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 study the convergence rate of the distributed dual coordinate ascent for distributed machine learning problems in a general tree-structured network. Since a tree network model can be understood as the generalization of a star network, our algorithm can be thought of as the generalization of the distributed dual coordinate ascent in a star network. We provide the convergence rate of the distributed dual coordinate ascent over a general tree network in a recursive manner and analyze the network effect on the convergence rate. Secondly, by considering network communication delays, we optimize the distributed dual coordinate ascent algorithm to maximize its convergence speed. From our analytical result, we can choose the optimal number of local iterations depending on the communication delay severity to achieve the fastest 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

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 [1], e.g., recommender services [2], disease diagnosis and analysis [3], or even signal recovery [4]. However, due to limited storage volumes in storage servers 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

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).

Lifeng Lai's work was partially supported by the National Science Foundation (NSF) under grants CCF-1717943, ECCS-2000415. Weiyu Xu's research was partially funded by NSF ECCS-2000425.

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 [5, 6]. 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 [7–17] and the references therein. More specially, synchronous Stochastic Gradient Decent (SGD) [7, 8], synchronous Stochastic Dual Coordinate Ascent (SDCA) [9–11, 13, 14], asynchronous SGD [12, 15], and asynchronous SDCA [16, 17] for distributed data have been intensively investigated in the literature. Among them, [13] 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, [9–11] 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 [9–11] analyze the convergence rate of the distributed SDCA in terms of communication rounds. Especially, the strong aspects of the proposed distributed optimization framework in [10, 11] 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 important, 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 [9–11] in terms of overall computational time instead of the number of communication rounds. The authors in [18] analyzed the convergence bound in terms of time by considering communication delays in a network for a consensus optimization problem. Additionally, the research [19–23] studied separable consensus problems to each worker by using ADMM techniques. We remark that the regularized loss minimization problem considered in [9–11] is a different problem from the consensus problems considered in [19–23] in the aspect of separability. Moreover, in [24–26], the authors considered a distributed deep neural network model. By introducing auxiliary variables, the authors made the non-convex problem separable, which can lead to a consensus problem over a network. Unlike the works in [24–26], we consider to solve distributed machine learning problems in an augmented manner by taking into account communication networks. Therefore, our work focuses on the communication and network topology's effect on the distributed machine learning algorithms, while the works in [24–26] focus on the alternating or block coordinate descent algorithm itself to solve neural network problems with auxiliary variables without considering network constraints.

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 ratio 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. The proposed algorithm and its convergence rate without a proof were introduced in our previous conference paper [27]. In this journal paper, we provide the full proof of our theorem in Appendix A, the analysis of network topology and communication effect on the algorithm in Sections V and VI respectively, and additional numerical experiments in Section VII.

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 [9–11, 14, 16, 17]:

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

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 [19–23], 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., $\|\boldsymbol{x}_i\| \le 1$, $i = 1, 2, ..., m$, and the dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ is stored in a distributed manner over a network having K local workers. More 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 $|\cup_{k=1}^{K} [k]| = m$.

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 $a, b \in \mathbb{R}$:

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

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 [10, 14], 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 [28], 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). Additionally, if the loss function $\ell_i(\cdot)$ is non-convex, the primal problem will become a non-convex problem. However, the dual problem is still expressed in a convex problem [28]. Therefore, our algorithm to tackle the dual problem can provide an optimal solution to the dual problem. Unfortunately, in this case, there is no guarantee that the optimal solution to the dual problem becomes an optimal solution to the primal problem.

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 [9–11, 16], where a star network topology for the network is considered as shown in Figure 1. In particular, the authors in [10] introduced a distributed dual coordinate ascent framework, called the Communication-Efficient Distributed Dual Coordinate Ascent (CoCoA), and later proposed CoCoA+ [11], 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 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 [10].

Suppose a star network has K local workers and each local worker has disjoint parts of dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$. With this problem setting, the authors in [10] 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($\cdot$), which represents any dual method to solve (2), e.g. Stochastic Dual Coordinate Ascent (SDCA), simply denoted by LocalSDCA($\cdot$), 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
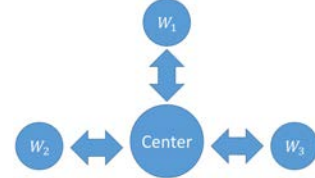

Fig. 1. Illustration of a star network having one central station and three local workers W_1, W_2 and W_3.

Algorithm 1: Communication-efficient Distributed Dual Coordinate Ascent (CoCoA) [10]

Input: $T \geq 1$
Output: $\boldsymbol{w}$, $\boldsymbol{\alpha}$
Data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ distributed over K local workers
Initialization: $\boldsymbol{\alpha}_{[k]}^{(0)} \leftarrow \boldsymbol{0}$ for all local workers, and $\boldsymbol{w}^{(0)} \leftarrow \boldsymbol{0}$
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

---

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 [10].

Theorem 1 ([10, Theorem 2])**.** *Suppose that Algorithm 1 is run for T outer iterations of K local computers with the procedure LocalSDCA($\cdot$) 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:*

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

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

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

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

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

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($\cdot$), 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 [11] proposed CoCoA+, which has the same
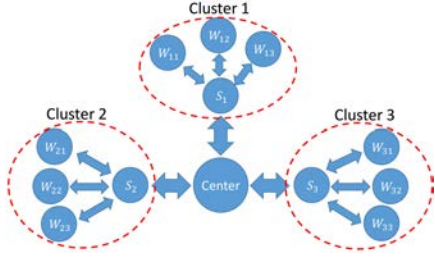
Fig. 2. Illustration of a tree-structured network, which has two layers. In the network, a central station (root node) has three direct child nodes S_1, S_2 and S_3. Each node S_i has three direct child nodes W_{ij}, $j = 1, 2, 3$.

Fig. 3. Illustration of a subtree including a node Q on the i-th layer and its direct and indirect child nodes.

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 as a direct virtual-link through intermediate nodes. However, since communication delays 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, and provide its convergence analysis. Since every connected network has a spanning tree, we choose to investigate the distributed algorithm over a tree 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. 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. A node not having any child node is called as a leaf node. Without loss of generality, we assume that only leaf nodes have the distributed data, which are 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 has 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 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ are distributed only to leaf nodes.

