Communication-Efficient Network Topology in Decentralized Learning: A Joint Design of Consensus Matrix and Resource Allocation

Jingrong Wang, *Graduate Student Member, IEEE,* Ben Liang, *Fellow, IEEE,*
Zhongwen Zhu, *Senior Member, IEEE,* Emmanuel Thepie Fapi, and Hardik Dalal

Abstract—In decentralized machine learning over a network of workers, each worker updates its local model as a weighted average of its local model and all models received from its neighbors. Efficient consensus weight matrix design and communication resource allocation can increase the training convergence rate and reduce the wall-clock training time. In this paper, we jointly consider these two factors and propose a novel algorithm termed Communication-Efficient Network Topology (CENT), which reduces the latency in each training iteration by removing unnecessary communication links. CENT enforces communication graph sparsity by iteratively updating, with a fixed step size, a trade-off factor between the convergence factor and a weighted graph sparsity. We further extend CENT to one with an adaptive step size (CENT-A), which adjusts the trade-off factor based on the feedback of the objective function value, without introducing additional computation complexity. We show that both CENT and CENT-A preserve the training convergence rate while avoiding the selection of poor communication links. Numerical studies with real-world machine learning data in both homogeneous and heterogeneous scenarios demonstrate the efficacy of CENT and CENT-A and their performance advantage over state-of-the-art algorithms.

Index Terms—Distributed machine learning, consensus weight matrix, resource allocation, sparse graph

1 Introduction

Large-scale machine learning (ML) often requires distributed storage and computing. Distributed ML is a special form of distributed optimization, often characterized by high-dimensional decision variables that can lead to a huge amount of communication among the participants. While most well-known distributed ML algorithms and systems are built in a centralized fashion (e.g., with a dedicated parameter server) [2]–[4], recent works have demonstrated the efficacy of decentralized ML. In decentralized ML, a network of workers cooperate to train ML models by communicating with their neighbors. This can alleviate the problem of computation and communication bottleneck at a central parameter server.

The training performance of decentralized ML is affected by how the model information is exchanged among neighboring workers. Specifically, in each training iteration, each worker takes a weighted average of the models that are aggregated from its neighbors. Those weights can be stacked into a matrix called the *consensus weight matrix*. It has been shown that the convergence speed of decentralized ML is governed by the second-largest singular value of the consensus weight matrix [5]. The smaller this value is, the higher the convergence rate is, i.e., the fewer iterations are required to achieve the same level of training accuracy in decentralized ML.

The optimal consensus weight matrix that leads to the fastest convergence rate can be obtained by the Fastest Distributed Linear Averaging (FDLA) algorithm, which minimizes the second-largest singular value of the consensus weight matrix [5]. Other variations of FDLA have also been applied to decentralized learning [6]–[8]. In addition to optimization-based solutions, some heuristics based on the Laplacian matrix of the communication graph have been widely used to design the consensus weight matrix, e.g., best constant weight, maximum-degree weight, and Metropolis-Hastings weight [9].

In the above methods, all physical links of the underlying network are used in model training. However, this can lead to inefficient communication among the workers, especially in scenarios where limited network bandwidth is shared among them. Although a more connected network may result in fewer iterations in model training, it also introduces higher communication costs in each iteration [10]. Training ML models over a sparse communication graph could outperform a fully-connected network in terms of the wall-clock training time. This suggested that a properly designed *sparse* consensus weight matrix could accelerate the training process, which is achieved by removing low-quality communication links and by alleviating the impact of straggling workers.

A general design of the consensus weight matrix involves network topology design, since the non-zero elements reflect the chosen topology. Deploying decentralized ML over some standard sparse network topologies, e.g., a

- *J. Wang and B. Liang are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada. E-mail: jr.wang@mail.utoronto.ca, liang@ece.utoronto.ca.*
- *Z. Zhu, E. T. Fapi, and H. Dalal are with Ericsson Global AI Accelerator Montréal, Canada. E-mail: {zhongwen.zhu, Emmanuel.thepie.fapi, hardik.dalal}@ericsson.com.*
- *This work was funded in part by Ericsson and Mitacs of Canada. A preliminary version of this work was presented in IFIP Networking 2022 [1].*

ring, has been investigated to reduce the communication complexity [10]–[15]. However, such an approach is suboptimal because of the inflexibility of the prescribed topology. Further studies have considered finding an optimal network topology with the fastest convergence rate subject to some prescribed communication cost [16]–[23], or finding an optimal network topology that minimizes the communication cost subject to a prescribed convergence rate [5], [24], [25], or a prescribed type of connected graph, e.g., minimum spanning tree [26]. However, the total wall-clock training time is not only determined by the convergence rate but also by the latency in each training iteration, which is dominated by the stragglers and the slowest communication links. Furthermore, efficient communication resource allocation is also of importance to speed up the training process. It is coupled with the choice of the consensus weight matrix and could compensate for the latency introduced by those important but poor-quality links.

In this work, we aim to accelerate the training process of decentralized ML via jointly designing a sparse consensus weight matrix and allocating bandwidth to the communication links. We propose a novel algorithm named Communication-Efficient Network Topology (CENT), as well as a more robust variant that requires less parameter tuning, named CENT with adaptive step size (CENT-A). Both algorithms reduce the latency in each training iteration by enforcing the sparsity of the communication graph while retaining a comparable convergence rate as FDLA. Our main contributions are summarized as follows:

- We formulate the problem of joint consensus weight matrix design and communication resource allocation in decentralized ML, which minimizes the total wall-clock training time subject to a limited communication resource budget. The wall-clock training time is characterized both by the computation and communication latency in each training iteration and by the number of iterations needed to reach convergence.
- We propose a novel CENT algorithm for joint consensus weight matrix design and communication resource allocation. It iteratively enforces graph sparsity while retaining the convergence rate. When enforcing graph sparsity, we weigh each link with additional coefficients based on the link quality to avoid selecting bad links. We show the convergence of CENT. We further analyze the convergence of decentralized ML while applying the output of CENT. We show that CENT can terminate after a finite number of steps while still guaranteeing that its output leads to convergence in decentralized ML.
- To reduce the tuning complexity of CENT, we propose CENT-A, which adaptively updates the tradeoff factor between the ML convergence rate and the network graph sparsity as the algorithm progresses. Rather than enforcing a linear growth of the tradeoff factor as in CENT, we propose to tune the tradeoff factor based on the objective function values, i.e., the ratio between the convergence factor and the sparsity of the consensus weight matrix. We show the convergence of decentralized ML while applying the output of CENT-A.
- Experimental results on decentralized training of neural networks further demonstrate the performance advantage of CENT and CENT-A over state-of-the-art algorithms, in terms of significantly faster wall-clock training time. Moreover, both algorithms provide robust performance under various system setups, e.g., various network sizes and data heterogeneity.

