ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1 /24/$31.00 USG | DOI: 10.1109/ICASSP48485.2024.10447341

TREE NETWORK DESIGN FOR FASTER DISTRIBUTED MACHINE LEARNING PROCESS WITH DISTRIBUTED DUAL COORDINATE ASCENT

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

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

ABSTRACT

This paper delves into the subject of designing a tree network, enabling the application of Distributed Dual Coordinate Ascent on a general tree network (DDCA-Tree) introduced in [1–3] for distributed Machine Learning (ML) process. We assume that a network is characterized by communication delays proportional to the distance between any two nodes. To efficiently managing distributed data across the network, we propose the Minimum Worst-Distance Tree (MWDT) algorithm for designing a tree network with a specified target depth yielding a network structure where the communication delay in worst path between a leaf node and its parent node is minimized, consequently enhancing the convergence speed of DDCA-Tree. In numerical experiments, to validate the effectiveness of our approach, we compared the communication delay in worst path on a tree network generated by our algorithm against a minimum spanning tree which provides minimum weight (i.e., distance) sum, and showed our network design has reduced distance in worst path.

Index Terms— Distributed machine learning, Distributed dual coordinate ascent, Network design, Distributed dataset

1. INTRODUCTION

Owing to limited storage capacity within individual computers or servers, a huge amount of data, simply called *big data*, are typically stored in a distributed manner. Consequently, a natural inquiry arises regarding how to effectively deal with these distributed datasets when performing Machine Learning (ML) and Artificial Intelligence (AI) operations. Moreover, while certain distributed algorithms exist for processing such data in ML/AI operations, the act of sharing intermediate results, such as learning parameters, poses a significant challenges due to communication constraints such as limited communication bandwidth, communication delays, as well as restrictions on communication power and energy consumption. This scenario prompts a range of inquiries surrounding the process of distributed ML/AI operations on networks. For instance, how can one design networks to facilitate optimal distributed ML/AI processing? Alternatively, given a specific network, what strategies can be devised to create efficient algorithms tailored for distributed ML/AI operations?

Numerous endeavors have been dedicated to the development of distributed ML/AI algorithms tailored to specific network topologies, such as star, tree, or mesh networks. These efforts have aimed to address the challenges associated with handling distributed datasets. For instance, researchers in [4–8] extensively investigated synchronous Stochastic Gradient Decent (SGD). Moverover, studies like [9–14], delved into synchronous Stochastic Dual Coordinate Ascent (SDCA). Furthermore, asynchronous approaches like asynchronous SGD [15–19] and asynchronous SDCA [20–24] were explored to tackle the intricacies of handling distributed datasets across networks.

A noteworthy observation is that most of the research has primarily focused on designing distributed algorithms specifically for star network topology, which is a rather specialized scenario. It's imperative to stress that ***datasets are not inherently confined to a star network structure***. In many cases, nodes within a network might not directly interface with a central node. To address this, a linear sequence of nodes could be envisioned as a virtual node, directly linked to the central node. Yet, this approach may inadvertently introduce severe communication delays due to the necessity of transmitting intermediate parameters through the node sequence to reach the central node. This brings up a pivotal question: how can one formulate efficient distributed algorithms adaptable to diverse network topologies? To elucidate this query, the authors in [1–3] conducted research to devise a Distributed Dual Coordinate Ascent on a general tree network (DDCA-Tree) for facilitating distributed ML processes with convergence analysis. It is worth noting that since every interconnected network inherently possesses a spanning tree, the methodology introduced in [1–3] can be applied to various network topologies with performance guarantee, as long as there are no isolated nodes within the network. However, there are still remaining questions regarding the design of a network for a given dis-

Email of corresponding author: michael.cho@csun.edu

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

Lifeng Lai's research is supported by NSF under grant ECCS-2000415.

tributed algorithm. This paper delves into this issue within the context of optimizing the ML process through DDCA-Tree.