For a tree node Q, we can consider a subtree including the tree node Q and its indirect and direct child nodes up to leaf nodes, simply called the subtree Q. Figure 3 illustrates
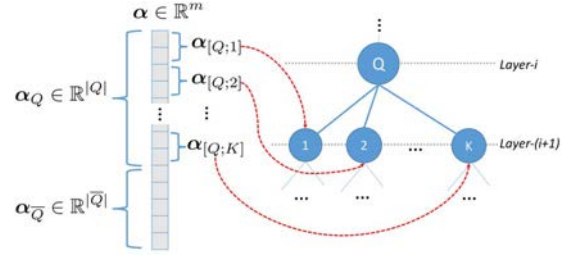
the subtree Q. We also denote the set of indices of all data points stored in the subtree Q as Q, and the set of indices of data points in the k-th direct child node of Q as $[Q;k]$. Therefore, $[Q;k] \subset Q$. Then, $\boldsymbol{\alpha}_{[Q;k]}$ represents the partial vector of $\boldsymbol{\alpha} \in \mathbb{R}^m$ corresponding to the data points in the subtree with the k-th direct child node of Q. Since each node is used for an index set, we denote the number of data points stored in the subtree Q, i.e., the cardinality of Q, as $|Q|$. In a tree network, we also 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 over a general tree-structured network. For simplicity, we consider the tree network in Figure 2, where the number of layers, p, is 2. In a leaf node W_{ij}, TreeDualMethod in Procedure P is run with a local dataset for T_p iterations, and then, the intermediate value $\triangle \boldsymbol{w}$ is shared with its direct parent node, i.e., the sub-central node S_i. In the sub-central node S_i, a global variable $\boldsymbol{w}$ for the i-th cluster is updated, and distributed to local workers W_{ij}'s. After running this process for T_i times independently in clusters, the variables $\triangle \boldsymbol{w}$'s from clusters are shared with the central node. The central node updates and shares the global variable $\boldsymbol{w}$ for whole distributed nodes. And the algorithm repeats this process until some stopping criteria holds. 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 generates $\boldsymbol{\alpha}$ 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, and the dimension of $\boldsymbol{w}$ is much smaller than m, which is normally the case in big data, our distributed algorithm will have less communication burden.

V. CONVERGENCE ANALYSIS OF TREEDUALMETHOD OVER 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 will show a recursive relation between the convergence rate of the algorithm at a tree node Q

Algorithm 2: TreeDualMethod: Distributed Dual Coordinate Ascent for the Root Node Q on the layer-0

Input: $T_0 \geq 1$
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** T_0 **do**
    for *all direct child nodes* $k = 1, 2, ..., K_0$ *in parallel* **do**
        $(\triangle\boldsymbol{\alpha}_{[Q;k]}, \triangle\boldsymbol{w}_k) \leftarrow \text{TreeDualMethod}(\boldsymbol{\alpha}_{[Q;k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$
        $\boldsymbol{\alpha}_{[Q;k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[Q;k]}^{(t-1)} + \frac{1}{K_0} \triangle \boldsymbol{\alpha}_{[Q;k]}$
    end
    $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{1}{K_0} \sum_{k=1}^{K_0} \triangle\boldsymbol{w}_k$
end
Output: $\boldsymbol{\alpha}^{(T_0)}$, and $\boldsymbol{w}^{(T_0)}$

Algorithm 3: TreeDualMethod: Distributed Dual Coordinate Ascent for a General Tree Node Q on the layer-i, $i = 1, 2, ..., p-1$

Input: $T_i \geq 1$, $\boldsymbol{\alpha}_Q$, $\boldsymbol{w}$
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_i **do**
    for *all direct child nodes* $k = 1, 2, ..., K_i$ *of* Q *in parallel* **do**
        $(\triangle\boldsymbol{\alpha}_{[Q;k]}, \triangle\boldsymbol{w}_k) \leftarrow \text{TreeDualMethod}(\boldsymbol{\alpha}_{[Q;k]}^{(t-1)}, \boldsymbol{w}^{(t-1)})$
        $\boldsymbol{\alpha}_{[Q;k]}^{(t)} \leftarrow \boldsymbol{\alpha}_{[Q;k]}^{(t-1)} + \frac{1}{K_i} \triangle \boldsymbol{\alpha}_{[Q;k]}$
    end
    $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \frac{1}{K_i} \sum_{k=1}^{K_i} \triangle\boldsymbol{w}_k$
end
Output: $\triangle\boldsymbol{\alpha}_Q \triangleq \boldsymbol{\alpha}_Q^{(T_i)} - \boldsymbol{\alpha}_Q^{(0)}$, and $\triangle\boldsymbol{w}_Q \triangleq \boldsymbol{A}_Q \triangle \boldsymbol{\alpha}_Q$

---

and that at the node Q's direct child 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.

For clear description, let us consider a general tree network model having $p + 1$ layers from the root node to leaf nodes, where the root node is on the layer-0 and the leaf nodes are on the layer-p. Suppose a node Q on the i-th layer has K direct child nodes on the $(i + 1)$-th layer shown in Figure 3. We use $\boldsymbol{\alpha}_{[Q;k]}$ to denote the partial dual variable vector corresponding to its k-th direct child node, $1 \leq k \leq K$. Then, define the local suboptimality gap for the k-th direct child node of Q as

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

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. We then introduce the following assumption about the local geometric improvement of TreeDualMethod at the k-th direct child node of Q.

Procedure P. TreeDualMethod: Distributed Dual Coordinate Ascent for a Leaf Node Q on the layer-p

Input: $T_p \geq 1$, $\boldsymbol{\alpha}_Q \in \mathbb{R}^{|Q|}$, and $\boldsymbol{w} \in \mathbb{R}^d$ consistent with other coordinate blocks of $\boldsymbol{\alpha}$ s.t. $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{\alpha}$
Data: $\{(\boldsymbol{x}_i, y_i)\}_{i \in Q}$
Initialization: $\triangle\boldsymbol{\alpha}_Q \leftarrow 0 \in \mathbb{R}^{|Q|}$, and $\boldsymbol{w}^{(0)} \leftarrow \boldsymbol{w}$
for $h = 1$ **to** T_p **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 $\triangle\boldsymbol{w}_Q \triangleq \boldsymbol{A}_Q \triangle \boldsymbol{\alpha}_Q$

Assumption 1 (Geometric improvement of TreeDualMethod at a direct child node). *For a tree node Q on the i-th layer, we assume that there exists $\Theta_{i+1} \in [0, 1)$ 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*

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

Note that Assumption 1 here is introduced for an arbitrary tree node in a general tree network and used as a starting assumption in mathematical induction for recursive convergence analysis, while Assumption 1 of [10] 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 [10], and 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 $\boldsymbol{\alpha}_{Q \smallsetminus B}$ in Procedure P.

Proposition 1 ([10, 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*

$$\Theta_p = \left(1 - \frac{\lambda m \gamma}{1 + \lambda m \gamma} \frac{1}{m_B}\right)^{T_p}. \quad (7)$$

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

Basically, the 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), where s in (4) is $\frac{\lambda m \gamma}{1 + \lambda m \gamma}$ in (7).