The rest of the paper is organized as follows. Section 2 introduces the related work. In Section 3, the system model of decentralized ML and the problem formulation of wall-clock training time minimization are presented. In Section 4, we present the design of CENT for joint consensus weight matrix design and communication resource allocation. In Section 5, we present the design of CENT-A as well as the corresponding convergence analysis. Experimental results are shown and discussed in Section 6. Section 7 presents the conclusion.

2 Related Work

In this section, we summarize the state of the art in the design of consensus weight matrix for general distributed optimization and distributed learning, with and without considering communication.

2.1 Consensus Weight Matrix Design without Considering Communication

Many designs for the consensus weight matrix have been proposed to ensure the convergence of existing distributed optimization techniques such as distributed averaging [5], distributed subgradient methods [27], weight-averaging [28], and push-pull gradient methods [29], with subsequent works extending their application to decentralized machine learning [30]–[32]. To find the optimal consensus weight matrix that leads to the *fastest* convergence rate in terms of iteration, Xiao and Boyd [5] proposed FDLA that minimizes the spectral norm of the consensus weight matrix. Further studies on the variations of FDLA have been investigated [6]–[8]. Some heuristics to construct the consensus weight matrix have also been proposed to guarantee the convergence of decentralized ML. A naive heuristic is to treat each link equally and design a constant edge weight based on the Laplacian matrix of the graph, e.g., the best constant weight and the maximum degree weight. The consensus weight matrix can also be designed locally by selecting the maximum degree of the two adjacent workers, called Metropolis-Hastings weights [9]. A common disadvantage of the above methods is that they use all physical links of a given underlying network.

2.2 Communication-Efficient Consensus Weight Matrix Design

Although a fully connected network leads to the fastest convergence rate, recent works suggested that a sparse network can lead to faster convergence in terms of the wall-clock training time. Some previous works have examined certain common sparse graphs to facilitate communication-efficient decentralized ML, e.g., ring [10], [11], path [12], regular expander graphs [13], [14], and EquiStatic [15]. However, such sparse graphs are sub-optimal in general.

Existing optimization-based solutions can be grouped into the following two approaches:

2.2.1 Maximizing the convergence rate subject to limited communication budget

Dai and Mesbahi [16] proposed to find the optimal network topology that maximizes the algebraic connectivity subject to a prescribed number of edges, noting that the convergence rate is determined by the algebraic connectivity of the topology, i.e., the second smallest eigenvalue of the Laplacian matrix. Delvenne et al. [17] and Kempton et al. [18] optimized the consensus weight matrix by maximizing the algebraic connectivity of the network subject to upper bounds on the degrees of workers, i.e., the diagonal entries of the weighted Laplacian matrix. Ogiwara et al. [19] maximized the algebraic connectivity subject to constraints on the number of workers and communication links. Similarly, Gusrialdi et al. [20] proposed to remove a prescribed number of links such that the largest eigenvalue of the adjacency matrix is minimized. Tang et al. [21] solved the problem of maximum match for a given graph where each worker only communicate with a single neighbor. Meng et al. [22] solved the link selection problem with reinforcement learning subject to communication resource and energy consumption constraints. Wang et al. [23] proposed to randomly select disjoint pairs of workers based on the matchings' selection probabilities which optimize the algebraic connectivity of the expected topology subject to a given communication budget.

2.2.2 Minimizing the communication cost subject to a required convergence rate

Given the underlying connected graph, Xiao and Boyd [5] extended FDLA to sparse graph design by zeroing out the elements in the consensus weight matrix, subject to a prescribed convergence rate. Rafiee and Bayen [24] minimized the number of communication links subject to the constraint that the algebraic connectivity is larger than a prescribed positive value. Zhou et al. [25] proposed to randomly select neighbors by minimizing the probabilities of the workers selecting themselves. Marfoq et al. [26] proposed to find a strongly connected directed graph such that the time per communication round is minimized.

All of these prior works overlook the impact of consensus weight matrix design on the total wall-clock training time, which is determined by both the convergence rate and the computation and communication latency in each training iteration. The design of the consensus weight matrix should consider the trade-off between the convergence rate and communication latency. Furthermore, since the latency in each training iteration is dominated by the straggling workers and the slowest communication links, efficient communication resource allocation is also essential.

2.3 Other Communication-Efficient ML Techniques

There are many recent works on developing ML techniques with improved communication efficiency, which do not involve network topology design. Examples include compression and coding [33]–[35], gossip and push-sum algorithms [36], [37], and efficient scheduling [38]–[41]. Most of these techniques are orthogonal to our work and can be combined with the proposed solution.

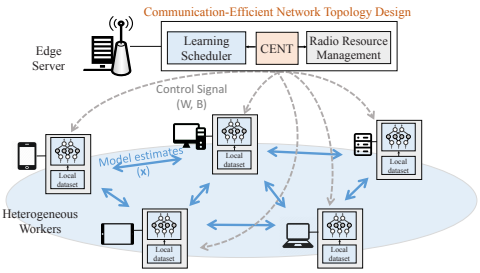

Fig. 1. CENT for decentralized ML.

3 System Model and Problem Formulation

In this section, we present the system model for decentralized ML. We further formulate the problem of joint consensus weight matrix design and communication resource allocation.

3.1 Decentralized ML

We consider a network of workers $\mathcal{N} = \{1, 2, ..., N\}$ that cooperatively train a shared model. The physical network topology is represented by an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{E}$ is the edge set. We have $(i, j) \in \mathcal{E}$ if there exists a link between worker i and worker j and $i \neq j$. We use $A = [A_{i,j}]$ to denote the adjacency matrix of $\mathcal{G}$. Without loss of generality, we assume that $\mathcal{G}$ is connected, i.e., there exists a path between any two workers.