2. PROBLEM STATEMENT

We are engaged in a machine learning operation applied to a distributed dataset, which we denote as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$. Here, $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the i-th data point, while y_i corresponds to the associated measurement or label information. Our dataset is spread across K nodes, as depicted in Fig. 1(a). Each node, denoted as the k-th node or local worker, contains a subset of the dataset denoted as $\{(\boldsymbol{x}_i, y_i)\}_{i\in[k]}$, where $|[k]| < m$ and $k = 1, 2, ..., K$. Our goal is finding a global optimal solution, $\boldsymbol{w}^\star$, by solving the following regularized loss minimization problem with the distributed dataset:

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

Here, the functions $\ell_i(\cdot)$, $i = 1, 2, ..., m$, serve as loss functions, and $\lambda \geq 0$ is a tuning parameter. Depending on the specific form of the loss functions, the optimization problem (1) can take the shape of either a regression problem or a classification problem. An additional assumption we make is that the data points are normalized, ensuring that the ℓ_2 norm of each $\boldsymbol{x}_i$ remains 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 subsequent dual problem from the primal problem (1):

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

where the matrix $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ is constructed with each column being $\frac{1}{\lambda m}\boldsymbol{x}_i$, and α_i represents the dual variable corresponding to the i-th data point $\boldsymbol{x}_i$, $i = 1, ..., m$. By introducing the notation $\boldsymbol{w}(\boldsymbol{\alpha}) \triangleq \boldsymbol{A}\boldsymbol{\alpha}$, we can establish a duality gap denoted as $P(\boldsymbol{w}(\boldsymbol{\alpha})) - D(\boldsymbol{\alpha})$. It can serve as a measurable indicator of the proximity between an estimated solution and an optimal one. It is noteworthy that the weak duality theorem [25] guarantees that for any $\boldsymbol{w}$, the condition $P(\boldsymbol{w}) \geq D(\boldsymbol{\alpha})$ holds. Substituting $\boldsymbol{w}$ with $\boldsymbol{w}(\boldsymbol{\alpha})$ maintains this condition, resulting in $P(\boldsymbol{w}(\boldsymbol{\alpha})) \geq D(\boldsymbol{\alpha})$. If we can find a specific variable $\boldsymbol{\alpha}^\star$ for which $P(\boldsymbol{w}(\boldsymbol{\alpha}^\star)) = D(\boldsymbol{\alpha}^\star)$, then, the principle of strong duality asserts that $\boldsymbol{\alpha}^\star$ and $\boldsymbol{w}(\boldsymbol{\alpha}^\star)$ are optimal solutions for the dual problem (2) and the primal problem (1) respectively.

This paper aims to design a network, as depicted in Fig. 1, in an optimal manner to efficiently solve (2) with DDCA-Tree. In this problem, we consider a scenario that dataset is distributed across nodes situated at coordinates (x, y) on a plane. The task at hand involves establishing network connections among these nodes, with the communication delay being directly proportional to the Euclidean distance separating any pair of nodes.

3. PREVIOUS WORKS ON NETWORK DESIGN

The optimization of network structures based on given node locations has been a subject of study for several decades. One well-known problem in this domain is the Minimum Spanning Tree (MST) problem [26]. This problem involves creating a communication network with the lowest possible cost, specifically, by minimizing the sum of edge weights in the connected network. Various algorithms, such as Kruskal's algorithm, Prim's algorithm, and the Reverse-Delete algorithm [26], have been developed to address this problem. Previous works have also considered additional constraints on weight sums when finding a minimum spanning tree [27, 28].

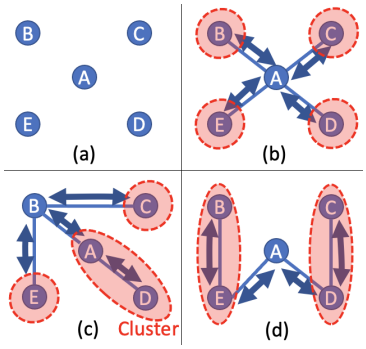


Fig. 1. Example of network design. For given locations of nodes (a), various tree networks (b)-(d) are designed, where narrow blue line, blue arrow, and red dotted area represent communication connection, intermediate parameter sharing and a cluster in a network.

