2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP) | 979-8-3503-2411-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/MLSP55844.2023.10285903

DISTRIBUTED DUAL COORDINATE ASCENT WITH IMBALANCED DATA ON A GENERAL TREE NETWORK

Myung Cho[⋆§] *Lifeng Lai*[†] *Weiyu Xu*[‡]

[⋆] Department of Electrical and Computer Engineering, California State University, Northridge, CA, USA
[§] Department of Electrical and Computer Engineering, Penn State Behrend, Erie, PA, USA
[†] Department of Electrical and Computer Engineering, University of California, Davis, CA, USA
[‡] Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA, USA

ABSTRACT

In this paper, we investigate the impact of imbalanced data on the convergence of distributed dual coordinate ascent in a tree network for solving an empirical loss minimization problem in distributed machine learning. To address this issue, we propose a method called delayed generalized distributed dual coordinate ascent that takes into account the information of the imbalanced data, and provide the analysis of the proposed algorithm. Numerical experiments confirm the effectiveness of our proposed method in improving the convergence speed of distributed dual coordinate ascent in a tree network.

***Index Terms*—** distributed machine learning, federated learning, tree network, network topology, imbalanced data

1. INTRODUCTION

In the field of Machine Learning (ML) and Artificial Intelligence (AI), the use of large amounts of data, known as *big data*, plays a crucial role in the performance of ML and AI techniques. Due to the exceptional performance of ML/AI with big data, these techniques are becoming increasingly popular and are being applied to many different applications.

However, in practice, processing ML/AI operations with big data poses many challenges such as limited hardware resources including storage space and processing power, and privacy and security issues. Specifically, due to limited storage space, big data is often stored in a distributed manner over a network, raising questions about how to deal with these distributed data when processing ML/AI operations. Furthermore, even though there are distributed algorithms that can process distributed data for ML/AI operations, communication to share intermediate results such as learning parameters can be a significant challenge due to constraints such as limited communication bandwidth, delay, and power. Thus, it is important to design efficient algorithms for distributed ML/AI processes that take into account these communication network constraints when dealing with distributed data.

To address the challenge of handling distributed data on a network, many research were conducted to develop efficient distributed ML/AI algorithms. Researchers in [1–5] studied synchronous Stochastic Gradient Decent (SGD). Synchronous Stochastic Dual Coordinate Ascent (SDCA) and its variations were investigated in [6–13]. Asynchronous SGD was studied in [14–16]. Asynchronous SDCA was investigated for handling distributed data on a network in [17–20].

However, most of the research in this area has focused on designing distributed algorithms on a star network, which is a very simple network topology. It's worth noting that data are not always stored on a star network. In reality, networks can have various topologies such as star, tree, ring, bus, or mesh, etc. Additionally, in a network, some nodes may not directly communicate with a central node. In this case, a line of nodes may be considered as a virtual node directly connected to a central node to apply distributed algorithms for a star network. However, the system can easily suffer from significant communication delays caused by passing intermediate results through the line of nodes to a central node. Thus, it is important to design efficient distributed ML/AI algorithms by considering different network topologies. To address this problem, the researchers in [12, 13] have designed a distributed dual coordinate ascent for distributed ML process on a general tree network, and provided the convergence analysis of the algorithm on a general tree network under the assumption that the dataset is evenly distributed on a tree network. Since every connected network has its spanning tree, the distributed algorithm can be applied to various network topologies as long as there are no isolated nodes in the network.

In practice, however, due to different circumstances and situations in data acquisition such as different frequency of data acquisition in nodes, different sensitivity of sensors, and noise or bias during data acquisition, we can have imbalanced data on a network in distributed ML process. In this paper, we investigate the effect of imbalanced data on distributed dual

Lifeng Lai's research is supported by National Science Foundation (NSF) under grant ECCS-2000415.

Weiyu Xu's research is supported by NSF under grant ECCS-2000425 and ECCS-2133205.

979-8-3503-2411-2/23/$31.00 ©2023 IEEE

coordinate ascent in a general tree network, and propose our method called a generalized distributed dual coordinate ascent on a general tree network to mitigate the effect of imbalanced data on processing distributed data on a tree network.