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 on the i-th layer which has K_i direct child nodes satisfying the local geometric improvement requirement introduced in Assumption 1, with parameters Θ_{i+1}^1, Θ_{i+1}^2, ..., and $\Theta_{i+1}^{K_i}$. We assume that Algorithm 3 (or Algorithm 2) has an input $\boldsymbol{w}$ and is run for T_i iterations. We further assume that loss functions $\ell_i(\cdot)$'s are $1/\gamma$-smooth.*

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

$$\mathbb{E}[D(\boldsymbol{\alpha}_Q^*, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_Q^{(T_i)}, \boldsymbol{\alpha}_{\overline{Q}})] \tag{8}$$

$$\leq \left(1 - (1 - \Theta_{i+1})\frac{1}{K_i}\frac{\lambda m \gamma}{\rho_i + \lambda m \gamma}\right)^{T_i}\left(D(\boldsymbol{\alpha}_Q^*, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_Q^{(0)}, \boldsymbol{\alpha}_{\overline{Q}})\right),$$

where $\Theta_{i+1} = \max_k \Theta_{i+1}^k$, and ρ_i is any real number satisfying

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

Note that the parameter ρ_i is related to the overlapping level among the datasets in the subtree Q. When we have more overlap among local datasets in the subtree, the parameter ρ_i can become larger, which will lead to slower convergence rate.

Proposition 1 is for the local geometric improvement of TreeDualMethod at a leaf node. Namely, Assumption 1 holds for leaf nodes. Theorem 2 is for the local geometric improvement of TreeDualMethod at any non-leaf tree node. Note that $\left(1 - (1 - \Theta_{i+1})\frac{1}{K_i}\frac{\lambda m \gamma}{\rho_i + \lambda m \gamma}\right)^{T_i}$ in (8) becomes Θ_i for a tree node Q on the i-th layer, and then, (8) is interpreted as the local geometric improvement of TreeDualMethod at the direct child node by the direct parent node of Q, which is a node on the $(i-1)$-th layer. Basically, for the convergence rate of the generalized dual coordinate ascent over the whole tree network, we use the mathematical induction, where Proposition 1 is the base case, Assumption 1 is the starting assumption of the mathematical induction, and Theorem 2 completes the induction for the recursive convergence analysis. 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 with the fact that Assumption 1 holds true for every node in a tree network. Figure 4 illustrates the structure of the tree network factor in convergence rate, shown through Θ_1 and Θ_2.

We remark that Theorem 2 is different from Theorem 2 of [10] in three aspects. Firstly, Theorem 2 is applicable to any tree node in a general tree network, beyond a star network discussed in [10]. 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 [10], 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 [10] 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 [10], 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 [10], we do not consider the different local-dual problem introduced in Eqn. (8) of [10] 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.
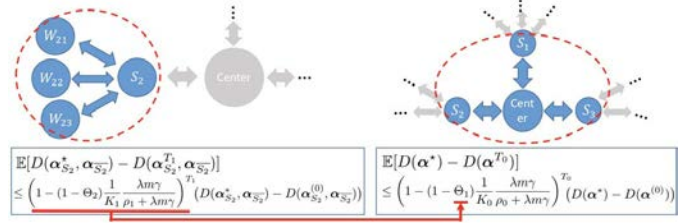

Fig. 4. Illustration of the structure of the tree network factor in convergence analysis.

By denoting the convergence bound in (8) as Θ_i, i.e.,

$$\Theta_i = \left(1 - (1 - \Theta_{i+1})C_i/K_i\right)^{T_i}, \tag{9}$$

where C_i represents $\frac{\lambda m \gamma}{\rho_i + \lambda m \gamma}$, T_i is the outer iteration in a tree node on the i-th layer, and K_i is the number of direct child nodes attached to a tree node on the i-th layer, we can express the convergence bound on the whole tree-network, i.e., Θ_0, in terms of the number of layers p, and the number of nodes K_i's and C_i's, as follows:

$$\Theta_0 = \left(1 - (1 - \Theta_1)\frac{C_0}{K_0}\right)^{T_0} \tag{10}$$

$$= \left(1 - \left(1 - \left(1 - \cdots\left(1 - \Theta_p\right)\frac{C_{p-1}}{K_{p-1}}\right)^{T_{p-1}}\cdots\frac{C_1}{K_1}\right)^{T_1}\right)\frac{C_0}{K_0}\right)^{T_0},$$

where Θ_p is introduced in (7). For simplicity, we assume that all tree nodes on the i-th layer have the same number of direct child nodes K_i.

If $\left|T_i \cdot \frac{C_i \Theta_{i+1}}{K_i - C_i}\right| \ll 1$, for $i = 0, 1, ..., p-1$, by applying the binomial approximation, we can have

$$\Theta_i = \left(\frac{K_i - C_i}{K_i}\right)^{T_i}\left(1 + \frac{C_i}{K_i - C_i}\Theta_{i+1}\right)^{T_i}$$

$$\approx \left(\frac{K_i - C_i}{K_i}\right)^{T_i}\left(1 + \frac{C_i}{K_i - C_i}\Theta_{i+1} \cdot T_i.\right)$$

and approximate (10) as follows:

$$\Theta_0 \approx \left(\frac{K_0 - C_0}{K_0}\right)^{T_0} + \sum_{r=1}^{p-1}\prod_{i=0}^{r}\left(\frac{K_i - C_i}{K_i}\right)^{T_i} \cdot \prod_{j=0}^{r-1}\frac{C_j T_j}{K_j - C_j}$$

$$+ \prod_{i=0}^{p-1}\left(\frac{K_i - C_i}{K_i}\right)^{T_i}\left(\frac{C_i T_i}{K_i - C_i}\right)\Theta_p. \tag{11}$$

In Section VII, we will investigate the gap between (10) and (11) through numerical experiments as well as analyze the network topology's effect including the number of workers K_i and the number of layers p on the convergence bounds over the whole tree network introduced in (10) and (11).

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 layers, the number of nodes, and 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 will 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 will optimize the number of local iterations T_p in Procedure P and T_i in Algorithm 3 for maximum convergence speed.

VI. COMMUNICATION IMPACTS ON THE CONVERGENCE RATE OF DISTRIBUTED DUAL COORDINATE ASCENT

Earlier works [9–11] 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 dual coordinate ascent. The research [18, 29–31] 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 that we deal with in this paper, to the best of our knowledge, our paper is the first one to analytically study the communication delay's impact on the convergence rate, and finds the optimal number of local iterations depending on the communication delay severity.

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 denote the round-trip communication delay between a local worker and the central station as 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.