Let $F_i(\mathbf{x}) = \mathbb{E}_{\pi \sim \xi_i} f_i(\mathbf{x}; \pi)$ denote the local training loss function of worker i, where $\mathbf{x}$ denotes the ML model parameters, ξ_i is the set of local data samples at worker i, $f_i(\cdot)$ is the loss function defined by the learning model, e.g., cross-entropy loss. Let $F(\mathbf{x}) = \frac{1}{N} \sum_{i \in \mathcal{N}} F_i(\mathbf{x})$ denote the global training loss over all workers. Typically, decentralized ML can be posed as minimizing the training loss as follows:

$$\min \quad F(\mathbf{x}), \tag{1}$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X}, \tag{2}$$

where $\mathcal{X}$ denotes the feasible set of the training model.

Take the commonly used stochastic gradient descent to solve (1) as an example. A typical training process is as follows. Each worker i stores a local estimate $\mathbf{x}_i$ of the global model $\mathbf{x}$. In each training iteration t, each worker i updates its local model with local gradients $\mathbf{g}_i(t)$ taken at $\mathbf{x}_i(t)$ and transmits the updated model to its neighbors. Each worker then further updates its local model as a weighted average of its local model and all received models. Specifically, let $W_{i,j}$ denote the weight from worker i to worker j for weighted aggregation, and let $W = [W_{i,j}]$. The model update rule is

$$\mathbf{X}(t+1) = \underbrace{\mathcal{P}_{\mathcal{X}}}_{\text{Projection}} \left(\underbrace{(\mathbf{X}(t) - \beta(t)\mathbf{G}(t))}_{\text{Local training}} \underbrace{W}_{\text{Consensus}} \right), \tag{3}$$

where $\mathbf{X}(t) = [\mathbf{x}_1(t), ..., \mathbf{x}_i(t), ..., \mathbf{x}_N(t)]$, $\mathcal{P}_{\mathcal{X}}(\cdot)$ is a projection onto $\mathcal{X}$, $\beta(t)$ is the step size at iteration t, and $\mathbf{G}(t) = [\mathbf{g}_1(t), ..., \mathbf{g}_i(t), ..., \mathbf{g}_N(t)]$. The training process repeats until the convergence of the ML model or until some pre-defined maximum number of training iterations

is reached. Afterward, the training model can be finalized by selecting either one of the local estimates or the average of all local models [4].

3.2 Communication Model

In various application scenarios such as distributed robotic systems in 5G [42], UAV networks [43], and smart connected vehicles [44], decentralized learning with centralized management has demonstrated superior system performance [45]. Furthermore, a central coordinator already exists in many distributed computing and communication systems. For example, in cellular networks, a base station can serve as the coordinator to manage decentralized learning among devices connected by device-to-device (D2D) links. Typically, the base station maintains a global view of the network state information and facilitates centralized optimization for efficient resource allocation among the D2D links. Indeed, 3GPP support for distributed ML promotes using a central controller within the network infrastructure to leverage the established network analytics and control functions [46]. We note that it is important to distinguish the central coordinator from a parameter server. While the coordinator performs topology design and resource allocation, and ensures compliance among all workers, it does not have access to the local ML models or their gradients [21]–[23], [26].

As shown in Fig. 1, the N workers form a decentralized ML network, while a coordinator, e.g., edge server, assists with the design of W and network resource allocation. The edge server communicates with the workers via a dedicated control channel of an access network, e.g., LTE or 5G access. To facilitate D2D communication, we consider frequency division multiple access (FDMA) system with a total bandwidth budget $\bar{B}$. Let $B_{i,j}$ denote the bandwidth allocated to the link from worker i to worker j. The communication latency from i to j is

$$l_{i,j}^{\mathrm{C}}(B_{i,j}) = \frac{D_i}{B_{i,j}\eta_{i,j}}, \forall i \neq j, \tag{4}$$

where D_i is the size of the model sent by worker i and $\eta_{i,j}$ is the spectrum efficiency of the link from worker i to j. In Section 4, we will use the Shannon bound for illustration, such that $\eta_{i,j} = \log_2(1 + \frac{p_i h_{i,j}^2(d_{i,j})}{\sigma_{i,j}^2})$, where p_i is the transmission power of worker i, $d_{i,j}$ is the distance between workers i and worker j, $h_{i,j}^2(d_{i,j})$ is the wireless channel power gain, and $\sigma_{i,j}^2$ is the white noise power. We further define $l_{i,i}^{\mathrm{C}} = 0, \forall i \in \mathcal{N}$. The latency corresponding to the link from worker i to worker j is

$$L_{i,j}(B_{i,j}) = l_i^{\mathrm{P}} + l_{i,j}^{\mathrm{C}}(B_{i,j}), \tag{5}$$

where l_i^{P} denotes the processing latency of worker i for gradient calculation. Let $B = [B_{i,j}]$. The latency in each training iteration is determined by the straggler, which is given by

$$g(W, B) = \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}(B_{i,j}) \mathbb{1}_{\{W_{i,j} \neq 0\}} \right\}, \tag{6}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. Note that $\mathbb{1}_{\{W_{i,j} \neq 0\}} = 0$ indicates that there is no information exchange from worker i to worker j.

We note that in decentralized ML, (6) is determined by the model we select to train as well as the computation capacities of the workers. Once we specify the training model, the coefficients in the latency function can be obtained. We assume that the processing time of the workers $l_i^{\mathrm{P}}, \forall i$, and channel information $\eta_{i,j}, \forall i, j$, are known and constant over the training iterations.

3.3 Problem Formulation

We first characterize the impact of the consensus weight matrix on the training process. Let $\rho(W) = \|W - \frac{\mathbf{1}\mathbf{1}^\top}{N}\|_2$, where $||\cdot||_2$ denotes the spectral norm of a matrix and $\mathbf{1}$ is the all-one column vector. Let T_ϵ denote the number of training iterations required to approximate the ideal training model by a desired error ϵ. Under various assumptions that are common in the literature, it is known that T_ϵ is inversely proportional to $1 - \rho(W)$ [27], [47], [48], i.e.,

$$T_\epsilon \in \mathcal{O}\left(\frac{1}{\epsilon^2(1-\rho(W))}\right). \tag{7}$$

Thus, $\rho(W) < 1$ guarantees the convergence of decentralized ML, and a smaller $\rho(W)$ suggests faster network consensus. For example, (7) in [48] was derived under the following assumptions:

Assumption 1. *(Smoothness): There exists $L > 0$ such that $\|\nabla F_i(\mathbf{x}) - \nabla F_i(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \forall \mathbf{x}$ and $\mathbf{y}$.*

Assumption 2. *(Lower bounded): There exists $F_{\inf}$ such that $F_i(\mathbf{x}) \geq F_{\inf}, \forall \mathbf{x}$.*