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

2. PROBLEM FORMULATION

We perform an ML operation on a distributed dataset, denoted as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the i-th data point and y_i is the measurement or label information associated with it. Our dataset is distributed in a tree-shaped network consisting of K local workers, as illustrated in Fig. 1. The k-th local worker holds a subset of the dataset, specifically $\{(\boldsymbol{x}_i, y_i)\}_{i\in[k]}$ where $|[k]| < m$ and $k = 1, 2, ..., K$. Our objective is to find the global optimal solution, $\boldsymbol{w}^\star$, by solving the following regularized loss minimization problem with the distributed dataset:

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\ P(\boldsymbol{w}) \triangleq \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^m \ell_i(\boldsymbol{w}^T\boldsymbol{x}_i), \qquad (1)$$

where $\ell_i(\cdot)$, $i = 1, 2, ..., m$, are loss functions, and $\lambda \geq 0$ is a tuning parameter. Depending on the loss functions, the optimization problem (1) can be a regression problem or a classification problem. For instance, when the loss function is $\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) = (\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2$ with measurement data $y_i \in \mathbb{R}$, the problem can be interpreted as a regression problem. When the loss function is $\ell_i(\boldsymbol{w}^T\boldsymbol{x}_i) = \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i))$ with label information $y_i \in \mathbb{R}$, (1) becomes a Support Vector Machine (SVM) classification problem. Furthermore, we assume that the data points are normalized to ensure that the ℓ_2 norm of each $\boldsymbol{x}_i$ is bounded, i.e., $\|\boldsymbol{x}_i\|_2 \leq 1$, for $i = 1, 2, ..., m$.

By considering the conjugate function of the loss function $\ell_i(a)$, defined as $\ell_i(a) = \sup_b ab - \ell_i^*(b)$, we can derive the following dual problem from the primal problem (1):

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}}\ D(\boldsymbol{\alpha}) \triangleq -\frac{\lambda}{2}\|\boldsymbol{A}\boldsymbol{\alpha}\|_2^2 - \frac{1}{m}\sum_{i=1}^m \ell_i^*(-\alpha_i), \qquad (2)$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ is a data matrix whose i-th column is $\frac{1}{\lambda m}\boldsymbol{x}_i$, and α_i is the i-th dual variable corresponding to the i-th datum $\boldsymbol{x}_i$, $i = 1, ..., m$. By defining $\boldsymbol{w}(\boldsymbol{\alpha}) \triangleq \boldsymbol{A}\boldsymbol{\alpha}$, we can have a duality gap as $P(\boldsymbol{w}(\boldsymbol{\alpha})) - D(\boldsymbol{\alpha})$ which can be used as a measurable quantity for how close an estimated solution is to an optimal solution. Remark that the weak duality theorem [21] ensures that for all $\boldsymbol{w}$, $P(\boldsymbol{w}) \geq D(\boldsymbol{\alpha})$.

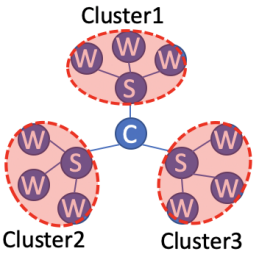


Fig. 1. Illustration of a tree-structured network, which has two layers. In the network, a central station (root node) has three direct child nodes, i.e., sub-central node, denoted by S. Each sub-central node S has three direct child nodes, i.e., local workers, denoted by W.

When we substitute $\boldsymbol{w}$ with $\boldsymbol{w}(\boldsymbol{\alpha})$, this condition still holds with $P(\boldsymbol{w}(\boldsymbol{\alpha})) \geq D(\boldsymbol{\alpha})$. If we can find a variable $\boldsymbol{\alpha}^\star$ such that $P(\boldsymbol{w}(\boldsymbol{\alpha}^\star)) = D(\boldsymbol{\alpha}^\star)$, then, from the strong duality, $\boldsymbol{\alpha}^\star$ and $\boldsymbol{w}(\boldsymbol{\alpha}^\star)$ can be recognized as optimal solutions to the dual problem (2) and the primal problem (1) respectively. This paper aims to design a distributed algorithm that efficiently solves (2) while considering the information of imbalanced data, with the goal of mitigating the effect caused by data imbalance.