Suppose that each local worker performs T_p local iterations before communicating with the central station, and there are T_0 outer iterations in total. Then, the total experienced time is $t_{total} = (t_{lp}T_p + t_{delay} + t_{cp}) \cdot T_0$. Hence, the number of outer iterations T_0 is given by

$$T_0 = t_{total} / (t_{lp}T_p + t_{delay} + t_{cp}). \tag{12}$$

From (8), for T_0 outer iterations, the expected gap between the optimal objective value and the current objective value with Algorithm 1 is expressed as

$$\left(1 - \left(1 - [1-\delta]^{T_p}\right)\frac{C}{K}\right)^{T_0}, \tag{13}$$

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 (13) for a given total time t_{total}, we introduce the following

optimization problem over the number of local iterations T_p by plugging (12) into (13):

$$\underset{T_p \geq 0}{\text{minimize }} F(T_p) \triangleq \left(1 - \left(1 - [1-\delta]^{T_p}\right)\frac{C}{K}\right)^{\frac{t_{total}}{t_{lp}T_p + t_{delay} + t_{cp}}}. \tag{14}$$

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

$$\ln F(T_p) = \underbrace{\frac{t_{total}/t_{lp}}{T_p + (t_{delay} + t_{cp})/t_{lp}}}_{(A)} \underbrace{\ln\left[\frac{K-C}{K} + \frac{C}{K}[1-\delta]^{T_p}\right]}_{(B)}. \tag{15}$$

(15) 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 T_p. And for the logarithm part (B), as T_p increases, (B) goes to $\ln((K-C)/K)$, which is less than zero, due to the condition $0 \leq 1-\delta < 1$. At $T_p = 0$, $\ln F(T_p)$ is 0 due to $(B) = 0$. As T_p goes to infinity, $\ln F(T_p)$ will go to 0 due to $(A) = 0$. Therefore, we can expect at least a critical point at some T_p. In order to figure out the critical point of (15), which is the same critical point of $F(T_p)$, we calculate the first order condition as follows:

$$\frac{d \ln F(T_p)}{dT_p} = \frac{(\frac{K-C}{K})(\frac{t_{total}}{t_{lp}})(1-\delta)^{T_p}\ln(1-\delta)}{(\frac{K-C}{K} + \frac{C}{K}[1-\delta]^{T_p})(T_p + \frac{t_{delay}+t_{cp}}{t_{lp}})}$$
$$- \frac{(\frac{t_{total}}{t_{lp}})\ln(\frac{K-C}{K} + \frac{C}{K}[1-\delta]^{T_p})}{(T_p + \frac{t_{delay}+t_{cp}}{t_{lp}})^2} = 0. \tag{16}$$

By simplifying (16) and denoting $\frac{t_{delay}+t_{cp}}{t_{lp}}$ to r, we have the first order condition over T_p as

$$\underbrace{\frac{K-C}{K}(T_p + r)[1-\delta]^{T_p}\ln(1-\delta)}_{(C)} \tag{17}$$
$$\underbrace{-\left(\frac{K-C}{K} + \frac{C}{K}[1-\delta]^{T_p}\right)\ln\left(\frac{K-C}{K} + \frac{C}{K}[1-\delta]^{T_p}\right)}_{(D)} = 0.$$

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

$$\frac{K-C}{K}(T_p + r)[1-\delta]^{T_p}\ln(1-\delta) = \frac{K-C}{K}\ln\left(\frac{K-C}{K}\right). \tag{18}$$

Note that (18) has Lambert W-function [32], which is defined as when $xe^x = a$, the solution x is $W(a)$, where $W(\cdot)$ is the Lambert W-function. By the definition of the Lamber W-function, we have the following optimal local iteration T_p:

$$T_p = \frac{1}{\ln(1-\delta)}W\left([1-\delta]^r \ln\left(\frac{K-C}{K}\right)\right) - r. \tag{19}$$

From the recursive manner of the convergence analysis in a tree network as introduced in Section V, the optimal number of iterations T_i in Algorithm 3 for a node Q can also be obtained by using aforementioned equation (14) with slightly different interpretation. In the tree network, the number of local iterations T_p in (14) is understood as the number of local iteration T_i 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 (14) with different interpretation, the optimal number of local iterations for a general tree node Q can be obtained as (19).

Since the objective function $F(T_p)$ in (14) represents the convergence bound in terms of time, it is clearly recognized that for a fixed local iteration T_p, the larger communication severity $r = t_{delay}/t_{lp}$ exists, the slower convergence rate we have. Additionally, if in a network, a central node, sub-central nodes and local workers are needed to be chosen, by considering the convergence analysis shown in a recursive manner and the communication delay between layers, choosing a root node making the depth of the connected network shallow will be better for fast convergence. In the next section, we will further investigate the impact of the communication delay severity r, and other parameters including C, K, and δ in (19) on the optimal number of local iterations T_p.

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] [33], and wine quality dataset[3] [34]. 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. And, we further numerically investigate the effect of network topology on the convergence of the distributed dual coordinate ascent over a tree network.

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 [10], [11] compared the distributed dual coordinate ascent in a star network, so-called CoCoA, with other well known methods including mini-batch SDCA [35], local SGD and mini-batch-SGD [36], we focus on comparing our generalized distributed dual coordinate ascent in tree networks with that in star networks

---

[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

---

by considering network constraints, especially, communication delay and latency. Additionally, since we are interested in the communication network's effect on the convergence speed of the synchronous distributed dual coordinate ascent, considering the CoCoA+ [11], which is the updated version of CoCoA, or an asynchronous method, is out of the 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 95412 instances. We consider the following specific optimization problem by setting $\ell_i(\boldsymbol{w}^T \boldsymbol{x}_i) = (\frac{1}{\lambda m} \boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$:

$$\operatorname*{minimize}_{\boldsymbol{w} \in \mathbb{R}^d} \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\|\boldsymbol{A}^T\boldsymbol{w} - \boldsymbol{y}\|^2, \tag{20}$$

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 (20):

$$\operatorname*{maximize}_{\boldsymbol{\alpha} \in \mathbb{R}^m} -\frac{\lambda}{2}\|\boldsymbol{A}\boldsymbol{\alpha}\|^2 - \lambda^2 m \sum_{i=1}^m \left(\frac{\alpha_i^2}{4} - \frac{y_i \alpha_i}{\lambda m}\right). \tag{21}$$