Assumption 3. *(Unbiased gradients): $\mathbb{E}_{\xi_i|\mathbf{x}}[\mathbf{g}_i(\mathbf{x})] = \nabla F_i(\mathbf{x}), \forall \mathbf{x}$.*

Assumption 4. *(Bounded variance): There exists $\sigma^2 \geq 0$ such that $\mathbb{E}_{\xi_i|\mathbf{x}}\|\mathbf{g}_i(\mathbf{x}) - \nabla F_i(\mathbf{x})\|^2 \leq \sigma^2, \forall \mathbf{x}$.*

Assumption 5. *(Bounded Dissimilarities): There exists $\kappa^2 \geq 0$ such that $\frac{1}{N}\sum_{i=1}^N \|\nabla F_i(\mathbf{x}) - \nabla F(\mathbf{x})\|^2 \leq \kappa^2, \forall \mathbf{x}$.*

Assumption 6. *(Mixing Matrix): $W\mathbf{1} = \mathbf{1}$, $W = W^\top$, and $\rho(W) < 1$.*

The wall-clock time for training T_ϵ iterations is $T_\epsilon g(W, B)$. Therefore, to reduce the total training time, we should minimize $\frac{1}{1-\rho(W)} g(W, B)$. We formulate the problem of joint consensus weight matrix design and communication resource allocation as follows:

$$\min_{W,B} \quad \frac{1}{1-\rho(W)} g(W, B), \tag{8}$$

$$\text{s.t.} \quad \sum_{i,j \in \mathcal{N}} B_{i,j} \leq \bar{B}, \tag{9}$$

$$B_{i,j} \geq 0, \forall i, j \in \mathcal{N}, \tag{10}$$

$$\rho(W) < 1, \tag{11}$$

$$W\mathbf{1} = \mathbf{1}, \tag{12}$$

$$W = W^\top, \tag{13}$$

$$W \in S_A, \tag{14}$$

where

$$S_A = \{W \in \mathbb{R}^{N \times N} | W_{i,j} = 0 \text{ if } A_{i,j} = 0 \text{ and } i \neq j\}.$$

Constraints (9)-(10) state that the total allocated resource should not exceed the resource budget. Constraint (11) guarantees the convergence of decentralized ML. We set constraints (12) and (13) so that W is a symmetrical matrix and each row or column sums to 1, since using symmetric weights leads to only small decrease in the convergence rate but substantial reduction in computation [5]. Constraint (14) indicates the selected links are restricted by the physical network topology.

The joint design of W and B in problem (8) brings new challenges when compared with optimizing them separately. The choices of consensus matrix and communication resource allocation are coupled and restricted by the physical network topology. Moreover, due to the existence of the indicator function, the objective function of problem (8) is non-smooth and non-convex with respect to W. Finally, due to the vast search space, exhaustive search is computationally expensive.

Remark 1. Existing solutions to the minimization of a multivariable non-convex function cannot be directly applied to problem (8). Since the indicator function is non-differentiable, common gradient-based solutions to non-convex optimization, e.g., successive convex approximation and majorization minimization, are inapplicable to this problem. Another naive solution could be the coordinate descent method, which minimizes the objective function with respect to one decision variable at a time. In our case, if we optimize B with some given W, we will obtain an optimal solution to bandwidth allocation such that the latency is equal for all links. Then, when we optimize W with this B, since at least one link needs to be selected to satisfy constraint (11), the value of $g(W, B)$ in problem (8) stays constant for all feasible W and thus the problem is reduced to FDLA, which means that the non-zero elements of W remain the same as in the previous cycle. As a result, the solutions to bandwidth allocation will not change and the coordinate descent method becomes stuck after the above two iterations.

Remark 2. Constraint (9) considers the case where all workers use orthogonal channels and share the same bandwidth pool. This system setting can also be extended to the case where there is a bandwidth constraint for each worker, i.e.,

$$\sum_{j\in\mathcal{N}} B_{i,j} \leq \bar{B}_i, \forall i, \tag{15}$$

where $\bar{B}_i$ is the resource budget at worker i. Such extensions will not alter the convergence of the proposed algorithms and the details on how to adapt the proposed algorithms CENT and CENT-A to this can be found in Remark 5. When used in conjunction with frequency reuse, (15) helps reduce the spectrum overlapping, and thus mutual interference, among workers. However, the joint design of topology design and frequency reuse, and possibly further transmission scheduling techniques, is a challenging open problem that will be left for future research.

4 Joint Consensus weight matrix Design and Bandwidth Allocation

In this section, we present the design of CENT, which extracts a sparser subgraph of the physical network topology and jointly calculates the consensus weight matrix and bandwidth allocation. We observe from problem (8) that there exists a trade-off between the convergence factor $\rho(W)$ and the latency in each training iteration $g(W, B)$. A more connected network results in a higher convergence rate, but since the network bandwidth is shared among workers, it also leads to a higher communication latency in each training iteration. This inspires us to reduce the communication cost by removing certain communication links, while guaranteeing the convergence of decentralized ML.

4.1 Communication-Efficient Network Topology Design

In this work, we introduce a trade-off factor to balance the convergence rate and the sparsity of the consensus weight matrix. We design this trade-off factor through iterative calculation. Moreover, when enforcing sparsity, we distinguish the links based on the computation time of the workers and the channel conditions of the links.

At step k, we maintain a graph represented by the adjacency matrix $A^{(k)}$. To differentiate links when enforcing graph sparsity, we weigh each link with its latency under equal resource allocation. Thus, we first calculate the number of links in the current graph, i.e., $||A^{(k)}||_{1,1}$, where $||\cdot||_{1,1}$ denotes entry-wise matrix norm, so that $||A^{(k)}||_{1,1} = \sum_{i,j\in\mathcal{N}} |A^{(k)}_{i,j}|$. We *hypothetically* allocate equal resource to each link, i.e., setting $B^{(k)}_{i,j} = \frac{\bar{B}}{||A^{(k)}||_{1,1}}$, for all i and j such that $A^{(k)}_{i,j} = 1$. We consider an estimated latency matrix, $L^{(k)} = [L^{(k)}_{i,j}]$, which indicates the goodness of the links as follows:

$$L^{(k)}_{i,j} = \begin{cases} L_{i,j}(B^{(k)}_{i,j}), & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases} \tag{16}$$

Remark 3. The rationale behind equal resource allocation here is to capture the inherent goodness of the links. Otherwise, suppose we apply min-max resource allocation to optimize the bandwidth allocation and obtain the latency matrix. Then the latency would be equal for all links and that would defeat the purpose of $L^{(k)}$ to differentiate the links. We emphasize that equal resource allocation here is only for calculating $L^{(k)}$. The proposed solution will include optimization of resource allocation.