3. REVIEW OF DISTRIBUTED ALGORITHMS WITH BALANCED OR IMBALANCED DATA

In previous studies, many researchers have addressed the issue of handling balanced or imbalanced data in the design of distributed algorithms for ML operations. For example, the authors in [22, 23] used the Alternating Direction Method of Multipliers (ADMM) technique to deal with various types of imbalanced data on a star network for classification problems. The authors in [24] considered a separable optimization problem on a star network with imbalanced data in the case when the number of local workers is larger than the number of data points. Unfortunately, most of the research focused on designing distributed algorithms on a star network, which is a very simple and special network topology.

To deal with different types of network topologies, the researchers in [12, 13] extended the Distributed Dual Coordinate Ascent method to a general tree network with p layers (DDCA-Tree), where the root node is located in the 0-th layer and local workers are located in the p-th layer. Under the assumption that every node on the network has balanced data, the design of the distributed algorithm was studied. However, the design of an efficient distributed algorithm that considers imbalanced data in a general tree network has not been fully studied. This paper aims to address this research gap and investigate this topic.

4. GENERALIZED DISTRIBUTED DUAL COORDINATE ASCENT IN A TREE NETWORK

We aim to investigate the effect of imbalanced data on the convergence rate of DDCA-Tree when applied to a general tree network. We will also propose an enhanced version of

DDCA-Tree that takes into account the imbalanced data to improve its performance. Specifically, we will address the following questions: (Q1) How does the convergence rate of DDCA-Tree differ when applied to imbalanced data? (Q2) How can we design DDCA-Tree to better handle imbalanced data by incorporating information about the imbalance?

To mitigate the effect of imbalanced data distribution, we can consider the information on the imbalanced data distribution in the accumulation of global parameters, which is the concept of Generalized distributed Dual Coordinate Ascent on a general tree network (GDCA-Tree). In GDCA-Tree, described in Algorithm 1, we propose to use different weights for global parameters that local workers have in updating a global parameter at the t-th iteration, $\boldsymbol{w}^{(t)}$. More specifically, in the accumulation of the global parameters, by considering the information of imbalanced data, we use a weighted sum updating scheme for global parameter $\boldsymbol{w}^{(t)}$ in a general tree node Q having K child nodes as

$$\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t-1)} + \sum_{k=1}^{K} \beta_k \triangle \boldsymbol{w}_k, \tag{3}$$

where β_k is a weight considering imbalanced data factor for the k-th updating parameter $\triangle \boldsymbol{w}_k$, $\sum_{k=1}^{K} \beta_k = 1$, $0 \leq \beta_k \leq 1$.

For the convergence analysis of GDCA-Tree, let us define the local sub-optimality gap, $\epsilon_{Q,k}$, at the k-th direct child node of a general tree node Q as follows:

$$\epsilon_{Q,k}(\boldsymbol{\alpha}) \triangleq \underset{\hat{\boldsymbol{\alpha}}_{[Q;k]}}{\text{maximize}}\ D(\boldsymbol{\alpha}_{[Q;1]}, ..., \hat{\boldsymbol{\alpha}}_{[Q;k]}, ..., \boldsymbol{\alpha}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[Q;1]}, ..., \boldsymbol{\alpha}_{[Q;k]}, ..., \boldsymbol{\alpha}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}}), \tag{4}$$

where $[Q;k]$ is the set of indices of data points in the k-th direct child node of Q, $\boldsymbol{\alpha}_{[Q;k]}$ is the partial vector of dual vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ that corresponds to the data points that the k-th direct child node of Q has, and $\overline{Q}$ is complement of an index set Q. This local sub-optimality gap represents the maximum objective value gap that the k-th direct child node of Q can achieve from the current $\boldsymbol{\alpha}$ value. With this definition, we introduce the following assumption.