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

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

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 95410 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, 9541 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 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
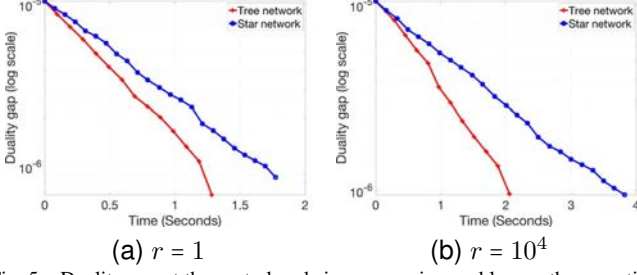
Fig. 5. Duality gap at the central node in a regression problem as the operation time of the algorithms goes. The distributed dual coordinate ascent in a tree network (red) and a star network (blue), i.e., CoCoA, are considered when the communication delay, t_{delay}, exists between the central node 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.
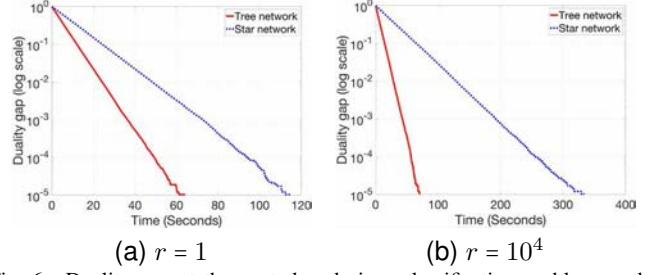

Fig. 6. Duality gap at the central node in a classification problem as the operation time of the algorithms goes. The distributed dual coordinate ascent in a tree network (red solid line) and a star network (blue dotted line), i.e., CoCoA, are considered when the communication delay, t_{delay}, exists 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.

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 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 [37], which is a binary classification dataset having 581012 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 $\|\boldsymbol{x}_i\| \le 1$, we normalize the dataset and $y_i \in \{-1, 1\}$, $i = 1, ..., m$. 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 with 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:

$$\underset{\boldsymbol{w} \in \mathbb{R}^d}{\text{minimize}} \ \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\max(\boldsymbol{0}, \boldsymbol{1} - \boldsymbol{A}^T\boldsymbol{w}), \quad (22)$$

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 (22) is stated as follows:

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \ \sum_{i=1}^{m}\alpha_i - \frac{1}{2\lambda}\|\boldsymbol{A}\boldsymbol{\alpha}\|^2 \quad \text{subject to} \ 0 \le \alpha_i \le \frac{1}{m}, \ \forall i. \ (23)$$

Note here that while deriving the dual problem (23), 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:

$$\underset{\boldsymbol{\alpha}_Q \in \mathbb{R}^{|Q|}}{\text{maximize}} \ -\frac{\lambda}{2}\|\overline{\boldsymbol{w}} + \frac{1}{\lambda}\boldsymbol{A}_Q\boldsymbol{\alpha}_Q\|^2 + \sum_{i \in Q}\alpha_i + \sum_{i \in \overline{Q}}\alpha_i$$

$$\text{subject to} \ 0 \le \alpha_i \le \frac{1}{m}, \ \forall i \in Q, \quad (24)$$

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:

$$\triangle\alpha = \underset{\triangle\alpha}{\text{argmax}} -\frac{\lambda}{2}\|\boldsymbol{w}^{(h-1)} + \frac{1}{\lambda^2 m}\triangle\alpha y_i\boldsymbol{x}_i\|^2 + (\alpha_i^{(h-1)} + \triangle\alpha)$$

$$\text{subject to} \ 0 \le \alpha_i^{(h-1)} + \triangle\alpha \le \frac{1}{m}. \quad (25)$$

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 (25), we calculate the optimal solution of (25) without the box constraint, i.e., $0 \le \alpha_i^{(h-1)} + \triangle\alpha \le \frac{1}{m}$, and then project the optimal solution onto the box constraint as follows:

$$\triangle\alpha = \begin{cases} 1/m - \alpha_i^{(h-1)} & \text{if } \alpha_i^{(h-1)} + \triangle\alpha > 1/m \\ -\alpha_i^{(h-1)} & \text{if } \alpha_i^{(h-1)} + \triangle\alpha < 0 \end{cases}. \quad (26)$$

Figure 6 shows the duality gap as the operation time of the algorithms goes. As 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 T_p, we provide Figure 7 to show the optimal number of local iterations T_p by finding the critical point of (16). 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
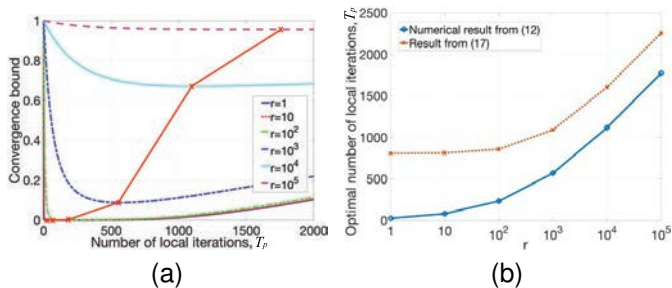
Fig. 7. (a) The objective value of (14), which is the convergence bound (or improvement), when the number of iterations T_p is varied from 1 to 2000, where $(C, K, \delta, t_{total}, t_{lp}, t_{cp}) = (0.5, 3, 1/300, 1, 4 \times 10^{-5}, 3 \times 10^{-5})$ and $t_{delay} = r \times t_{lp}$. The red line 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.

objective values of (14) when T_p 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 (16) 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 (19) with given aforementioned parameters, while the blue solid line is obtained by numerically calculating (14) and finding the optimal T_p 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. It is noteworthy that in Figure 7(b), the difference between the numerical results from (14) and the analytical solution in (19) is observed. Especially, there is a big gap in the small communication delay severity, e.g., $r = 1$. This gap occurs because in the derivation of the analytical solution in (19), we approximate (D) of (17) by assuming that the local iteration T_p is large enough. Hence, the gap becomes smaller when the communication delay severity is increased.

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 [34] in a star network. For the number of iterations in local workers, we vary T_p 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 $T_p = 2000$, while when $r = 10^5$, the fastest convergence is obtained at $T_p = 10000$. 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 (19) to see whether the analytical solution for the optimal number of local iterations fits to the simulation results. We set parameters to $\delta = 1/1000$, $K = 4$, and $C = 0.9$ by reflecting the network and simulation settings. With those
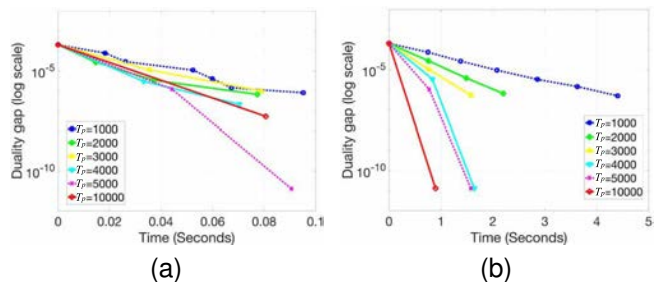

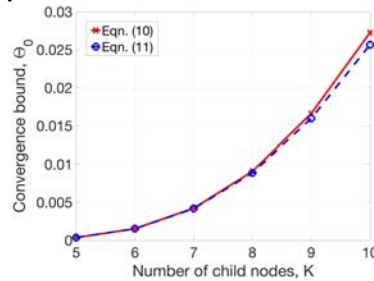Fig. 8. (a) Duality gap when the delay severity r is 1. (b) Duality gap when the delay severity r is 10^5.