To enforce graph sparsity and avoid selecting bad links, we consider a trade-off factor, denoted by $\lambda^{(k)}$, between the convergence factor $\rho(W)$ and a weighted graph sparsity $||L^{(k)} \odot W||_{1,1}$ at step k, which will be updated over time. We solve the following optimization problem:

$$\min_{W} \quad \lambda^{(k)}\rho(W) + ||L^{(k)} \odot W||_{1,1}, \tag{17}$$

$$\text{s.t.} \quad W \in S_{A^{(k)}}, \quad (12) - (13), \tag{18}$$

where $\odot$ denotes the Hadamard product of matrices, so that $||L^{(k)} \odot W||_{1,1} = \sum_{i,j\in\mathcal{N}} |L_{i,j}W_{i,j}|$. Here $S_{A^{(k)}}$ follows the same definition as S_A below (14). Since $W \in S_{A^{(k)}}$,

Algorithm 1 Communication-Efficient Network Topology (CENT)

Input: Number of rounds K, set of workers $\mathcal{N}$, physical network topology A, global network conditions $D_i, l_i^{\mathrm{P}}, \eta_{i,j}, \forall i,j$, bandwidth budget $\bar{B}$, and step size $\Delta_\lambda > 0$.
Output: consensus weight matrix $\hat{W}$, bandwidth allocation $\hat{B}$.

1: $\lambda^{(0)} \leftarrow 0, A^{(0)} \leftarrow A$, $\hat{A}^{(0)} = \mathbf{0}$;
2: **for** $k = 0, ..., K-1$ **do**
3: $A^{(k)} \leftarrow A$ if $\hat{A}^{(k)} = \mathbf{0}$; otherwise $A^{(k)} \leftarrow \hat{A}^{(k)}$;
4: $B_{i,j}^{(k)} \leftarrow \frac{\bar{B}}{||A^{(k)}||_{1,1}}, \forall (i,j)$;
▷ Equal allocation among edges
5: Compute matrix $L^{(k)}$ with (16);
▷ Capture the inherent goodness of the links
6: Compute $W^{(k)}$ with (17);
▷ Enforce sparsity and avoid bad links
7: Update the adjacency matrix $\hat{A}^{(k+1)}$ with (19);
▷ Update adjacency matrix
8: **if** $\hat{A}^{(k+1)} = A^{(k)}$ **or** $\hat{A}^{(k+1)} = \mathbf{0}$ **then**
9: $\lambda^{(k+1)} \leftarrow \lambda^{(k)} + \Delta_\lambda$; ▷ Enforce consensus
10: **else**
11: $\lambda^{(k+1)} \leftarrow \lambda^{(k)}$;
12: **end if**
13: **end for**
14: $\hat{W} \leftarrow$ FDLA$(A^{(K)})$;
15: $\hat{B} \leftarrow$ Min-Max-RA$(A^{(K)})$;
16: **return** $\hat{W}, \hat{B}$

we further have $||L^{(k)} \odot W||_{1,1} = \sum_{\{(i,j)|A_{i,j}^{(k)}=1\}} |L_{i,j} W_{i,j}|$. Placing this norm in the objective encourages graph sparsity. Furthermore, since $L^{(k)}$ reflects the quality of the links, a link that corresponds to the stragglers and poor channels is penalized with a large coefficient $L_{i,j}^{(k)}$. We note that constraint (11) is not considered in problem (17) but will be naturally guaranteed by our design, as shown in the next section.

Problem (17) can be efficiently solved through semidefinite programming (SDP). We reformulate (17) by introducing scalar variables s_1 and s_2 to bound $\rho(W)$ and $||L^{(k)} \odot W||_{1,1}$, respectively. Then, we express the norm bound constraint as linear matrix inequalities (LMI):

$$\begin{aligned} \min_{W, s_1, s_2} \quad & \lambda^{(k)} s_1 + s_2, \\ \text{s.t.} \quad & -s_1 I \preceq W - \frac{\mathbf{1}\mathbf{1}^\top}{N} \preceq s_1 I, \\ & \sum_{\{(i,j)|A_{i,j}^{(k)}=1\}} |L_{i,j} W_{i,j}| \le s_2, \\ & W \in S_{A^{(k)}}, \\ & (12) - (13), \end{aligned}$$

where $\preceq$ denotes the general matrix inequality, i.e., for any matrix A and B, $A \preceq B$ means that $B - A$ is a positive semidefinite matrix. The reformulated problem is readily transformed to an SDP with linear objective, LMI constraints, and linear inequality constraints [49].

Let $W^{(k)} = [W_{i,j}^{(k)}]$ be any solution to problem (17). Note that the existence of such an minimizer is guaranteed by the Weierstrass Theorem [50]. Then, the adjacency matrix is updated by replacing nonzero elements of the weight matrix with ones, i.e.,

$$\hat{A}_{i,j}^{(k+1)} = \begin{cases} \mathbb{1}_{\{W_{i,j}^{(k)} \neq 0\}}, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases} \tag{19}$$

If the adjacency matrix changes, we will run the same process to further extract a sparser graph with the current trade-off factor $\lambda^{(k)}$. If the adjacency matrix does not change, we will enforce consensus by increasing the weight of $\rho(W)$ with $\lambda^{(k+1)} = \lambda^{(k)} + \Delta_\lambda$, where Δ_λ is a positive step size.

The above procedure is repeated for K iterations. As shown in the next section, there exists some positive integer k_0, such that for all $K > k_0$, we have $\rho(W^{(K)}) < 1$ and consequently $\rho(\hat{W}) < 1$, meaning that constraint (11) is satisfied by the output of CENT. Furthermore, we will show in the next section that larger K leads to better performance in terms of our objective.