Assumption 4.1 (Geometric improvement of GDCA-Tree at a direct child node)**.** For a tree node Q on the i-th layer, we assume that for any given $\boldsymbol{\alpha}$ at the k-th direct child node of Q, the GDCA-Tree provides an update $\triangle \boldsymbol{\alpha}_{[Q;k]}$ such that

$$\mathbb{E}[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[Q;1]}, ..., \boldsymbol{\alpha}_{[Q;k-1]}, \boldsymbol{\alpha}_{[Q;k]} + \triangle \boldsymbol{\alpha}_{[Q;k]}, ..., \boldsymbol{\alpha}_{[Q;K]}, \boldsymbol{\alpha}_{\overline{Q}})] \leq \Theta_{i+1} \cdot \epsilon_{Q,k}(\boldsymbol{\alpha}), \tag{5}$$

where $\Theta_{i+1} \in [0, 1)$ is local improvement.

It represents that if this assumption holds, then, the expectation of the local sub-optimality gap at the k-th direct child node of a tree node Q is decreased by the factor of local improvement Θ_{i+1}. When using Local Stochastic Dual Coordinate Ascent (LocalSDCA), with a local dataset in a leaf node in the p-th layer of a tree network, it is possible to achieve the following proposition concerning the parameter Θ_{i+1}:

Proposition 4.2. *([8]) Assume that loss functions $\ell_i(\cdot)$ are $1/\gamma$-smooth. For a tree node Q on the $(p-1)$-th layer, its direct child node is a leaf node. Then, for the leaf node B in the p-th layer using LocalSDCA, Assumption 4.1 holds with*

$$\Theta_p = (1 - \frac{\lambda m \gamma}{1 + \lambda m \gamma} \cdot \frac{1}{m_B})^{T_p}, \tag{6}$$

where m_B is the size of data points that the leaf node B has, and T_p is the number of iterations in LocalSDCA.

Algorithm 1: Generalized distributed Dual Coordinate Ascent in a general tree node Q (GDCA-Tree) on the layer-i, $i = 1, 2, ..., p-1$

Input: $T_i \geq 1$, $\boldsymbol{\alpha}_Q$, $\boldsymbol{w}$
Initialization: $\boldsymbol{\alpha}^{(0)}_{[Q;k]} \leftarrow \boldsymbol{\alpha}_{[Q;k]}$ 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**
    $(\triangle \boldsymbol{\alpha}_{[Q;k]}, \triangle \boldsymbol{w}_k) \leftarrow$ GDCA-Tree$(\boldsymbol{\alpha}^{(t-1)}_{[Q;k]}, \boldsymbol{w}^{(t-1)})$
    $\boldsymbol{\alpha}^{(t)}_{[Q;k]} \leftarrow \boldsymbol{\alpha}^{(t-1)}_{[Q;k]} + \beta_k \triangle \boldsymbol{\alpha}_{[Q;k]}$
  end
  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)} + \sum_{k=1}^{K_i} \beta_k \triangle \boldsymbol{w}_k$
end
Output: $\triangle \boldsymbol{\alpha}_Q \triangleq \boldsymbol{\alpha}^{(T_i)}_Q - \boldsymbol{\alpha}^{(0)}_Q$, and $\triangle \boldsymbol{w}_Q \triangleq \boldsymbol{A}_Q \triangle \boldsymbol{\alpha}_Q$

Procedure P. Generalized Distributed Dual Coordinate Ascent (GDCA-Tree) 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 $\triangle \boldsymbol{w}_Q \triangleq \boldsymbol{A}_Q \triangle \boldsymbol{\alpha}_Q$

From Proposition 4.2 and Assumption 4.1, over randomly chosen data points in the LocalSDCA at a leaf node, we can have guaranteed improvement at its parent's node in terms of the expectation of local sub-optimality gap.

For Algorithm 1, we have the following convergence theorem at a general tree node Q:

