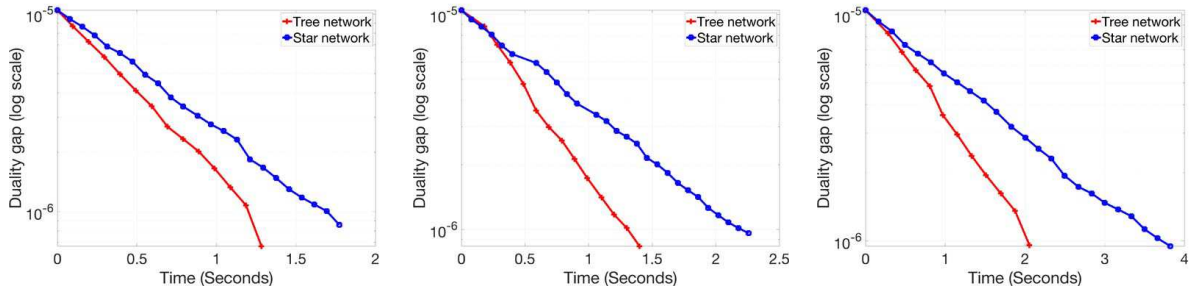
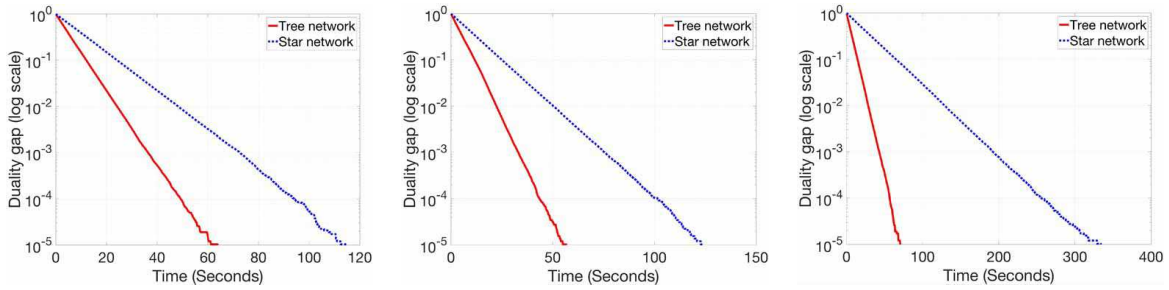


Fig. 5. Duality gap at the central node in a regression problem as the operation time of the algorithms goes. The distributed dual coordinate ascent in a tree network (red) and a star network (blue), i.e., CoCoA, are considered when the communication delay, t_{delay}, exists between the central node and its direct child nodes. $t_{delay} = r \times t_{lp}$, where t_{lp} represents the computational time for one local iteration at a local worker, and r represents the delay severity level.

central node and local workers, while a tree network has the delay between the central node and the sub-central node. We assume that communication delays between sub-central nodes and local workers are negligible. We set the communication delay $t_{delay} = r \times t_{lp}$, where t_{lp} is the computational time for one local iteration at a worker and the delay severity r is varied from 1 to 10^4. Hence, if the delay severity r is huge, then, there exists huge communication delay in the network when it is compared to the local processing time for one iteration. For the algorithm in the tree network, we set the number of local iterations in local workers and the number of communications between the local workers and the sub-central node to 1000 and 2 respectively. For the algorithm in the star network, the number of local iterations at local workers is set to 1000. Figure 5 shows the duality gap at the central node as the operation time goes, and demonstrates that as the communication delay severity increases, the gap between a tree network and a star network in the convergence speed of the distributed algorithm is increased, which indicates the distributed algorithm in a star network can suffer more from the communication delay effect.

*2) **Covertype dataset classification problem**:* We further conduct the comparison between the distributed dual coordinate scent in a star network and a tree network with a standard hinge loss ℓ_2 regularized SVM. We assume that the communication delay between the central node and its direct child nodes exists in the communcaiton networks. In this experiment, we use the preprocessed Covertype dataset [32], which is a binary classification dataset having 581,012 instances and 12 attributions including label information. The 12 attributions are expressed as 54 columns of data with 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. In order to satisfy the condition $\|\boldsymbol{x}_i\| \leq 1$, we normalize the dataset and $y_i \in \{-1, 1\}$,



(a) $r = 1$ (b) $r = 10^2$ (c) $r = 10^4$

Fig. 6. Duality gap at the central node in a classification problem as the operation time of the algorithms goes. The distributed dual coordinate ascent in a tree network (red solid line) and a star network (blue dotted line), i.e., CoCoA, are considered when the communication delay, t_{delay}, exists between the central node and its direct child nodes. $t_{delay} = r \times t_{lp}$, where t_{lp} represents the computational time for one local iteration at a worker, and r represents the delay severity level.

$i = 1, ..., m$. In this simulation, we organize a tree network having one central node, two sub-central nodes, and eight local workers. Each sub-central node has four local workers. Each local worker has evenly divided instances of the dataset without overlap. For the tree network, the number of communications between the local workers and the sub-central node is set to 10. The number of local iterations in both networks is set to 300.

For SVM, we consider the soft-margin SVM classification having hinge loss function, i.e, $\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) \triangleq \max(0, 1 - y_i(\frac{1}{\lambda m}\boldsymbol{w}^T\boldsymbol{x}_i))$ as follows:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}} \ \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\max(\boldsymbol{0}, \boldsymbol{1} - \boldsymbol{A}^T\boldsymbol{w}), \tag{21}$$

where $\boldsymbol{A}_i$, the i-th column of the matrix $\boldsymbol{A}$, is $\frac{1}{\lambda m}y_i\boldsymbol{x}_i$, $\max(\cdot)$ is element-wise operator, and $\boldsymbol{0} \in \mathbb{R}^m$ and $\boldsymbol{1} \in \mathbb{R}^m$ are the all 0 and all 1 vectors respectively.