Our current inquiry pertains to whether the minimum spanning tree is the most suitable choice for employing the DDCA-Tree in distributed ML operations. The convergence rate of the DDCA-Tree is influenced not only by the distance (i.e., communication delay) between any two connected nodes within the network but also by other factors such as the tree's depth and width (e.g., the number of clusters or child nodes). These factors were not taken into account when finding the minimum spanning tree. Moreover, to achieve a faster convergence speed in the DDCA-Tree, which is a synchronous algorithm, it is essential to minimize the farthest distances within a cluster rather than between a root node and its direct child nodes (i,e., cluster's root node). This is due to the fact that if communication between a root node and clusters occurs more frequently than communication within a cluster, over multiple iterations, no updated information is shared between the root node and the clusters. Consequently, as illustrated in Fig. 2, we propose a network design that establishes a minimum farthest distance (i.e., bottle-neck path) within a cluster, while ensuring that all nodes remain connected for a given tree depth. This approach allows us to have faster convergence speed of the DDCA-Tree. In the subsequent section, we will present our algorithm in detail.

4. TREE NETWORK DESIGN WITH P LAYERS FOR THE FASTER CONVERGENCE SPEED OF DISTRIBUTED DUAL COORDINATE ASCENT

As documented in prior research [1–3], if we can devise a tree network comprising p layers, we can subsequently derive the

Fig. 2. Illustration of a desired network among central node (i.e., root node) R, sub-central node S, and local worker W in the consideration of communication delay which is proportional to distance. Since communication between a local worker and a sub-central node occurs more frequently than between a sub-central node and a central node, Fig. (a) is a more desired network for faster DDCA-Tree.

convergence outcome of the DDCA-Tree on that tree network with specific depth. Additionally, as depicted in Fig. 2, it is desired to minimize the distance between leaf nodes (i.e., local workers) and their direct parent nodes (i.e., sub-central nodes). This naturally raises the question of how to construct a tree network with p layers while maintaining shorter distances between leaf nodes and their direct parent nodes to enhance the convergence speed of the DDCA-Tree. This section addresses this inquiry by introducing our contribution. We first present a procedure by modifying the MST algorithm to attain a target depth of p for the resultant tree originating from a start node s, denoted as $MST(s,p)$. And then, we introduce our Minimum Worst-Distance Tree (MWDT) algorithm using $MST(s,p)$ in the subsequent subsection.

4.1. Minimum spanning tree with a target depth p

Unlike conventional MST algorithms, such as Prim's and Krustkal's algorithms [26], our objective is to create a tree network with a specific target depth, denoted as p. This network structure enables us to precisely assess the convergence outcomes of the DDCA-Tree. To achieve this goal, we adapt the conventional Prim's algorithm, resulting in a modified version capable of generating a depth-p MST network starting from a designated node. Procedure $MST(s,p)$ provides a detailed description of the steps involved. In essence, when attaching a new node to the current tree using the modified Prim's algorithm, we carefully monitor and evaluate the depth of the node.

4.2. Minimum worst-distance tree between leaf nodes and their direct parent nodes

With our modified MST algorithm, designed to achieve a target depth of p, we introduce an algorithm called as the Minimum Worst-Distance Tree (MWDT). This algorithm aims to minimize the farthest distance between leaf nodes (representing local workers) and their parent nodes. By doing so, we can reduce the bottle-neck path in DDCA-Tree, resulting in enhancing the efficiency of the distributed ML process using the DDCA-Tree with performance guarantees.

In the context of DDCA-Tree, a synchronous algorithm, the communication delay between local workers and their direct parent nodes can become a bottle-neck. Therefore, our algorithm, detailed in Algorithm 1, focuses on minimizing the worst-case communication delay, ultimately in faster convergence of the DDCA-Tree. For a target tree depth of p, our

Procedure $MST(s,p)$: Modified Minimum Spanning Tree(MST) with a target depth p from a start node s

Input: Target tree depth (i.e., number of layers) $p \geq 1$, start node s, weighted adjacency matrix with distance as weight, total number of nodes K, a set of total node V
Initialization: A set of nodes in a tree $C \leftarrow C \cup \{s\}$, a set of edges in a tree $E \leftarrow \phi$
for $k = 1$ **to** $K-1$ **do**
  for $u \in C$ **do**
    for $v \in V \setminus C$ **do**
      Find a node v minimizing a weight between two node v and u, where $v \notin C$, $u \in C$
      if *The depth of the node v from the start node s* $\leq p$ **then**
        $C \leftarrow C \cup \{v\}$
        $E \leftarrow E \cup edge(u,v)$
      end
    end
  end
end
Output: Graph $G(C,E)$