Theorem 4.3. *For a tree node Q on the i-th layer, $i = 0, ..., p-1$, having K_i direct child nodes satisfying Assumption 4.1, with parameters Θ^1_{i+1}, Θ^2_{i+1}, ..., and $\Theta^{K_i}_{i+1}$. Assume that loss functions $\ell_i(\cdot)$'s are $1/\gamma$-smooth. Then for any input $\boldsymbol{w}$ to Algorithm 1 with T_i iterations, the following geometric convergence rate holds for Q:*

$$\mathbb{E}[D(\boldsymbol{\alpha}_Q^*, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_Q^{(T_i)}, \boldsymbol{\alpha}_{\overline{Q}})] \quad (7)$$
$$\leq \underbrace{\left(\max_{k=1,\dots,K_i}\left(1-(1-\Theta_{i+1}^k)\beta_k\right)\frac{\lambda m\gamma}{\rho_i+\lambda m\gamma}\right)^{T_i}}_{\textit{Convergence bound}}$$
$$\times \left(D(\boldsymbol{\alpha}_Q^*, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_Q^{(0)}, \boldsymbol{\alpha}_{\overline{Q}})\right),$$

where ρ_i is any real number larger than ρ_{min} defined as

$$\rho_{min} \triangleq \underset{\boldsymbol{\alpha}_Q \in \mathbb{R}^{|Q|}}{\textit{maximize}}\ \lambda^2 m^2 \frac{\sum_{k=1}^{K_i} \|\boldsymbol{A}_{[Q;k]}\boldsymbol{\alpha}_{[Q;k]}\|_2^2 - \|\boldsymbol{A}_Q\boldsymbol{\alpha}_Q\|_2^2}{\|\boldsymbol{\alpha}_Q\|_2^2} \geq 0.$$

Proof. The dual objective value for a tree node Q that has K direct child nodes is bounded by the following equation:

$$D(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}) = D(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \sum_{k=1}^{K} \beta_k \triangle \boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})$$
$$\geq \sum_{k=1}^{K} \beta_k D(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}), \quad (8)$$

where $\boldsymbol{\alpha}_{<[k]>}$ is $\boldsymbol{\alpha}_{[k]}$ with zero-padding to increase the dimension, $Q = [1:K] = \cup_{k=1}^{K}[k]$ is the index set corresponding to the node Q, and $\boldsymbol{\alpha}_{\overline{Q}}$ is the un-updated coordinate. The inequality in (8) is obtained by using Jensen's inequality. Then, we have

$$\mathbb{E}\Big[D(\boldsymbol{\alpha}_{[1:K]}^{(t+1)}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\Big]$$
$$\geq \mathbb{E}\Big[\sum_{k=1}^{K} \beta_k \big(D(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}}) - D(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\big)\Big]$$
$$= \mathbb{E}\Big[\sum_{k=1}^{K} \beta_k \big(\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}}) - \epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})\big)\Big]$$
$$= \sum_{k=1}^{K} \beta_k \big(\mathbb{E}[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})] - \mathbb{E}[\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)} + \triangle\boldsymbol{\alpha}_{<[k]>}, \boldsymbol{\alpha}_{\overline{Q}})]\big)$$
$$\geq \sum_{k=1}^{K} \beta_k (1-\Theta^k)\big(\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\big)$$
$$\geq \Bigg(\min_{k=1,2,\dots,K} \underbrace{(1-\Theta^k)\beta_k}_{\substack{\text{imbalanced data factor} \\ \text{and its compensation parameter}}}\Bigg) \underbrace{\sum_{k=1}^{K} \big(\epsilon_{Q,k}(\boldsymbol{\alpha}_{[1:K]}^{(t)}, \boldsymbol{\alpha}_{\overline{Q}})\big)}_{(A)}$$

The lower-bound of (A) can be obtained in Appendix A of [13], which leads to (7). Due to space limitation, we omit the remaining proof. □

This theorem indicates that for a tree node Q, if its child nodes satisfy the local sub-optimality gap with local improvement, then, at the node Q, GDCA-Tree can have the convergence rate shown in (7). From Proposition 4.2, we know that Assumption 4.1 holds at leaf nodes using LocalSDCA. Then, for the convergence of GDCA-Tree in a general tree network, the left hand side of (7) can be considered as the local sub-optimality gap that the tree node Q can achieve from $(\boldsymbol{\alpha}_Q^{(T_i)}, \boldsymbol{\alpha}_{\overline{Q}})$, and the convergence bound in (7) can be thought of as local improvement Θ from the parent node of Q. In this way, the convergence of GDCA-Tree in a whole tree network can be understood in a recursive manner.