Then, the dual problem of (21) is stated as follows:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \ \sum_{i=1}^{m}\alpha_i - \frac{1}{2\lambda}\|\boldsymbol{A}\boldsymbol{\alpha}\|^2 \qquad \text{subject to } \ 0 \le \alpha_i \le \frac{1}{m}, \ \forall i. \tag{22}$$

Note here that while deriving the dual problem (22), we have $\boldsymbol{w} = \frac{1}{\lambda}\boldsymbol{A}\boldsymbol{\alpha}$ as the dual-primal variable relation. Then, the local problem for a local worker Q is stated as follows:

$$\underset{\boldsymbol{\alpha}_Q\in\mathbb{R}^{|Q|}}{\text{maximize}} \ -\frac{\lambda}{2}\|\overline{\boldsymbol{w}} + \frac{1}{\lambda}\boldsymbol{A}_Q\boldsymbol{\alpha}_Q\|^2 + \sum_{i\in Q}\alpha_i + \sum_{i\in\overline{Q}}\alpha_i \qquad \text{subject to } \ 0 \le \alpha_i \le \frac{1}{m}, \ \forall i \in Q, \tag{23}$$

where $\overline{\boldsymbol{w}} \triangleq \frac{1}{\lambda}\boldsymbol{A}_{\overline{Q}}\boldsymbol{\alpha}_{\overline{Q}} = \boldsymbol{w} - \frac{1}{\lambda}\boldsymbol{A}_Q\boldsymbol{\alpha}_Q$. Then, in Procedure P for updating $\triangle\alpha$, we solve the following optimization problem:

$$\triangle\alpha = \underset{\triangle\alpha}{\text{argmax}} -\frac{\lambda}{2}\|\boldsymbol{w}^{(h-1)} + \frac{1}{\lambda^2 m}\triangle\alpha y_i\boldsymbol{x}_i\|^2 + (\alpha_i^{(h-1)} + \triangle\alpha) \quad \text{subject to } \ 0 \le \alpha_i^{(h-1)} + \triangle\alpha \le \frac{1}{m}. \tag{24}$$

Here, we update the randomly chosen i-th coordinate of $\boldsymbol{\alpha}$, where $i \in Q$. It is also possible to update the variable $\boldsymbol{\alpha}_Q$ with a block coordinate method. In order to solve (24), we calculate the optimal solution of (24) without the box constraint, i.e., $0 \leq \alpha_i^{(h-1)} + \triangle\alpha \leq \frac{1}{m}$, and then project the optimal solution onto the box constraint as follows:

$$\triangle\alpha = \begin{cases} 1/m - \alpha_i^{(h-1)} & \text{if } \alpha_i^{(h-1)} + \triangle\alpha > 1/m \\ -\alpha_i^{(h-1)} & \text{if } \alpha_i^{(h-1)} + \triangle\alpha < 0 \end{cases}. \tag{25}$$

Figure 6 shows the duality gap as the operation time of the algorithms goes. As shown in Figure 6, it is better to run more local iterations before sharing intermediate results with the central node when there is huge communication delay in a network.

B. Impact of communication delay on the convergence speed

In order to see the impact of the communication delay severity r, which is the ratio between the communication delay and the local processing time for one iteration, on the optimal number of local iterations H, we provide Figure 7 to show the optimal number of local iterations H by finding the critical point of (14). In the simulation, we set $(C, K, \delta, t_{total}, t_{lp}, t_{cp}) = (0.5, 3, 1/300, 1, 4 \times 10^{-5}, 3 \times 10^{-5})$. We set $t_{delay} = r \times t_{lp}$, where r is a parameter indicating how severe the communication delay is. Figure 7 (a) shows the objective values of (12) when H is varied from 1 to 2000. The red line represents the optimal convergence bound at the optimal number of local iterations, i.e., the critical point of (14) with different delay severity. Figure 7 (b) shows the optimal number of local iterations to achieve the fastest convergence rate in different communication delay severity, where r is varied from 1 to 10^5. The red dotted line is obtained by calculating the given analytical solution introduced in (17) with given aforementioned parameters, while the blue solid line is obtained by numerically calculating (12) and finding the optimal H which minimizes the objective value. This simulation results in Figure 7 show that when the delay severity becomes larger, the more local iterations are desired for the fast convergence speed of the overall algorithm.

In order to see the impact of the optimal local iterations on a practical machine learning problem, we similarly conduct a regression task with wine quality dataset [29] in a star network. For the number of iterations in local workers, we vary H from 1000 to 10000, and evaluate the convergence speed in terms of operation time and duality gap. Figures 8 (a) and (b) show the duality gap as the operation time goes when the delay severity levels r are set to 1 and 10^5 respectively. When $r = 1$, the fastest convergence is obtained at $H = 2000$, while when

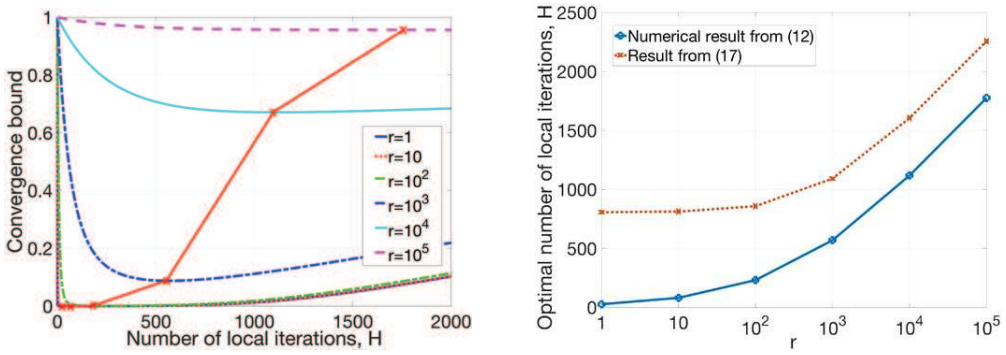


Fig. 7. (a) The objective value of (12), which is the convergence bound (or improvement), when the number of iterations H is varied from 1 to 2000, where $(C, K, \delta, t_{total}, t_{lp}, t_{cp}) = (0.5, 3, 1/300, 1, 4 \times 10^{-5}, 3 \times 10^{-5})$ and $t_{delay} = r \times t_{lp}$. The red line represents the optimal number of local iterations to achieve the fastest convergence rate. (b) Optimal number of iterations to achieve the fastest convergence rate, when the parameters are the same as (a) and r is varied from 1 to 10^5.