Algorithm 1: Minimum Worst-Distance Tree (MWDT) on leaf-node-connection with p layers

Input: A set of locations of K nodes, $V = \{v_1, v_2, ..., v_K\}$, Target tree depth p
Initialization: Calculate all distances between any two nodes and create weighted adjacency matrix from a fully connected graph, worst distance $d_{min} \leftarrow \infty$
for $s = 1$ **to** K **do**
  Graph $G \leftarrow$ Run Procedure $MST(s,p)$
  $d \leftarrow$ Worst distance on leaf-node connections in G
  if $d_{min} > d$ **then**
    Best graph $G_{best} \leftarrow G$.
  end
end
Output: Minimum worst-distance tree G_{best}

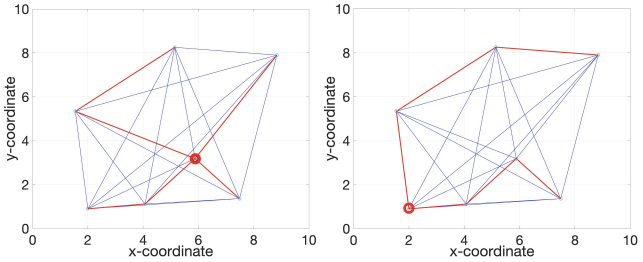
algorithm iteratively applies the $MST(s,p)$ procedure while varying the start node s from 1 to K. This iterative process allows us to select an MST configuration that minimizes the farthest distance between leaf nodes and their parent nodes.

5. NUMERICAL EXPERIMENTS

To assess the performance of our proposed network design strategy for accelerating the convergence of the DDCA-Tree within a tree network containing p layers, we conducted simulations and compared it against the conventional MST algorithm. For node locations, we considered a scenario with $K = 7$ nodes, and randomly selected their coordinates (x, y) within the $[0,10] \times [0,10]$ plane, as stated in Table 1. With the randomly chosen coordinates for the K nodes, we applied our proposed algorithm to obtain a tree network with p layers while minimizing the worst distance between leaf nodes and

Node	1	2	3	4	5	6	7
x-coordinate	5.14	8.84	5.88	1.55	2.00	4.07	7.49
y-coordinate	8.26	7.90	3.19	5.34	0.90	1.12	1.36

Table 1. Example of randomly chosen locations of $K = 7$ nodes on $[0, 10] \times [0, 10]$ plane.



(a) MWDT (b) MST

Fig. 3. Example of tree networks from given locations of nodes in Table 1. (a) Minimum Worst-Distance Tree (MWDT) obtained from Algorithm 1 with target tree depth $p = 2$, whose weight sum is 22.30, (b) Minimum Spanning Tree (MST) obtained from the conventional MST algorithm, whose weight sum is 20.07.

their direct parent nodes. As a point of comparison, we executed the conventional MST algorithm. For the best MST, by varying a start node s from 1 to K, we ran the conventional MST algorithm, and chose one having minimum weight sum as the best MST. In the MST, we chose a root node to have a minimum tree depth. It is worth noting that the depth of the MST may exceed p, as the conventional MST algorithm does not consider the desired depth.

Fig. 3 (a) and (b) show the tree networks obtained from our algorithm providing a tree with the minimum worst-distance on leaf-node connections and the conventional MST algorithm, respectively. In these figures, red solid lines and a red circle represent the network connections among all possible edges in blue lines and a root node, respectively. We compare the worst distance (correspondingly worst communication delay) from one layer to another on the two different tree networks, which becomes the bottle-neck of DDCA-Tree in sharing intermediate results between connected nodes in distributed ML process. Table 2 shows the worst distance of communication route between nodes in adjacent layers. For clear comparison, we calculated the sum of distances in the worst path from a leaf node to a root node in Table 2. As shown in Table 2, the distance between a leaf node to a root node in worst path is reduced, which will be shown in the improvement of convergence speed of DDCA-Tree, even though the weight sum is increased in MWDT. This is because MST provides a tree having the minimum weight sum, while our algorithm focuses on minimizing the worst communication links from a leaf node (i.e., local worker) to its parent node.