4.1. Impact of imbalanced data on the convergence speed

From Proposition 4.2, among all leaf nodes, with the same number of local iterations, T_p, in LocalSDCA, the leaf node having the largest number of data points will have the largest Θ_p (i.e., close to 1) due to $1/m_B$ term in (6), and become a bottleneck in the convergence speed. Thus, $1/m_B$ in (6) (or broadly $(1-\Theta^k)$) can be thought of as imbalanced data factor in a leaf node, while β_k can be considered as a compensation parameter for the imbalanced data factor at the k-th leaf node by providing more weights on a bottleneck node. Note that in the case of balanced data, $\beta_k = 1/K$, $k = 1, ..., K$.

Furthermore, the parameter ρ_{min} is a value indicating the similarity of global parameter $\boldsymbol{w}$'s among the K child nodes. If we have a smaller ρ value, i.e., smaller similarity between the global parameters $\boldsymbol{w}$ in child nodes, then we will have a larger convergence bound. In other words, as the global parameters among child nodes become similar, the convergence speed will decrease.

4.2. Determining the compensation parameter β_k for imbalanced data

In order to determine the weight parameter β_k that compensates imbalanced data effect, we consider the case where each node on a network has different numbers of data points. Suppose we have a general node Q which has K direct child nodes. The k-th child node has a partial dataset, where the set of indices of data points is denoted by $[Q;k]$. Due to the scenario that we consider here, the cardinality of the set $[Q;k]$, i.e., the number of data points in the k-th direct child node of Q, is different from each other. By considering the imbalanced numbers of data scenario, we propose the following weight calculation for β_k in the weighted sum updating scheme (3):

$$\beta_k = |[Q;k]|/|Q|, \quad (9)$$

where the cardinality of the set of indices of data points in the k-th direct child node of Q is represented by $|[Q;k]|$. The weighted sum updating scheme is used to put more weight on local updating parameters obtained from processing more data. The intuition behind this is that if one local worker (e.g., W_A) has most data and other workers (e.g., W_B) have only a few, then, the solution obtained from W_A is prone to be closer to the global solution $\boldsymbol{w}^\star$. Thus, the global parameter $\boldsymbol{w}^{(t)}$ is updated based on the portion of data in each local worker or sub-central node. The computation of β_k can be done as a preprocessing step and is negligible when compared to the coordinate ascent operation.

4.3. Number of iterations considering imbalanced data

The convergence bound in (7) is a function of the number of iterations as well as the imbalanced data factor. Therefore, the question raised is that is there any relationship between the number of local iterations and the imbalanced data? Let us answer this question here. For clarity and simplicity, we consider a cluster shown in Fig. 1, where the sub-central node

is denoted by Q. For the optimal number of local iterations to have minimum execution time in convergence, as in [13], we consider the following optimization problem:

$$\underset{T_p \geq 0}{\text{minimize}} \left(1-(1-\Theta_p^k)\beta_k \frac{\lambda m\gamma}{\rho_i+\lambda m\gamma}\right)^{T_{p-1}}, \tag{10}$$

where T_p and T_{p-1} are the numbers of local iterations in a leaf node and its parent node respectively. Θ_p is introduced in (6). Since the total time, t_{total}, for distributed ML process in a cluster can be expressed as $t_{total} = (t_{lp}T_p + t_{delay} + t_{cp}) \cdot T_{p-1}$, where t_{lp} is local processing time per one iteration in a leaf node, t_{delay} is time for communication delay in round trip between a leaf node and its parent node, i.e., sub-central node in Fig. 1, and t_{cp} is time for accumulation at a sub-central node. Then, by plugging (6) into (10), denoting $\frac{\lambda m\gamma}{1+\lambda m\gamma}$ as c_1, and $\frac{\lambda m\gamma}{\rho_i+\lambda m\gamma}$ as c_2, we have

$$\underset{T_p \geq 0}{\text{minimize}} \left(1-(1-(1-\frac{c_1}{|[Q;k]|})^{T_p})c_2 \cdot \frac{|[Q;k]|}{|Q|}\right)^{\frac{t_{total}}{t_{lp}T_p+t_{delay}+t_{cp}}}.$$