$r = 10^5$, the fastest convergence is obtained at $H = 10000$. As we expect in Section VI, when the communication delay is severe, it is better to perform the more local iterations before sharing the intermediate results with the central node. Also, if the communication delay is small, frequently sharing the intermediate results with the central node is helpful to improve the overall convergence speed. Moreover, we calculate the optimal number of iterations in local workers from the analytical solution (17) to see whether the analytical solution for the optimal number of local iterations fits to the simulation results. The parameters δ, K, and C are set to $\delta = 1/1000$, $K = 4$, and $C = 0.9$ by reflecting the network and simulation settings. With those parameter values, we obtain $2.1167e3$ for $r = 1$ and $6.0281e3$ for $r = 10^5$, while in the simulation, $H = 2000$ for $r = 1$ and $H = 10000$ for $r = 10^5$ provide the best convergence speed. Despite a little difference between the simulation result and the analytical solution for the optimal local number of iterations, (17) can still be used as a guideline for the number of local iterations in local workers.

C. Other parameters for faster convergence speed

In order to investigate the optimal number of local iterations which achieves the fastest convergence speed, from (17), we generate Figure 9 by varying each parameter r, C, K, and δ. In Figure 9(a), the communication delay severity parameter r is varied with fixed other parameters, $(C, K, \delta) = (0.5, 3, 1/300)$. As shown in Figure 9(a) and the previous subsection, when the communication delay severity r increases, the more number of local iterations before communication with the central node is desired for better convergence rate. Additionally, the

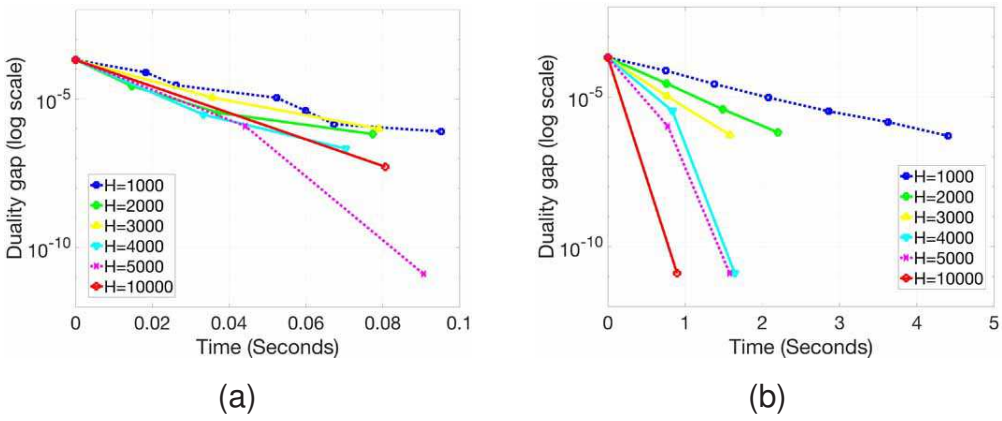


Fig. 8. (a) Duality gap when the delay severity r is 1. (b) Duality gap when the delay severity r is 10^5.

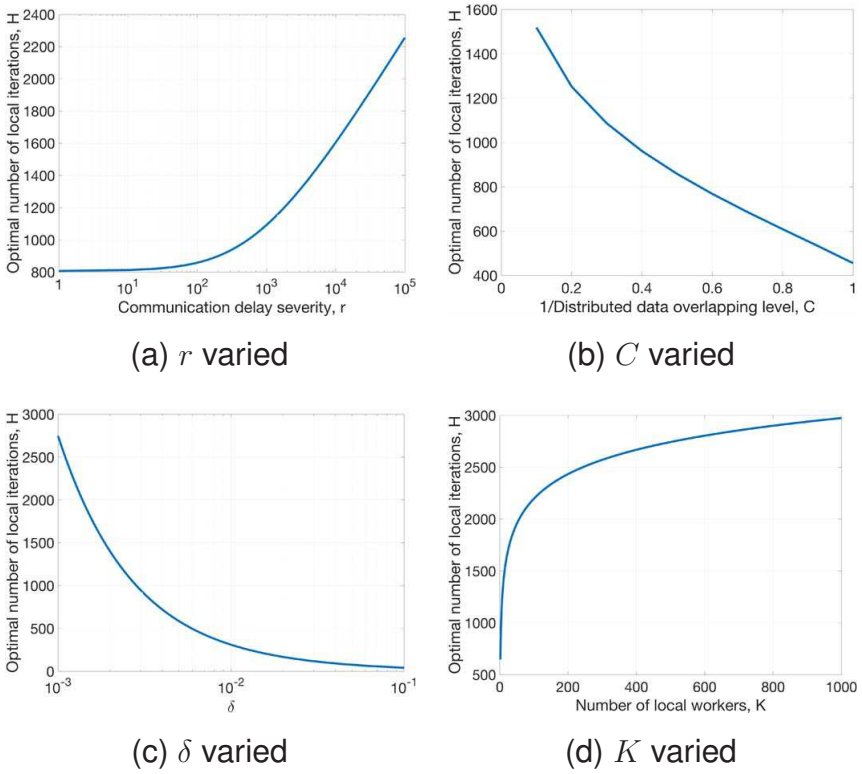


Fig. 9. Optimal number of local iterations, H, by varying parameters. Except for the varying parameter, other parameters are fixed to $(C, K, \delta, r) = (0.5, 3, 1/300, 100)$.

parameter ρ, which is reciprocal of the parameter C in (17), indicates the distributed data overlapping level; namely, smaller ρ, less overlapping data among local workers. In order to check the impact of the data overlapping level on the optimal local iteration H, we vary C with fixing other parameters to $(K, \delta, r) = (3, 1/300, 100)$, and draw the graph in Figure 9(b). From Figure 9(b), when local workers have more overlapping dataset among them. i.e., larger ρ value or smaller C value, it is desired to run more local iterations to have better convergence speed. And as δ decreases, correspondingly the step size of the algorithm in a local worker decreases,

the more number of local iterations is desired. This is understandable, because with a small step size, more iterations are needed to reach an optimal point. From Figure 9(d), as the number of local workers, K, increases, the optimal number of local iterations, H, is also increased. Since we fixed other parameters except for K, increasing K represents increasing the total size of dataset. And due to the bigger size of dataset in total, we think that more variance in the intermediate results from local workers may lead to more local iterations to reduce the variance.