In order to obtain statistical results for comparison, we

Comm. Route	Worst dist. on MWDT	Worst dist. on MST
layer 0 $\leftrightarrow$ layer 1	5.5680	4.4640
layer 1 $\leftrightarrow$ layer 2	4.6298	4.6298
layer 2 $\leftrightarrow$ layer 3	-	3.7157
Sum (in worst path)	10.1978	12.8095

Table 2. Worst distance of communication route between nodes in adjacent layers on designed tree networks in Fig. 3.

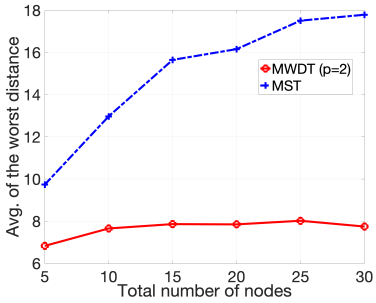


Fig. 4. Average of the worst distance between a leaf node to a root node from 100 random trials for each number of nodes.

varied the number of nodes from 5 to 30. For each specific number of nodes, we conducted 100 trials with randomly chosen node locations and calculated the average of the worst (i.e., bottle-neck) distance from a leaf node to a root node. In the case of MWDT, we set $p = 2$. For MST, we selected a root node among all nodes to minimize the depth of the MST. Fig. 4 illustrates the average worst distance from a leaf node to a root node as the number of nodes increases. As depicted in Fig. 4, MWDT consistently provides a shorter worst-distance between a leaf node to a root node compared to MST.

6. CONCLUSION

This paper explores the design of a tree network for usage of Distributed Dual Coordinate Ascent on a general tree network (DDCA-Tree) for distributed ML processes. We operate under the assumption that the network exhibits communication delays proportional to the distance between any two nodes. To efficiently manage distributed data across the network, we propose Minimum Worst-Distance Tree (MWDT) algorithm which provides a tree network having minimum distance between a leaf node and its parent node for a given tree depth. This results in a network structure that minimizes the communication delay in the worst path, consequently enhancing the convergence speed of DDCA-Tree. To validate the effectiveness of our proposed approach in optimized network design for DDCA-Tree, we conducted numerical experiments involving comparison in the average communication delay in the worst path within a tree network generated by our algorithm against that of a Minimum Spanning Tree (MST), which provides the minimum weight sum, and showed that MWDT can provide a tree network with a target depth having reduced communication delay in worst path.

7. REFERENCES

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

[2] 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.

[3] M. Cho, L. Lai, and W. Xu, "Distributed dual coordinate ascent with imbalanced data on a general tree network," in *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2023, pp. 1–6.

[4] 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.

[5] S. Lee, Q. Kang, A. Agrawal, A. Choudhary, and W.-K. Liao, "Communication-efficient local stochastic gradient descent for scalable deep learning," in *Proceedings of IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 718–727.

[6] D. Das, S. Avancha, D. Mudigere, K. Vaidynathan, S. Sridharan, D. Kalamkar, B. Kaul, and P. Dubey, "Distributed deep learning using synchronous stochastic gradient descent," *arXiv preprint arXiv:1602.06709*, 2016.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2657–2681, 2016.

[14] 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.

[15] 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.

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

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

[18] H. Zhang, C.-J. Hsieh, and V. Akella, "Hogwild++: A new mechanism for decentralized asynchronous stochastic gradient descent," in *Proceedings of IEEE International Conference on Data Mining*. IEEE, 2016, pp. 629–638.

[19] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *Proceedings of International Conference on Machine Learning*. PMLR, 2017, pp. 4120–4129.

[20] Z. Huo, X. Jiang, and H. Huang, "Asynchronous dual free stochastic dual coordinate ascent for distributed data mining," in *Proceedings of IEEE International Conference on Data Mining*, 2018, pp. 167–176.

[21] 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.

[22] B. Gu, X. Geng, W. Shi, Y. Shan, Y. Huang, Z. Wang, and G. Zheng, "Solving large-scale support vector ordinal regression with asynchronous parallel coordinate descent algorithms," *Pattern Recognition*, vol. 109, pp. 107592, 2021.

[23] 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.

[24] 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.

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

[26] J. Kleinberg and E. Tardos, *Algorithm design*, Pearson, 2006.

[27] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[28] S. Pettie and V. Ramachandran, "An optimal minimum spanning tree algorithm," *Journal of the Association for Computing Machinery*, vol. 49, no. 1, pp. 16–34, 2002.