From [13], the optimal number of local iterations T_p at the leaf node can be obtained as follows:

$$T_p = \frac{1}{\ln(1-\frac{c_1}{|[Q;k]|})} W\left((1-\frac{c_1}{|[Q;k]|})^r \ln(1-c_2 \cdot \frac{|[Q;k]|}{|Q|})\right) - r,$$

where $c_1, c_2 \in [0, 1)$, communication delay severity level between the local processing time and the communcation delay is denoted by r, i.e., $r = {}^{(t_{delay}+t_{cp})}/_{t_{lp}}$, and $W(\cdot)$ is the Lambert W-function [25]. If the delay severity r is zero or small enough to ignore, then, the Lambert W-function term can be expressed as a small negative constant, and $\ln(1-x) \approx -x$ for small x, we have

$$T_p \propto |[Q;k]|. \tag{11}$$

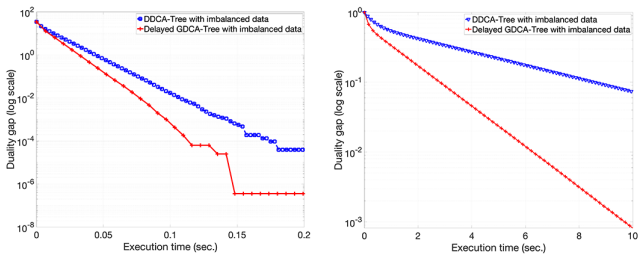
Namely, the number of local iterations need to be increased proportionally to the number of data points in a node in order to minimize the execution time during convergence. Therefore, by taking into account the size of the data in the bottleneck node, we propose the delayed GDCA-Tree, where the number of local iterations is determined by the size of the local dataset in a bottleneck node. In the delayed GDCA-Tree method, the sharing of information between local workers and the sub-central node is delayed, but with a larger number of local iterations. This results in faster convergence speed.

5. NUMERICAL EXPERIMENTS

To validate the performance of the delayed GDCA-Tree, we conduct simulations and compare it to the standard DDCA-Tree when dealing with imbalanced data on a tree network. We test the method on various machine learning tasks including regression and classification using the following datasets: wine quality dataset[1] and Covtype dataset[2].

[1] https://archive.ics.uci.edu/ml/datasets/wine+quality

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



(a) Wine quality (b) Covtype

Fig. 2. Comparison between standard DDCA-Tree (blue) and delayed GDCA-Tree (red) with imbalanced data.

In the wine quality dataset, there are 6493 data points with 12 attributes. The twelfth attribute, quality, is used as the measurement $\boldsymbol{y}$. Each instance is normalized with ℓ_2 norm to ensure $\|\boldsymbol{x}_i\|_2 \leq 1$, $i = 1, ..., m$. To distribute the dataset over a tree network, we organize a two-layered tree network with one central node, two sub-central nodes (denoted by S_1 and S_2), and four local workers (denoted by W_i, $i = 1, ..., 4$). Each sub-central node has two local workers. The tuning parameter λ in (2) is set to 1. We consider the case where each local worker has a different number of data points. To simulate this scenario, we unevenly distribute the wine quality data into four local workers. 30% of the total dataset are allocated to three local workers, W_1, W_2, and W_3 (10% each, which is 649 data points). The remaining 70% of the total dataset (4546 data points) is allocated to one local worker, i.e., W_4, without overlapping the data among local workers.

In the delayed GDCA-Tree, we use (9) to calculate β_k for each local worker. Therefore, for each updating global parameter $\triangle \boldsymbol{w}_k$ that each local worker has from its local dataset, $\beta_1 = \beta_2 = 0.5 (= 649/(649+649))$, $\beta_3 = 0.1249 (= 649/(649+4546))$, and $\beta_4 = 0.8752 (= 4546/(649+4546))$. At the sub-central nodes S_1 and S_2, $0.2 (= (649+649)/6493)$ and $0.8 (= (649+4546)/6493)$ are used for β_k respectively.