VIII. Conclusion and discussion

In this paper, we study the distributed dual coordinate ascent in a general tree-structured network, where a central node, sub-central nodes and local workers are connected over the communication network, and its analysis. Additionally, since the communication becomes a bottleneck in distributed network systems, we consider the communication delay in time in the convergence analysis of the distributed dual coordinate ascent and obtain the optimal number of iterations to achieve the best convergence speed. In the numerical experiments, we demonstrate the usability of our algorithm and analysis in synchronous machine learning scenarios over communication networks where local workers cannot directly reach to a central node due to communication constraints.

Addition to the work in the paper, the following topics are possible directions for the future research. We leave them for the future research.

- **Asynchronous updating scheme:** Due to the possible performance difference among local workers, it is quite natural to consider asynchronous scheme. Thus, the design and analysis of asynchronous dual coordinate ascent algorithm for generalized tree network topologies can be the next direction of the research.
- **Different network topologies:** Since every connected network has its spanning tree, in this paper, a general tree network topology is studied. However, in some network models organized in a mesh, thanks to the network connections in a mesh, the intermediate results from local workers can be easily shared with sub-central nodes and central node or even between local workers. Therefore, the distributed algorithm in mesh networks can have potential to have faster convergence speed than the algorithm in tree networks. Thus, studying distributed algorithms in mesh networks is of great interest for distributed machine learning operations.

- **Various network constraints:** The communication networks can have a variety of network constraints including communication delay, limited communication bandwidth, and limited transmission power. Motivated by these network constraints, the impact of communication delay on the convergence speed of distributed dual coordinate ascent is studied in this paper. It is also interesting to study the other communication constraints in distributed algorithms.

REFERENCES

[1] M. Cho, L. Lai, and W. Xu, "Generalized distributed dual coordinate ascent in a tree network for machine learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3512–3516. 1

[2] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014. 2

[3] J. P. Verma, B. Patel, and A. Patel, "Big data analysis: recommendation system with hadoop framework," in *Proceedings of IEEE International Conference on Computational Intelligence & Communication Technology*, 2015, pp. 92–97. 2

[4] J. Andreu-Perez, C. Poon, R. D. Merrifield, S. Wong, and G.-Z. Yang, "Big data for health," *IEEE journal of biomedical and health informatics*, vol. 19, no. 4, pp. 1193–1208, 2015. 2

[5] S. Efromovich, J. Lakey, M. C. Pereyra, and N. Tymes, "Data-driven and optimal denoising of a signal and recovery of its derivative using multiwavelets," *IEEE Transactions on Signal Processing*, vol. 52, no. 3, pp. 628–635, 2004. 2

[6] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," in *Proceedings of the International Conference on Learning Representations Workshop Track*, 2016. 2

[7] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 69–77. 2

[8] T. Yang, "Trading computation for communication: Distributed stochastic dual coordinate ascent," in *Advances in Neural Information Processing Systems*, 2013, pp. 629–637. 2, 3, 4, 5, 13

[9] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Advances in Neural Information Processing Systems*, 2014, pp. 3068–3076. 2, 3, 4, 5, 6, 7, 11, 12, 13, 16, 25

[10] C. Ma, V. Smith, M. Jaggi, M. I. Jordan, P. Richtárik, and M. Takáč, "Adding vs. averaging in distributed primal-dual optimization," in *Proceedings of the International Conference on Machine Learning*, 2015, vol. 37, pp. 1973–1982. 2, 3, 4, 5, 7, 13, 16, 17

[11] S.-Y. Zhao and W.-J. Li, "Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, pp. 2379–2385. 2

[12] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proceedings of International Conference on Machine Learning*. ACM, 2008, pp. 408–415. 2

[13] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, pp. 567–599, 2013. 2, 4, 5, 29

[14] R. Zhang, S. Zheng, and J. T. Kwok, "Fast distributed asynchronous SGD with variance reduction," *CoRR, abs/1508.01633*, 2015. 2

[15] Z. Huo and H. Huang, "Distributed asynchronous dual free stochastic dual coordinate ascent," in *Proceedings of the IEEE International Conference on Data Mining*, 2018. 2, 4, 5

[16] C.-J. Hsieh, H.-F. Yu, and I. S. Dhillon, "PASSCoDe: Parallel asynchronous stochastic dual co-ordinate descent," in *Proceedings of the International Conference on Machine Learning*, 2015, vol. 15, pp. 2370–2379. 2, 4

[17] K. Tsianos, S. Lawlor, and M. G. Rabbat, "Communication/computation tradeoffs in consensus-based distributed optimization," in *Advances in Neural Information Processing Systems*, 2012, pp. 1943–1951. 3, 14

[18] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, 2019. 3, 4

[19] T.-H. 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. 3, 4

[20] 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. 3, 4

[21] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010. 3, 4

[22] M. Hong and T. Chang, "Stochastic proximal gradient consensus over random networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 11, pp. 2933–2948, 2017. 3, 4

[23] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004. 5

[24] A. Nedic, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018. 14

[25] T. T. Doan, C. L. Beck, and R. Srikant, "Impact of communication delays on the convergence rate of distributed optimization algorithms," *arXiv preprint arXiv:1708.03277*, 2017. 14