Fig. 9. Convergence bound over the whole tree network, Θ_0, by varying number of child nodes K with fixed other parameters $(p, T_i, C_i) = (3, 40, 0.9)$ for all $i = 0, 1, ..., p - 1$, and $\Theta_p = 0.5$.

parameter values, we obtain 2117 for $r = 1$ and 6028 for $r = 10^5$ from the analytical solution in (19), while in the simulation, $T_p = 2000$ for $r = 1$ and $T_p = 10000$ 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, (19) can still be used as a guideline for the number of local iterations in local workers.

C. Network topology's effect on convergence bound

In this subsection, we numerically investigate the effect of network topology on the convergence bound over the whole tree network, i.e., Θ_0 introduced in (10). In order to see the effect of the number of nodes on the convergence bound, we firstly run simulations by varying the number of child nodes, K_i. For the simulation, we take into account a tree network having three layers, i.e., $p = 3$. For other parameters, we set (T_i, C_i) to $(40, 0.9)$, for all $i = 0, 1, ..., p$, and Θ_p to 0.5. We consider that all nodes have the same number of child nodes K, i.e., $K_i = K$, for all $i = 0, 1, ..., p - 1$, and vary K from 5 to 10. Figure 9 shows the convergence bound Θ_0 by varying the number of child node K. The red solid line and the blue dotted line represent the convergence bound expressed in (10), and its approximation introduced in (11). As the number of nodes is increased, the convergence bound Θ_0 is also increased.

We further run simulations to investigate the effect of the number of layers, p, on the convergence bound, Θ_0. For simulations, we set $\frac{K_i - C_i}{K_i} = \frac{K - C}{K}$ and $T_i = T$, for all i. From the setting, we can further simplify the approximated convergence bound introduced in (11) as

$$\Theta_0 = \left(\frac{K - C}{K}\right)^T + \sum_{r=1}^{p-1} \left(\frac{K - C}{K}\right)^{T(r+1)} \left(\frac{CT}{K - C}\right)^r$$
$$+ \left[\left(\frac{K - C}{K}\right)^T \left(\frac{CT}{K - C}\right)\right]^p \Theta_p. \quad (27)$$

For given K and C, if T is large enough to be $\left(\frac{K-C}{K}\right)^T \frac{CT}{K-C} < 1$, then, the convergence bound is expressed as
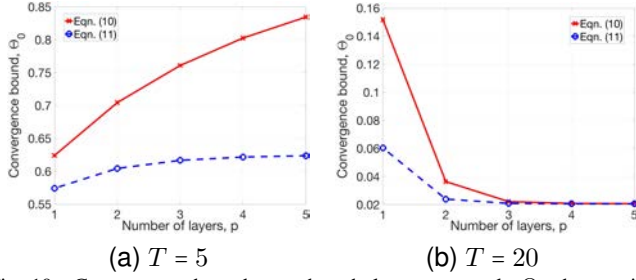
(a) $T = 5$ (b) $T = 20$

Fig. 10. Convergence bound over the whole tree network, Θ_0, by varying the number of layers p with fixed other parameters $(K, C_i) = (5, 0.9)$ for all $i = 0, 1, ..., p-1$, and $\Theta_p = 0.5$.

(a) r varied   (b) C varied
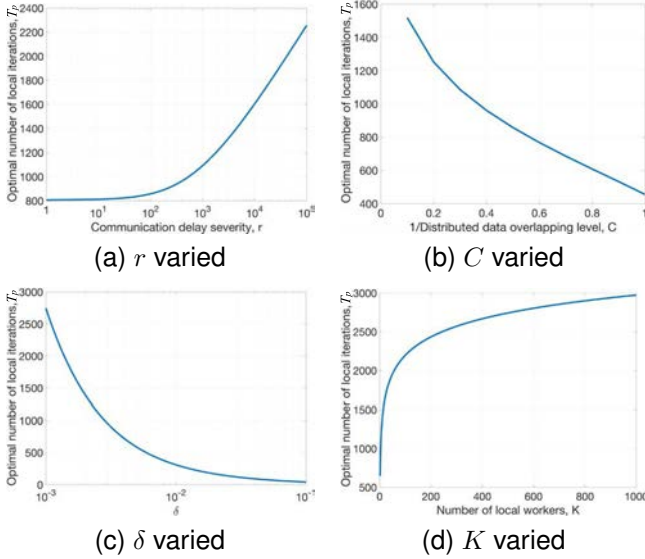

(c) δ varied   (d) K varied

Fig. 11. Optimal number of local iterations, T_p, by varying parameters. Except for the varying parameter, other parameters are fixed to $(C, K, \delta, r) = (0.5, 3, 1/300, 100)$.

$$\Theta_0 = \left(\frac{K-C}{K}\right)^T + \left(\frac{K-C}{K}\right)^T \cdot \frac{\left(\frac{K-C}{K}\right)^T \frac{CT}{K-C} - \left(\left(\frac{K-C}{K}\right)^T \frac{CT}{K-C}\right)^p}{1 - \left(\frac{K-C}{K}\right)^T \frac{CT}{K-C}}$$
$$+ \left(\frac{K-C}{K}\right)^{T \cdot p} \left(\frac{CT}{K-C}\right)^p \Theta_p. \tag{28}$$

Note that $\lim_{T \to \infty} \left(\frac{K-C}{K}\right)^T \frac{CT}{K-C} = 0$. If T is small enough to be $\left(\frac{K-C}{K}\right)^T \frac{CT}{K-C} > 1$, then, we have

$$\Theta_0 = \left(\frac{K-C}{K}\right)^T + \left(\frac{K-C}{K}\right)^T \cdot \frac{\left(\left(\frac{K-C}{K}\right)^T \frac{CT}{K-C}\right)^p - \left(\frac{K-C}{K}\right)^T \frac{CT}{K-C}}{\left(\frac{K-C}{K}\right)^T \frac{CT}{K-C} - 1}$$
$$+ \left(\frac{K-C}{K}\right)^{T \cdot p} \left(\frac{CT}{K-C}\right)^p \Theta_p, \tag{29}$$

Therefore, for given K and C, depending on the number of iteration T, the dominant term in (10) (or (11)) is changed like stated in (28) or (29), and the convergence bound, Θ_0, follows two different trends as shown in Figure 10. For Figures 10(a) and 10(b), we set the number of iterations T to 5 and 20 respectively with maintaining the other parameters the same. Note that when $p = 1$, it represents the star network, and when the number of iteration T is large enough, we can have the better convergence bound in a tree network as in Figure 11(b).