In order to obtain statistical results, we run 100 random trials. From the 100 trials, we calculate the average execution time at each outer iteration and compute the average duality gap $P(\boldsymbol{w}(\boldsymbol{\alpha}^{(t)})) - D(\boldsymbol{\alpha}^{(t)})$ at the central station. For the number of local iterations T_p in Procedure P, we use $T_p = 100$. And for delayed GDCA-Tree, we use 300 local iterations for T_p by considering the increased number of data points in W_4. As shown in Fig. 2(a), the delayed GDCA-Tree (red solid line) can improve the convergence speed, compared to the standard DDCA-Tree [13] with imbalanced data (blue dotted line). The figure illustrates that by using the delayed GDCA-Tree, we can mitigate the effect of imbalanced data on distributed dual coordinate ascent in a tree network.

Additionally, we run binary classification with Covtype dataset having a total of 581012 instances and 12 attributes. The dataset is normalized and the labels are set to be in the set {-1,1}. A standard hinge loss and ℓ_2 regularization are

used for a linear SVM. The tree network used has one central node, two sub-central nodes, and eight local workers, where each sub-central node has four local workers. The data is distributed in an imbalanced way, with 5% of the total dataset going to seven local workers, and 65% going to one local worker with no overlap. The number of communications between the local workers and the sub-central node is set to 10. The numbers of local iterations in local workers in the standard DDCA-Tree and the delayed GDCA-Tree are set to 1000 and 4000 respectively. Fig. 2(b) demonstrates that under the imbalanced data scenario, the delayed GDCA-Tree can improve the convergence speed of the DDCA-Tree by considering the information of imbalanced data.

6. REFERENCES

[1] M. Teng and F. Wood, "Bayesian distributed stochastic gradient descent," in *Proceedings of Advances in Neural Information Processing Systems*, 2018, vol. 31.

[2] N. Ferdinand and S. C. Draper, "Anytime stochastic gradient descent: A time to hear from all the workers," in *Proceedings of Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 552–559.

[3] G. Cong, O. Bhardwaj, and M. Feng, "An efficient, distributed stochastic gradient descent algorithm for deep-learning applications," in *Proceedings of International Conference on Parallel Processing*. IEEE, 2017, pp. 11–20.

[4] S. Shi, Q. Wang, X. Chu, and B. Li, "A DAG model of synchronous stochastic gradient descent in distributed deep learning," in *Proceedings of IEEE International Conference on Parallel and Distributed Systems*. IEEE, 2018, pp. 425–432.

[5] N. Ferdinand, B. Gharachorloo, and S. C. Draper, "Anytime exploitation of stragglers in synchronous stochastic gradient descent," in *Proceedings of IEEE International Conference on Machine Learning and Applications*. IEEE, 2017, pp. 141–146.

[6] 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.

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

[8] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Proceedings of Advances in Neural Information Processing Systems*, 2014, pp. 3068–3076.

[9] 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.

[10] Y. Deng and M. Mahdavi, "Local stochastic gradient descent ascent: Convergence analysis and communication efficiency," in *Proceedings of International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1387–1395.

[11] A. Devarakonda, K. Fountoulakis, J. Demmel, and M. W. Mahoney, "Avoiding communication in primal and dual block coordinate descent methods," *SIAM Journal on Scientific Computing*, vol. 41, no. 1, pp. C1–C27, 2019.

[12] 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*, 2019, pp. 3512–3516.

[13] M. Cho, L. Lai, and W. Xu, "Distributed dual coordinate ascent in general tree networks and communication network effect on synchronous machine learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2105–2119, 2021.

[14] 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.

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

[16] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proceedings of International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 3043–3052.

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

[18] 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.

[19] J. Liu, S. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," in *Proceedings of International Conference on Machine Learning*. PMLR, 2014, pp. 469–477.

[20] T. Sun, R. Hannah, and W. Yin, "Asynchronous coordinate descent under more realistic assumptions," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, vol. 30.

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

[22] H. Wang, Y. Gao, Y. Shi, and H. Wang, "A fast distributed classification algorithm for large-scale imbalanced data," in *Proceedings of IEEE International Conference on Data Mining*. IEEE, 2016, pp. 1251–1256.

[23] H. Wang, M. Xiao, C. Wu, and J. Zhang, "Distributed classification for imbalanced big data in distributed environments," *Wireless Networks*, pp. 1–12, 2021.

[24] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[25] 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.