[26] K. I. Tsianos, *The Role of the Network in Distributed Optimization Algorithms: Convergence Rates, Scalability, Communication/Computation Tradeoffs and Communication Delays*, Ph.D. thesis, McGill University, 2013. 14

[27] R. M. Corless, G. H. Gonnet, D. EG. Hare, D. J. Jeffrey, and D. E. Knuth, "On the LambertW function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359, 1996. 15

[28] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. 24, no. 3, pp. 131–151, 1999. 16

[29] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009. 16, 20

[30] M. Takáč, A. Bijral, P. Richtárik, and N. Srebro, "Mini-batch primal and dual methods for SVMs," in *Proceedings of the International Conference on Machine Learning*, 2013, vol. 28, pp. III–1022–III–1030. 16

[31] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 2011. 16

[32] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems," in *Advances in Neural Information Processing Systems*, 2002, pp. 633–640. 18

APPENDIX A
PROOF OF THEOREM 2

For this proof, we follow the proof of Theorem 2 of [9] with the additional difference, i.e., dealing with both updating coordinates $\boldsymbol{\alpha}_Q$ and un-updating coordinates $\boldsymbol{\alpha}_{\overline{Q}}$, and show that for a general tree node Q, the convergence analysis introduced in (8) holds.

Proof. Suppose the tree node Q has K direct child nodes, and we simply represent the child nodes from 1 to K. The convergence rate of the algorithm at a tree node Q is obtained by considering the updating scheme at the node Q as follows.

$$\boldsymbol{\alpha}^{(t+1)} = (\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}) = (\boldsymbol{\alpha}_{[1:K]}^{(t)} + \frac{1}{K}\sum_{k=1}^{K} \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}), \tag{26}$$

where $\boldsymbol{\alpha}_{<[k]>}$ is the zero-padding version of $\boldsymbol{\alpha}_{[k]}$ and $Q = [1:K] = \cup_{k=1}^{K}[k]$ is the index set corresponding to workers connected to the node Q. The optimal value at the node Q is stated as

$$\begin{aligned} D(\boldsymbol{\alpha}_Q, \boldsymbol{\alpha}_{\overline{Q}}) &= -\frac{\lambda}{2}\|\boldsymbol{A}_Q\boldsymbol{\alpha}_Q + \boldsymbol{A}_{\overline{Q}}\boldsymbol{\alpha}_{\overline{Q}}\|^2 - \frac{1}{m}\sum_{i\in Q}\ell_i^*(-\alpha_i) - \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^*(-\alpha_i) \\ &= -\frac{\lambda}{2}\|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \frac{1}{m}\sum_{i\in[1:K]}\ell_i^*(-\alpha_i) - \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^*(-\alpha_i), \end{aligned}$$

where $\boldsymbol{A}_Q$ is the partial matrix of $\boldsymbol{A}$ by choosing the columns of $\boldsymbol{A}$ over the index set Q, and $\boldsymbol{A}_{\overline{Q}}\boldsymbol{\alpha}_{\overline{Q}}$ is denoted as $\overline{\boldsymbol{w}}$. From (26), we have

$$\begin{aligned} D\big(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}\big) &= D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \frac{1}{K}\sum_{k=1}^{K}\triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}\big) = D\big(\frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}), \boldsymbol{\alpha}_{\overline{Q}}\big) \\ &\geq \frac{1}{K}\sum_{k=1}^{K} D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}\big), \end{aligned}$$

where the inequality is obtained from the Jensen's inequality. Then, we have

$$\begin{aligned} D\big(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}\big) &\geq \frac{1}{K}\sum_{k=1}^{K}\Big[D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}\big)\Big] \\ &= \frac{1}{K}\sum_{k=1}^{K}\Big[D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D((\boldsymbol{\alpha}_{[Q;1]}^{(t)}, ..., \boldsymbol{\alpha}_{[Q;k]}^{\star}, ..., \boldsymbol{\alpha}_{[Q;K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})) \\ &\quad + D((\boldsymbol{\alpha}_{[Q;1]}^{(t)}, ..., \boldsymbol{\alpha}_{[Q;k]}^{\star}, ..., \boldsymbol{\alpha}_{[Q;K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\Big] \\ &= \frac{1}{K}\sum_{k=1}^{K}\Big[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}) - \epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})\Big], \end{aligned}$$

where $\epsilon_{Q,k}(\cdot)$ is defined in (5) and the super-script $\star$ represents the optimal solution. Then, the expectation of $D\big(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}\big)$ is lower-bounded as follows:

$$\begin{aligned} \mathbb{E}[D(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})] &\geq \frac{1}{K}\sum_{k=1}^{K}\Big[\mathbb{E}[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})] - \mathbb{E}[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})]\Big] \\ &\geq \frac{1}{K}(1-\Theta)\sum_{k=1}^{K}\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}), \end{aligned}$$

where the last inequality is obtained from Assumption 1. And $\sum_{k=1}^{K}\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})$ can be bounded as follows.

$$
\begin{aligned}
&\sum_{k=1}^{K} \epsilon_{Q,k}(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \\
&= \sum_{k=1}^{K} \underset{\hat{\boldsymbol{\alpha}}_{[Q;k]}}{\text{maximize}} \left[D((\boldsymbol{\alpha}^{(t)}_{[Q;1]}, ..., \hat{\boldsymbol{\alpha}}_{[Q;k]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}})) - D((\boldsymbol{\alpha}^{(t)}_{[Q;1]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;k]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}})) \right] \\
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} \sum_{k=1}^{K} \left[D((\boldsymbol{\alpha}^{(t)}_{[Q;1]}, ..., \hat{\boldsymbol{\alpha}}_{[Q;k]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}})) - D((\boldsymbol{\alpha}^{(t)}_{[Q;1]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;k]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}})) \right]
\end{aligned}
$$