D. Parameter setting for faster convergence speed

In order to investigate the optimal number of local iterations which achieves the fastest convergence speed, from (19), we generate Figure 11 by varying each parameter r, C, K, and δ.

In Figure 11(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 11(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 parameter ρ, which is reciprocal of the parameter C in (19), 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 T_p, we vary C with fixing other parameters to $(K, \delta, r) = (3, 1/300, 100)$, and draw the graph in Figure 11(b). From Figure 11(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. From Figure 11(d), as the number of local workers, K, increases, the optimal number of local iterations, T_p, is also increased.

REFERENCES

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014. 1

[2] 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. 1

[3] 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. 1

[4] 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. 1

[5] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019. 1

[6] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19–25, 2020. 1

[7] 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. 1

[8] 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. 1

[9] T. Yang, "Trading computation for communication: Distributed stochastic dual coordinate ascent," in *Advances in Neural Information Processing Systems*, 2013, pp. 629–637. 1, 2, 3, 7

[10] 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. 1, 2, 3, 5, 6, 7, 8, 12

[11] 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. 1, 2, 3, 7, 8

[12] 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. 1

[13] 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. 1

[14] 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. 1, 2, 3, 14

[15] R. Zhang, S. Zheng, and J. T. Kwok, "Fast distributed asynchronous SGD with variance reduction," *CoRR, abs/1508.01633*, 2015. 1

[16] Z. Huo and H. Huang, "Distributed asynchronous dual free stochastic dual coordinate ascent," in *Proceedings of the IEEE International Conference on Data Mining*, 2018. 1, 2, 3

[17] 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. 1, 2

[18] 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. 2, 7

[19] 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. 2

[20] 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. 2

[21] 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. 2

[22] 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. 2

[23] 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. 2

[24] M. Carreira-Perpinan and W. Wang, "Distributed optimization of deeply nested systems," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2014, pp. 10–19. 2

[25] J. Zeng, T. T.-K. Lau, S. Lin, and Y. Yao, "Global convergence of block coordinate descent in deep learning," in *Proceedings of International Conference on Machine Learning*, 2019, pp. 7313–7323. 2

[26] T. T.-K. Lau, J. Zeng, B. Wu, and Y. Yao, "A proximal block coordinate descent algorithm for deep neural network training," *arXiv preprint arXiv:1803.09082*, 2018. 2

[27] 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. 2

[28] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004. 3

[29] 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. 7

[30] 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. 7

[31] 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. 7

[32] 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. 7

[33] 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. 8

[34] 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. 8, 10

[35] 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. 8

[36] 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. 8

[37] 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. 9

APPENDIX A
PROOF OF THEOREM 2

We follow the proof of Theorem 2 of [10] 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.

$$\boldsymbol{\alpha}^{(t+1)} = (\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}) = (\boldsymbol{\alpha}_{[1:K]}^{(t)} + \frac{1}{K}\sum_{k=1}^{K} \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}), \quad (30)$$

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

$$D(\boldsymbol{\alpha}_Q, \boldsymbol{\alpha}_{\overline{Q}}) \tag{31}$$
$$= -\frac{\lambda}{2}\|\boldsymbol{A}_Q\boldsymbol{\alpha}_Q + \boldsymbol{A}_{\overline{Q}}\boldsymbol{\alpha}_{\overline{Q}}\|^2 - \frac{1}{m}\sum_{i\in Q}\ell_i^*(-\alpha_i) - \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^*(-\alpha_i)$$
$$= -\frac{\lambda}{2}\|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \frac{1}{m}\sum_{i\in[1:K]}\ell_i^*(-\alpha_i) - \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^*(-\alpha_i),$$

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 (30), we have

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

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

$$D(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})$$
$$\geq \frac{1}{K}\sum_{k=1}^{K}\left[D(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\right]$$
$$= \frac{1}{K}\sum_{k=1}^{K}\left[D(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})\right.$$
$$- D((\boldsymbol{\alpha}_{[Q;1]}^{(t)}, ..., \boldsymbol{\alpha}_{[Q;k]}^{\star}, ..., \boldsymbol{\alpha}_{[Q;K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}))$$
$$+ D((\boldsymbol{\alpha}_{[Q;1]}^{(t)}, ..., \boldsymbol{\alpha}_{[Q;k]}^{\star}, ..., \boldsymbol{\alpha}_{[Q;K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\left.\right]$$
$$= \frac{1}{K}\sum_{k=1}^{K}\left[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}) - \epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})\right],$$

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

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

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.

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

$$
= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})
$$
$$
+ \frac{\lambda}{2} \Big(\|\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 - \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]}^{(t)} + \overline{\boldsymbol{w}}\|^2 \Big)
$$
$$
- \frac{\lambda}{2} \sum_{k=1}^{K} \Big[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}_{[k]}^{(t)} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2 \Big] + \lambda \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]}^{(t)} + \overline{\boldsymbol{w}}\|^2
$$
$$
- \lambda(\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]}^{(t)} + \overline{\boldsymbol{w}})^T(\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}})
$$
$$
= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})
$$
$$
- \frac{\lambda}{2} \sum_{k=1}^{K} \Big[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}_{[k]}^{(t)} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2 \Big]
$$
$$
+ \frac{\lambda}{2} \Big(\|\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}\|^2 + \|\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]}^{(t)} + \overline{\boldsymbol{w}}\|^2
$$
$$
- 2(\boldsymbol{A}_{[1:K]}\boldsymbol{\alpha}_{[1:K]}^{(t)} + \overline{\boldsymbol{w}})^T(\boldsymbol{A}_{[1:K]}\hat{\boldsymbol{\alpha}}_{[1:K]} + \overline{\boldsymbol{w}}) \Big)
$$
$$
= \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}) \tag{32}
$$
$$
- \frac{\lambda}{2} \underbrace{\Big[\sum_{k=1}^{K} \Big[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}_{[k]}^{(t)} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2 \Big] - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)})\|^2 \Big]}_{=(A)}
$$