With the extracted sparse topology $A^{(K)}$, the corresponding optimal consensus weight matrix design $\hat{W}$ and bandwidth allocation $\hat{B}$ can be obtained by solving the spectral norm minimization problem

$$\begin{aligned} \min_{W} \quad & \rho(W), \\ \text{s.t.} \quad & W \in S_{A^{(K)}}, \\ & (12) - (13), \end{aligned} \tag{20}$$

and the Min-Max-RA problem

$$\begin{aligned} \min_{B} \quad & \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}(B_{i,j}) \mathbb{1}_{\{A_{i,j}^{(K)} \neq 0\}} \right\}, \\ \text{s.t.} \quad & (9) - (10). \end{aligned} \tag{21}$$

Both are convex problems with known solutions, e.g., FDLA [5] and the primal-dual Lagrangian approach [51].

The pseudocode of CENT is given in Algorithm 1, where lines 2–13 extract a sparse graph $A^{(K)}$ and lines 14–15 compute the solution of consensus weight matrix design $\hat{W}$ and bandwidth allocation $\hat{B}$ based on the extracted graph.

4.2 Convergence Analysis of CENT

We will prove the following two properties: 1) the convergence of CENT, and 2) the convergence of decentralized ML that uses the output of CENT. As illustrated in Fig. 2, we argue that there exists a positive integer k_0 such that $\{\rho(W^{(k)})\}_{k>k_0}$ is a decreasing sequence and bounded below. To guarantee the convergence of decentralized ML, we will first show that $\rho(W^{(k_0+1)}) < 1$ and then with the decreasing sequence $\{\rho(W^{(k)})\}_{k>k_0}$, we conclude that $\rho(\hat{W}) < 1$.

The feasible set at step k of CENT is denoted as

$$\mathcal{W}^{(k)} = \{W | W\mathbf{1} = \mathbf{1}, W = W^\top, W \in S_{A^{(k)}}\}, \forall k. \tag{22}$$

We partition $\mathcal{W}^{(k)}$ into two subsets, i.e.,

$$\begin{aligned} \mathcal{W}_1^{(k)} &= \{W | W \in \mathcal{W}^{(k)}, \rho(W) \ge 1\}, \\ \mathcal{W}_2^{(k)} &= \{W | W \in \mathcal{W}^{(k)}, \rho(W) < 1\}. \end{aligned}$$

In [5] (Section 4), it is shown that, for any connected graph $\mathcal{G}$, we can always design some W such that $\rho(W) < 1$ by applying simple heuristics such as best constant weight,

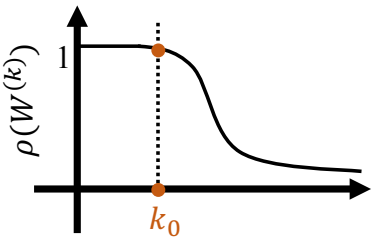

Fig. 2. Schematic of Lemmas 5 and 6.

maximum-degree weight, and Metropolis-Hastings weight. Therefore, we have the following lemma.

Lemma 1. *For connected graph $\mathcal{G}$, $\mathcal{W}_2^{(0)} \neq \emptyset$.*

Furthermore, we make the following observation:

Lemma 2. *For any $W \in \mathcal{W}^{(k)}, \forall k$, we have $||L^{(k)} \odot W||_{1,1} = 0$ if and only if W is the identity matrix I.*

Proof. If $W = I$, we have $||L^{(k)} \odot W||_{1,1} = \sum_{i,j} |L_{i,j}^{(k)} W_{i,j}| = \sum_{i=j} |0 \cdot W_{i,j}| + \sum_{i \neq j} |L_{i,j}^{(k)} \cdot 0| = 0$.

If there exists a pair (i,j) such that $W_{i,j} \neq 0$ and $i \neq j$, we have

$$||L^{(k)} \odot W||_{1,1} = \sum_{\{(i,j)|A_{i,j}^{(k)}=1\}} |L_{i,j}^{(k)} W_{i,j}| \overset{(a)}{>} 0,$$

where (a) holds since $L_{i,j}^{(k)} \neq 0$ for all i and j such that $A_{i,j}^{(k)} = 1$. Therefore, if $||L^{(k)} \odot W||_{1,1} = 0$, we must have $W_{i,j} = 0$ for all i and j such that $A_{i,j}^{(k)} = 1$. Since $W \in \mathcal{W}^{(k)}$, we further have $W = I$. □

Lemma 3. *For any k, if $A^{(k+1)} \neq A^{(k)}$, then we have $L_{i,j}^{(k)} > L_{i,j}^{(k+1)}$ for any i and j such that $L_{i,j}^{(k)} \neq 0$.*

Proof. If $A^{(k+1)} \neq A^{(k)}$, then we have $\lambda^{(k+1)} = \lambda^{(k)}$, $\mathcal{W}^{(k+1)} \subset \mathcal{W}^{(k)}$, and $||A^{(k+1)}||_{1,1} < ||A^{(k)}||_{1,1}$. By equally allocating resource to each link, we have $B_{i,j}^{(k+1)} > B_{i,j}^{(k)}$ for any i and j such that $A_{i,j}^{(k+1)} \neq 0$ (line 4 of Algorithm 1). Since $L_{i,j}$ is strictly decreasing with $B_{i,j}$, we further have $L_{i,j}^{(k)} > L_{i,j}^{(k+1)}$ for any i and j such that $L_{i,j}^{(k)} \neq 0$. □

To facilitate the rest of our analysis, we define

$$\lambda_0 = \inf_{W \in \mathcal{W}_2^{(0)}} \frac{||L^{(0)} \odot W||_{1,1}}{1 - \rho(W)}.$$

Lemma 4. *λ_0 is non-negative and finite.*

Proof. Since $I \notin \mathcal{W}_2^{(0)}$, from Lemma 2, we know that for any $W \in \mathcal{W}_2^{(0)}$, $||L^{(0)} \odot W||_{1,1} > 0$. Further combining the fact that for any $W \in \mathcal{W}_2^{(0)}$, $1 - \rho(W) > 0$, we have $\frac{||L^{(0)} \odot W||_{1,1}}{1-\rho(W)} > 0$. Therefore, we have $\lambda_0 \geq 0$. Finally, λ_0 is finite since $1 - \rho(W) \neq 0$. □

Let $k_0 = \lfloor \frac{\lambda_0}{\Delta_\lambda} \rfloor$, i.e., $k_0 \Delta_\lambda \leq \lambda_0$ and $(k_0 + 1)\Delta_\lambda > \lambda_0$. The following lemma reveals important properties of the proposed algorithm with respect to k_0.