$$
\begin{aligned}
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} \sum_{k=1}^{K} \left[-\frac{\lambda}{2} \|\boldsymbol{A}_{[1:K]}(\boldsymbol{\alpha}^{(t)}_{[Q;1]}, ..., \hat{\boldsymbol{\alpha}}_{[Q;k]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;K]}) + \overline{\boldsymbol{w}}\|^2 + \frac{\lambda}{2} \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \right] \\
&\qquad - \frac{1}{m} \sum_{i \in [1:K]} \ell_i^*(-\hat{\alpha}_i) + \frac{1}{m} \sum_{i \in [1:K]} \ell_i^*(-\alpha_i^{(t)}) \\
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} \left[\frac{1}{m} \sum_{i \in [1:K]} \left(-\ell_i^*(-\hat{\alpha}_i) + \ell_i^*(-\alpha_i^{(t)}) \right) \right] - \frac{\lambda}{2} \sum_{k=1}^{K} \left[\|\boldsymbol{A}_{[1:K]}(\boldsymbol{\alpha}^{(t)}_{[Q;1]}, ..., \hat{\boldsymbol{\alpha}}_{[Q;k]}, ..., \boldsymbol{\alpha}^{(t)}_{[Q;K]}) + \overline{\boldsymbol{w}}\|^2 \right. \\
&\qquad \left. - \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \right] \\
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} \left[-\frac{1}{m} \sum_{i \in [1:K]} \left(\ell_i^*(-\hat{\alpha}_i) - \ell_i^*(-\alpha_i^{(t)}) \right) \right] - \frac{\lambda}{2} \sum_{k=1}^{K} \left[\|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} - \boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]}) + \overline{\boldsymbol{w}}\|^2 \right. \\
&\qquad \left. - \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \right] \\
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} \left[D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + \frac{\lambda}{2} \|\boldsymbol{A}_{[1:K]} \hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - \frac{\lambda}{2} \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \right] \\
&\qquad - \frac{\lambda}{2} \sum_{k=1}^{K} \left[\|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} - \boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]}) + \overline{\boldsymbol{w}}\|^2 - \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \right] \\
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + \frac{\lambda}{2} \|\boldsymbol{A}_{[1:K]} \hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \frac{\lambda}{2} \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \\
&\qquad - \frac{\lambda}{2} \sum_{k=1}^{K} \left[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2 - 2(\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}})^T \boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]}) \right] \\
&= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + \frac{\lambda}{2} \left(\|\boldsymbol{A}_{[1:K]} \hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \|\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 \right) \\
&\qquad - \frac{\lambda}{2} \sum_{k=1}^{K} \left[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2 \right] + \lambda(\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}})^T (\boldsymbol{A}_{[1:K]} \boldsymbol{\alpha}^{(t)}_{[1:K]} - \boldsymbol{A}_{[1:K]} \hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}} - \overline{\boldsymbol{w}}) \quad (27)
\end{aligned}
$$

$$
\begin{aligned}
(27) = \underset{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^{|[1:K]|}}{\text{maximize}}\; & D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + \frac{\lambda}{2}\left(\|\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2\right) \\
& - \frac{\lambda}{2}\sum_{k=1}^{K}\left[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2\right] + \lambda\|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \lambda(\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}})^T(\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}) \\
= \underset{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^{|[1:K]|}}{\text{maximize}}\; & D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - \frac{\lambda}{2}\sum_{k=1}^{K}\left[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2\right] \\
& + \frac{\lambda}{2}\left(\|\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 + \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - 2(\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}^{(t)}_{[1:K]} + \overline{\boldsymbol{w}})^T(\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}})\right) \\
= \underset{\hat{\boldsymbol{\alpha}}\in\mathbb{R}^{|[1:K]|}}{\text{maximize}}\; & D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - \underbrace{\frac{\lambda}{2}\left[\sum_{k=1}^{K}\left[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2\right] - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]})\|^2\right]}_{=(A)}
\end{aligned}
\tag{28}
$$

We can lower-bound (28) by upper-bounding (A). For the upper-bound of (A), we have

$$
\begin{aligned}
(A) &= \sum_{k=1}^{K}\left[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}^{(t)}_{[k]} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2\right] - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]})\|^2 \\
&\le \sum_{i\in[1:K]}\|\boldsymbol{A}_i(\alpha^{(t)}_i - \hat{\alpha}_i)\|^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]})\|^2 \\
&\le \sum_{i\in[1:K]}\frac{1}{\lambda^2 m^2}\|\boldsymbol{x}_i\|^2(\alpha^{(t)}_i - \hat{\alpha}_i)^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]})\|^2 \\
&\le \frac{1}{\lambda^2 m^2}\sum_{i\in[1:K]}(\alpha^{(t)}_i - \hat{\alpha}_i)^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]})\|^2 \\
&\le \frac{1}{\lambda^2 m^2}\|\boldsymbol{\alpha}^{(t)}_{[1:K]} - \hat{\boldsymbol{\alpha}}_{[1:K]}\|^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]})\|^2 \\
&\le \frac{\rho}{\lambda^2 m^2}\|\boldsymbol{\alpha}^{(t)}_{[1:K]} - \hat{\boldsymbol{\alpha}}_{[1:K]}\|^2,
\end{aligned}
$$

where the second inequality is from $\boldsymbol{A}_i = \frac{1}{\lambda m}\boldsymbol{x}_i$, and the third inequality is obtained from the assumption of the scaled input data, i.e., $\|\boldsymbol{x}_i\| \le 1$. We can have the last inequality by introducing ρ_{min}, which is the minimum value of ρ, to hold the last inequality as follows:

$$
\rho \ge \rho_{min} \triangleq \underset{\boldsymbol{\alpha}\in\mathbb{R}^{|[1:K]|}}{\text{maximize}}\; \lambda^2 m^2 \frac{\sum_{k=1}^{K}\|\boldsymbol{A}_{[k]}\boldsymbol{\alpha}_{[k]}\|^2 - \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}\|^2}{\|\boldsymbol{\alpha}\|^2} \ge 0. \tag{29}
$$

The condition $\rho_{min} \ge 0$ can be shown by considering a feasible solution making $\sum_{k=1}^{K}\|\boldsymbol{A}_{[k]}\boldsymbol{\alpha}_{[k]}\|^2 - \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}\|^2 = 0$, e.g., $\boldsymbol{\alpha} = \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is a standard unit vector having 1 in the i-th entry and 0 elsewhere.