We can lower-bound (32) by upper-bounding (A). For the upper-bound of (A), we have

$$
(A) = \sum_{k=1}^{K} \Big[\|\boldsymbol{A}_{[k]}(\boldsymbol{\alpha}_{[k]}^{(t)} - \hat{\boldsymbol{\alpha}}_{[k]})\|^2 \Big] - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)})\|^2
$$
$$
\leq \sum_{i \in [1:K]} \|\boldsymbol{A}_i(\alpha_i^{(t)} - \hat{\alpha}_i)\|^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)})\|^2
$$
$$
\leq \sum_{i \in [1:K]} \frac{1}{\lambda^2 m^2} \|\boldsymbol{x}_i\|^2 (\alpha_i^{(t)} - \hat{\alpha}_i)^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)})\|^2
$$
$$
\leq \frac{1}{\lambda^2 m^2} \sum_{i \in [1:K]} (\alpha_i^{(t)} - \hat{\alpha}_i)^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)})\|^2
$$
$$
\leq \frac{1}{\lambda^2 m^2} \|\boldsymbol{\alpha}_{[1:K]}^{(t)} - \hat{\boldsymbol{\alpha}}_{[1:K]}\|^2 - \|\boldsymbol{A}_{[1:K]}(\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)})\|^2
$$
$$
\leq \frac{\rho}{\lambda^2 m^2} \|\boldsymbol{\alpha}_{[1:K]}^{(t)} - \hat{\boldsymbol{\alpha}}_{[1:K]}\|^2,
$$

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., $\|\boldsymbol{x}_i\| \leq 1$. We can have the last inequality by introducing ρ_{min}, which is the minimum value of ρ, to hold the last inequality as follows:

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

The condition $\rho_{min} \geq 0$ 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, (32), which is $\sum_{k=1}^{K} \epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})$, is lower-bounded as follows:

$$
\sum_{k=1}^{K} \epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})
$$
$$
\geq \underset{\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|[1:K]|}}{\text{maximize}} D(\hat{\boldsymbol{\alpha}}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})
$$
$$
- \frac{\rho}{2\lambda m^2} \|\hat{\boldsymbol{\alpha}}_{[1:K]} - \boldsymbol{\alpha}_{[1:K]}^{(t)}\|^2
$$

$$\geq \underset{\eta\in[0,1]}{\text{maximize}}\; D\big(\eta\boldsymbol{\alpha}^\star_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)$$
$$- \frac{\rho}{2\lambda m^2}\big\|\eta\boldsymbol{\alpha}^\star_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\big\|^2$$
$$\geq \underset{\eta\in[0,1]}{\text{maximize}}\; \eta D\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + (1-\eta) D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)$$
$$- D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + \frac{\gamma\eta(1-\eta)}{2m}\big\|\boldsymbol{\alpha}^\star_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\big\|^2$$
$$- \frac{\rho\eta^2}{2\lambda m^2}\big\|\boldsymbol{\alpha}^\star_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\big\|^2$$
$$\geq \underset{\eta\in[0,1]}{\text{maximize}}\; \eta D\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - \eta D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)$$
$$+ \frac{\gamma\eta(1-\eta)}{2m}\big\|\boldsymbol{\alpha}^\star_{[1:K]}$$
$$- \boldsymbol{\alpha}^{(t)}_{[1:K]}\big\|^2 - \frac{\rho\eta^2}{2\lambda m^2}\big\|\boldsymbol{\alpha}^\star_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\big\|^2$$
$$= \underset{\eta\in[0,1]}{\text{maximize}}\; \eta D\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - \eta D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)$$
$$+ \frac{\eta}{2m}\left(\gamma - \frac{\lambda m\gamma + \rho}{\lambda m}\eta\right)\big\|\boldsymbol{\alpha}^\star_{[1:K]} - \boldsymbol{\alpha}^{(t)}_{[1:K]}\big\|^2, \tag{33}$$

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 ℓ_i^* is γ strongly convex: for all $u, v \in \mathbb{R}$ and $\eta \in [0,1]$ [14]:

$$- \ell_i^*(\eta u + (1-\eta)v)$$
$$\geq -\eta\ell_i^*(u) - (1-\eta)\ell_i^*(v) + \frac{\gamma\eta(1-\eta)}{2}(u-v)^2. \tag{34}$$

From (34), for $D\big(\eta\boldsymbol{\alpha}^\star_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)$, we have

$$D\left(\eta\boldsymbol{\alpha}^\star_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\right)$$
$$= -\frac{1}{2}\left\|\boldsymbol{A}\big(\eta\boldsymbol{\alpha}^\star_{[1:K]} + (1-\eta)\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)\right\|^2$$
$$- \frac{1}{m}\sum_{i\in[1:K]}\ell_i^*\big(-\eta\alpha_i^\star - (1-\eta)\alpha_i^{(t)}\big)$$
$$- \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^*\big(-\eta\alpha_i - (1-\eta)\alpha_i\big)$$
$$= -\frac{1}{2}\left\|\eta\boldsymbol{A}\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + (1-\eta)\boldsymbol{A}\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)\right\|^2$$
$$- \frac{1}{m}\sum_{i\in[1:K]}\ell_i^*\big(-\eta\alpha_i^\star - (1-\eta)\alpha_i^{(t)}\big)$$
$$- \frac{1}{m}\sum_{i\in\overline{Q}}\ell_i^*\big(-\eta\alpha_i - (1-\eta)\alpha_i\big)$$
$$\overset{(34)}{\geq} -\frac{1}{2}\left\|\eta\boldsymbol{A}\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) + (1-\eta)\boldsymbol{A}\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)\right\|^2$$
$$- \frac{1}{m}\sum_{i\in[1:K]}\left[\eta\ell_i^*(-\alpha_i^\star) + (1-\eta)\ell_i^*(-\alpha_i^{(t)})\right.$$
$$\left. - \frac{\gamma\eta(1-\eta)}{2}(\alpha_i^\star - \alpha_i^{(t)})^2\right]$$
$$- \frac{1}{m}\sum_{i\in\overline{Q}}\left[\eta\ell_i^*(-\alpha_i) + (1-\eta)\ell_i^*(-\alpha_i)\right]$$

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

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.

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

$$(33) \geq \frac{\lambda m\gamma}{\lambda m\gamma + \rho}\left(D\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)\right)$$

Therefore, we have

$$\mathbb{E}[D(\boldsymbol{\alpha}^{(t+1)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}) \mid \overline{\boldsymbol{w}}, \boldsymbol{\alpha}^{(t)}_{[1:K]}]$$
$$\geq \frac{1}{K}(1-\Theta)\sum_{k=1}^{K}\epsilon_{Q,k}(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}})$$
$$\geq \frac{1}{K}(1-\Theta)\frac{\lambda m\gamma}{\lambda m\gamma + \rho}\left(D\big(\boldsymbol{\alpha}^\star_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big) - D\big(\boldsymbol{\alpha}^{(t)}_{[1:K]}, \boldsymbol{\alpha}_{\overline{Q}}\big)\right) \tag{35}$$

From (35), we have

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

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

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

$\square$