Lemma 5. *For $0 \leq k < k_0$, the identity matrix I is the unique minimizer to problem (17), so $\rho(W^{(k)}) = 1$. For $k = k_0$, $\rho(W^{(k)}) \leq 1$. For $k > k_0$, $\rho(W^{(k)}) < 1$.*

Proof. Let $f^{(k)}(W) = \lambda^{(k)} \rho(W) + ||L^{(k)} \odot W||_{1,1}$.

For $k < k_0$, we have $\lambda^{(k)} \leq k\Delta_\lambda < k_0 \Delta_\lambda \leq \lambda_0$. We start with $k = 0$. For any $W \in \mathcal{W}^{(0)}$, we have

$$\begin{aligned} &f^{(0)}(I) - f^{(0)}(W) \\ &= \lambda^{(0)} - (\lambda^{(0)} \rho(W) + ||L^{(0)} \odot W||_{1,1}) \\ &= \lambda^{(0)}(1 - \rho(W)) - ||L^{(0)} \odot W||_{1,1}. \end{aligned}$$

If $W \in \mathcal{W}_1^{(0)} \setminus I$, from Lemma 2, we know that $||L^{(0)} \odot W||_{1,1} > 0$. Since $\rho(W) \geq 1$, we further have $f^{(0)}(I) - f^{(0)}(W) < 0$. If $W \in \mathcal{W}_2^{(0)}$, we have $\rho(W) < 1$. Combining this with $\lambda^{(0)} < \lambda_0$, we have

$$f^{(0)}(I) - f^{(0)}(W) < \lambda_0(1 - \rho(W)) - ||L^{(0)} \odot W||_{1,1}.$$

Since by definition $\lambda_0 \leq \frac{||L^{(0)} \odot W||_{1,1}}{1-\rho(W)}$, for all $W \in \mathcal{W}_2^{(0)}$, we have $\lambda_0(1 - \rho(W)) - ||L^{(0)} \odot W||_{1,1} \leq 0$. Therefore, $f^{(0)}(I) - f^{(0)}(W) < 0$ and the identity matrix I is the unique minimizer, i.e., $W^{(0)} = I$ and $\rho(W^{(0)}) = 1$. This further implies that $\mathcal{W}^{(1)} = \mathcal{W}^{(0)}$, $L^{(1)} = L^{(0)}$, and $\lambda^{(1)} = \lambda^{(0)} + \Delta_\lambda$. The case of $0 < k < k_0$ can be proved in the same way since $\lambda^{(k)} < \lambda_0$. We conclude that in this case, the identity matrix I is the unique minimizer, i.e., $W^{(k)} = I$ and $\rho(W^{(k)}) = 1$. Note that this further implies that $\mathcal{W}^{(k_0)} = \mathcal{W}^{(0)}$, $L^{(k_0)} = L^{(0)}$, and $\lambda^{(k_0)} = k_0 \Delta_\lambda$.

For $k = k_0$, we have two cases depending on $\lambda^{(k_0)}$. If $\lambda^{(k_0)} < \lambda_0$, we can prove $W^{(k_0)} = I$ and $\rho(W^{(k_0)}) = 1$ in the same way as above. If $\lambda^{(k_0)} = \lambda_0$, for $W \in \mathcal{W}_1^{(k_0)} \setminus I$, from analysis similar to the above, we still have $f^{(k_0)}(I) - f^{(k_0)}(W) < 0$. Now we inspect the set $\mathcal{W}_2^{(k_0)}$. Since $\mathcal{W}^{(k_0)} = \mathcal{W}^{(0)}$ and $L^{(k_0)} = L^{(0)}$, by the definition of λ_0, we have for all $W \in \mathcal{W}_2^{(0)}$,

$$\begin{aligned} &f^{(k_0)}(I) - f^{(k_0)}(W) \\ &= \lambda_0(1 - \rho(W)) - ||L^{(0)} \odot W||_{1,1} \leq 0. \end{aligned}$$

We conclude that the identity matrix I is no worse than any solution in $\mathcal{W}_2^{(k_0)}$, and thus either $W^{(k_0)} = I$ or $W^{(k_0)} \in \mathcal{W}_2^{(0)}$, so $\rho(W^{(k_0)}) \leq 1$.

For $k = k_0 + 1$, we have two cases depending on $W^{(k_0)}$.

i) If $W^{(k_0)} = I$, we have $\mathcal{W}^{(k_0+1)} = \mathcal{W}^{(k_0)} = \mathcal{W}^{(0)}$, $L^{(k_0+1)} = L^{(k_0)} = L^{(0)}$, and $\lambda^{(k_0+1)} = (k_0 + 1)\Delta_\lambda > \lambda_0$. Therefore, for $W \in \mathcal{W}_1^{(k_0+1)} \setminus I$, we still have $f^{(k_0+1)}(I) - f^{(k_0+1)}(W) < 0$. Now we consider the other part of the feasible region $\mathcal{W}^{(k_0+1)}$. By the definition of λ_0, for some $0 < \epsilon < \lambda^{(k_0+1)} - \lambda_0$, there is a $\bar{W}$ such that $\frac{||L^{(0)} \odot \bar{W}||_{1,1}}{1-\rho(\bar{W})} < \lambda_0 + \epsilon$. In this case, since $\mathcal{W}_2^{(k_0+1)} = \mathcal{W}_2^{(0)}$, we also have $\bar{W} \in \mathcal{W}_2^{(k_0+1)}$. We observe that

$$\begin{aligned} &f^{(k_0+1)}(I) - f^{(k_0+1)}(\bar{W}) \\ &= \lambda^{(k_0+1)}(1 - \rho(\bar{W})) - ||L^{(k_0+1)} \odot \bar{W}||_{1,1} \\ &= \lambda^{(k_0+1)}(1 - \rho(\bar{W})) - ||L^{(0)} \odot \bar{W}||_{1,1} \\ &> (\lambda_0 + \epsilon)(1 - \rho(\bar{W})) - ||L^{(0)} \odot \bar{W}||_{1,1} \\ &> 0. \end{aligned}$$

Therefore, any minimizer to problem (17) must be in $\mathcal{W}_2^{(k_0+1)}$ and thus $\rho(W^{(k_0+1)}) < 1$.

ii) If $W^{(k_0)} \neq I$, we further have two cases depending on $A^{(k_0+1)}$ (lines 8-12 of Algorithm 1).