Then, (28), which is $\sum_{k=1}^{K}\epsilon_{Q,k}(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}})$, is lower-bounded as follows:

$$
\begin{aligned}
&\sum_{k=1}^{K} \epsilon_{Q,k}(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \\
&\geq \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}}\; D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - \frac{\rho}{2\lambda m^2}\|\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2 \\
&\geq \underset{\eta \in [0,1]}{\text{maximize}}\; D\big(\eta\boldsymbol{\alpha}^{\star}_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - \frac{\rho}{2\lambda m^2}\|\eta\boldsymbol{\alpha}^{\star}_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2 \\
&\geq \underset{\eta \in [0,1]}{\text{maximize}}\; \eta D\big(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + (1-\eta) D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + \frac{\gamma\eta(1-\eta)}{2m}\|\boldsymbol{\alpha}^{\star}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2 \\
&\qquad - \frac{\rho\eta^2}{2\lambda m^2}\|\boldsymbol{\alpha}^{\star}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2 \\
&\geq \underset{\eta \in [0,1]}{\text{maximize}}\; \eta D\big(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - \eta D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + \frac{\gamma\eta(1-\eta)}{2m}\|\boldsymbol{\alpha}^{\star}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2 - \frac{\rho\eta^2}{2\lambda m^2}\|\boldsymbol{\alpha}^{\star}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2 \\
&= \underset{\eta \in [0,1]}{\text{maximize}}\; \eta D\big(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - \eta D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + \frac{\eta}{2m}\left(\gamma - \frac{\lambda m\gamma + \rho}{\lambda m}\eta\right)\|\boldsymbol{\alpha}^{\star}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2, \qquad (30)
\end{aligned}
$$

where η in the second inequality is introduced for line search between the optimal solution $\boldsymbol{\alpha}^{\star}_{[1:K]}$ and $\boldsymbol{\alpha}^{(t)}_{[1:K]}$, and the equality holds when $\hat{\boldsymbol{\alpha}}_{[1:K]}$ is in the line between $\boldsymbol{\alpha}^{\star}_{[1:K]}$ and $\boldsymbol{\alpha}^{(t)}_{[1:K]}$. And the third inequality is obtained from the strong concavity of $D(\boldsymbol{\alpha})$. Specifically, we use the well-known fact that if a function $\ell_i(a)$ is $\frac{1}{\gamma}$-smooth, the conjugate function $\ell_i^{\star}$ is γ strongly convex: for all $u, v \in \mathbb{R}$ and $\eta \in [0,1]$ [13]:

$$
-\ell_i^{\star}(\eta u + (1-\eta)v) \geq -\eta\ell_i^{\star}(u) - (1-\eta)\ell_i^{\star}(v) + \frac{\gamma\eta(1-\eta)}{2}(u-v)^2. \qquad (31)
$$

From (31), we have the following inequality for $D\big(\eta\boldsymbol{\alpha}^{\star}_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)$:

$$
\begin{aligned}
&D\left(\eta\boldsymbol{\alpha}^{\star}_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right) \\
&= -\frac{1}{2}\left\|\boldsymbol{A}\big(\eta\boldsymbol{\alpha}^{\star}_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]},\ \boldsymbol{\alpha}_{\overline{Q}}\big)\right\|^2 - \frac{1}{m}\sum_{i\in[1:K]}\ell_i^{\star}(-\eta\alpha_i^{\star} - (1-\eta)\alpha_i^{(t)}) - \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^{\star}(-\eta\alpha_i - (1-\eta)\alpha_i) \\
&= -\frac{1}{2}\left\|\eta\boldsymbol{A}\big(\boldsymbol{\alpha}^{\star}_{[1:K]},\ \boldsymbol{\alpha}_{\overline{Q}}\big) + (1-\eta)\boldsymbol{A}\big(\boldsymbol{\alpha}^{(t)}_{[1:K]},\ \boldsymbol{\alpha}_{\overline{Q}}\big)\right\|^2 - \frac{1}{m}\sum_{i\in[1:K]}\ell_i^{\star}(-\eta\alpha_i^{\star} - (1-\eta)\alpha_i^{(t)}) - \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^{\star}(-\eta\alpha_i - (1-\eta)\alpha_i)
\end{aligned}
\qquad (32)
$$

$$
\begin{aligned}
(32) &\overset{(31)}{\geq} -\frac{1}{2}\left\|\eta \boldsymbol{A}\left(\boldsymbol{\alpha}^{\star}_{[1:K]},\, \boldsymbol{\alpha}_{\overline{Q}}\right) + (1-\eta)\boldsymbol{A}\left(\boldsymbol{\alpha}^{(t)}_{[1:K]},\, \boldsymbol{\alpha}_{\overline{Q}}\right)\right\|^2 \\
&\quad - \frac{1}{m}\sum_{i\in[1:K]}\left[\eta\ell_i^*(-\alpha_i^\star) + (1-\eta)\ell_i^*(-\alpha_i^{(t)}) - \frac{\gamma\eta(1-\eta)}{2}(\alpha_i^\star - \alpha_i^{(t)})^2\right] - \frac{1}{m}\sum_{i\in\overline{Q}}\left[\eta\ell_i^*(-\alpha_i) + (1-\eta)\ell_i^*(-\alpha_i)\right] \\
&\geq -\frac{\eta}{2}\left\|\boldsymbol{A}\left(\boldsymbol{\alpha}^{\star}_{[1:K]},\, \boldsymbol{\alpha}_{\overline{Q}}\right)\right\|^2 - \frac{(1-\eta)}{2}\left\|\boldsymbol{A}\left(\boldsymbol{\alpha}^{(t)}_{[1:K]},\, \boldsymbol{\alpha}_{\overline{Q}}\right)\right\|^2 \\
&\quad - \frac{1}{m}\sum_{i\in[1:K]}\left[\eta\ell_i^*(-\alpha_i^\star) + (1-\eta)\ell_i^*(-\alpha_i^{(t)}) - \frac{\gamma\eta(1-\eta)}{2}(\alpha_i^\star - \alpha_i^{(t)})^2\right] - \frac{1}{m}\sum_{i\in\overline{Q}}\left[\eta\ell_i^*(-\alpha_i) + (1-\eta)\ell_i^*(-\alpha_i)\right] \\
&= -\frac{\eta}{2}\left\|\boldsymbol{A}\left(\boldsymbol{\alpha}^{\star}_{[1:K]},\, \boldsymbol{\alpha}_{\overline{Q}}\right)\right\|^2 - \frac{\eta}{m}\left[\sum_{i\in[1:K]}\ell_i^*(-\alpha_i^\star) + \sum_{i\in\overline{Q}}\ell_i^*(-\alpha_i)\right] - \frac{(1-\eta)}{2}\left\|\boldsymbol{A}\left(\boldsymbol{\alpha}^{(t)}_{[1:K]},\, \boldsymbol{\alpha}_{\overline{Q}}\right)\right\|^2 \\
&\quad - \frac{(1-\eta)}{m}\left[\sum_{i\in[1:K]}\ell_i^*(-\alpha_i^{(t)}) + \sum_{i\in\overline{Q}}\ell_i^*(-\alpha_i)\right] + \frac{\gamma\eta(1-\eta)}{2m}\sum_{i\in[1:K]}(\alpha_i^\star - \alpha_i^{(t)})^2 \\
&= \eta D(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + (1-\eta)D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + \frac{\gamma\eta(1-\eta)}{2m}\|\boldsymbol{\alpha}^{\star}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\|^2.
\end{aligned}
$$

Notice that $\eta \in [0,1]$. Also note that we derive the equations by using $\boldsymbol{A}(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}})$; however, at each node, we do not know $\boldsymbol{A}_{\overline{Q}}$, but $\overline{\boldsymbol{w}}$. Therefore, for the term $\boldsymbol{A}(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}})$, $(\boldsymbol{A}_Q\boldsymbol{\alpha}^{\star}_{[1:K]} + \overline{\boldsymbol{w}})$ is the correct notation; however in order to clearly show the dual objective function, we use the term $\boldsymbol{A}(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}})$ instead of $(\boldsymbol{A}_Q\boldsymbol{\alpha}^{\star}_{[1:K]} + \overline{\boldsymbol{w}})$ with which the derivation can also go through.

(30) can be lower-bounded by choosing $\eta = \frac{\lambda m\gamma}{\lambda m\gamma + \rho} \geq 0$ as

$$
(30) \geq \frac{\lambda m\gamma}{\lambda m\gamma + \rho}\left(D\left(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right) - D\left(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right)\right)
$$

Therefore, we have

$$
\begin{aligned}
\mathbb{E}[D(\boldsymbol{\alpha}^{(t+1)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \mid \overline{\boldsymbol{w}}, \boldsymbol{\alpha}^{(t)}_{[1:K]}] &\geq \frac{1}{K}(1-\Theta)\sum_{k=1}^{K}\epsilon_{Q,k}(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \\
&\geq \frac{1}{K}(1-\Theta)\frac{\lambda m\gamma}{\lambda m\gamma + \rho}\left(D\left(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right) - D\left(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right)\right) \quad (33)
\end{aligned}
$$

From (33), we have

$$
\begin{aligned}
&\mathbb{E}[D(\boldsymbol{\alpha}^{(t+1)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) + D(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \mid \overline{\boldsymbol{w}}, \boldsymbol{\alpha}^{(t)}_{[1:K]}] \\
&\geq \frac{1}{K}(1-\Theta)\frac{\lambda m\gamma}{\lambda m\gamma + \rho}\left(D\left(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right) - D\left(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right)\right).
\end{aligned}
$$

By moving the term $D(\boldsymbol{\alpha}^{\star}_{[1:K]}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}})$ in LHS to RHS and multiplying -1 in both sides, we have

$$
\mathbb{E}[D(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t+1)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \mid \boldsymbol{\alpha}^{(t)}_{[1:K]}, \overline{\boldsymbol{w}}] \leq \left(1 - \frac{1}{K}(1-\Theta)\frac{\lambda m\gamma}{\lambda m\gamma + \rho}\right)\left(D\left(\boldsymbol{\alpha}^{\star}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right) - D\left(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right)\right)
$$

□

APPENDIX B
DERIVATION OF THE OPTIMAL NUMBER OF LOCAL ITERATIONS H

For the sake of simplicity of (14), by denoting $1-\delta$, $\frac{K-C}{K}$, $\frac{C}{K}$, and $(t_{delay}+t_{cp})/t_{lp}$ to a, b, c, and r respectively, we have the following first order condition over H for given a, b, c, and r:

$$b(H+r)a^H \ln(a) - (b+ca^H)\ln(b+ca^H) = 0, \tag{34}$$

where $a, b, c \in [0,1)$ and $b+c=1$. When H is large enough, $b(H+r)a^H\ln(a)$ is the dominant term of (34) and notice that $0<a<1$. Therefore, by approximating the term $(b+ca^H)\ln(b+ca^H)$ to $b\ln(b)$, we have $b(H+r)a^H\ln(a) = b\ln(b)$. And then, the equation is re-stated as follows:

$$(H+r)\ln(a)e^{H\ln(a)} = \ln(b) \Rightarrow (H+r)\ln(a)e^{(H+r)\ln(a)} = \ln(b)e^{r\ln(a)}$$

From the definition of the Lambert W-function, which is when $xe^x = a$, the solution x is $W(a)$, where $W(\cdot)$ is the Lambert W-function, we have

$$(H+r)\ln(a) = W\big(\ln(b)e^{r\ln(a)}\big).$$

Therefore, for the optimal number of local iterations H, we have

$$H = \frac{1}{\ln(a)}W(\ln(b)e^{r\ln(a)}) - r.$$