If $A^{(k_0+1)} = A^{(k_0)}$, then we have $\mathcal{W}_1^{(k_0+1)} = \mathcal{W}_1^{(k_0)} = \mathcal{W}_1^{(0)}$, $L^{(k_0+1)} = L^{(k_0)} = L^{(0)}$, and $\lambda^{(k_0+1)} = (k_0+1)\Delta_\lambda > \lambda_0$. This is identical to the previous case and thus $\rho(W^{(k_0+1)}) < 1$.

If $A^{(k_0+1)} \neq A^{(k_0)}$, from Lemma 3, we have $L_{i,j}^{(k_0)} > L_{i,j}^{(k_0+1)}$ for any i and j such that $L_{i,j}^{(k_0)} \neq 0$. This implies that

$$\begin{aligned} & f^{(k_0+1)}(I) - f^{(k_0+1)}(W^{(k_0)}) \\ &= \lambda^{(k_0+1)}(1-\rho(W^{(k_0)})) - ||L^{(k_0+1)} \odot W^{(k_0)}||_{1,1} \\ &= \lambda^{(k_0)}(1-\rho(W^{(k_0)})) - ||L^{(k_0+1)} \odot W^{(k_0)}||_{1,1} \\ &> \lambda^{(k_0)}(1-\rho(W^{(k_0)})) - ||L^{(k_0)} \odot W^{(k_0)}||_{1,1}. \end{aligned}$$

Since $W^{(k_0)}$ is a minimizer at step k_0, the last line above, which equals $f^{(k_0)}(I) - f^{(k_0)}(W^{(k_0)}) \geq 0$, is non-negative. From (19), we also know that $W^{(k_0)} \in \mathcal{W}^{(k_0+1)}$. Therefore, I is not a minimizer at step k_0+1. From analysis similar to the above, we still have $f^{(k_0+1)}(I) - f^{(k_0+1)}(W) < 0, \forall W \in \mathcal{W}_1^{(k_0+1)} \setminus I$, so the minimizer at step k_0+1 must be in $\mathcal{W}_2^{(k_0+1)}$ and thus $\rho(W^{(k_0+1)}) < 1$.

For $k > k_0+1$, we can apply induction using the same analysis as in case ii) above to conclude that $\rho(W^{(k)}) < 1$. □

We can further show that $\{\rho(W^{(k)})\}_{k>k_0}$ is a non-increasing sequence, as stated in the next lemma.

Lemma 6. $\rho(W^{(k)}) \geq \rho(W^{(k+1)}), \forall k > k_0$.

Proof. For $k > k_0$, depending on whether we increase $\lambda^{(k)}$ at step k of CENT (lines 8-12 of Algorithm 1), we have the following two cases.

If $A^{(k+1)} = A^{(k)}$, then $L^{(k+1)} = L^{(k)}$ and $\lambda^{(k)} < \lambda^{(k+1)}$. In this case, we have $W^{(k+1)} \in \mathcal{W}^{(k+1)} = \mathcal{W}^{(k)}$, and from (19), we also have $W^{(k)} \in \mathcal{W}^{(k+1)}$. Let $\eta^{(k)} = ||L^{(k)} \odot W^{(k)}||_{1,1}$. Since $W^{(k)}$ and $W^{(k+1)}$ are minimizers to problem (17) in steps k and $k+1$, respectively, we have

$$\begin{aligned} \lambda^{(k)}\rho(W^{(k)}) + \eta^{(k)} \leq \lambda^{(k)}\rho(W^{(k+1)}) + \eta^{(k+1)}, \\ \lambda^{(k+1)}\rho(W^{(k+1)}) + \eta^{(k+1)} \leq \lambda^{(k+1)}\rho(W^{(k)}) + \eta^{(k)}. \end{aligned}$$

Summing the above inequalities yields

$$(\lambda^{(k+1)} - \lambda^{(k)})(\rho(W^{(k+1)}) - \rho(W^{(k)})) \leq 0.$$

Since $\lambda^{(k)} < \lambda^{(k+1)}$, we must have $\rho(W^{(k)}) \geq \rho(W^{(k+1)})$.

If $A^{(k+1)} \neq A^{(k)}$, we have $\lambda^{(k+1)} = \lambda^{(k)}$. Let $\psi^{(k)}(W)$ denote the ratio between $||L^{(k+1)} \odot W||_{1,1}$ and $||L^{(k)} \odot W||_{1,1}$. As explained in the proof of Lemma 3, in this case we also have $L_{i,j}^{(k)} > L_{i,j}^{(k+1)}, \forall i,j$, so $\psi^{(k)}(W) \leq 1, \forall W \in \mathcal{W}^{(k+1)}$. From Lemma 5, we see that, for $k > k_0$, the identity matrix I is not a minimizer, so $\psi^{(k)}(W) \neq 0, \forall W \in \mathcal{W}^{(k+1)}$. Therefore, the objective of problem (17) in step $k+1$ can be equivalently written as

$$\min_{W \in \mathcal{W}^{(k+1)}} \frac{\lambda^{(k)}}{\psi^{(k)}(W)}\rho(W) + ||L^{(k)} \odot W||_{1,1}.$$

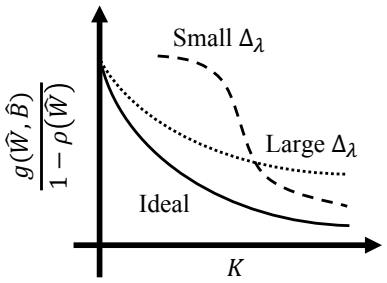

Fig. 3. Impact of step size Δ_λ when updating tradeoff factor λ in CENT.

This has exactly the same form as the previous case, except with $\lambda^{(k)} \leq \frac{\lambda^{(k)}}{\psi^{(k)}(W)}$ instead of $\lambda^{(k)} < \lambda^{(k+1)}$. Using a similar argument, we have $\rho(W^{(k)}) \geq \rho(W^{(k+1)})$. □

Theorem 7. *CENT converges as k approaches infinity. Furthermore, the objective $\frac{1}{1-\rho(W^{(k)})}g(W^{(k)}, B^{(k)})$ is non-increasing in k for $k > k_0$.*

Proof. From Lemma 6, we see that, regardless of whether we increase $\lambda^{(k)}$ at step k of CENT, $\{\rho(W^{(k)})\}_{k>k_0}$ is a non-increasing sequence. Furthermore, this sequence is bounded below by $\arg\min_{W \in \mathcal{W}^{(0)}} \rho(W)$. Therefore, the Monotone Convergence Theorem implies that CENT converges.

As explained in the proof of Lemma 5, $L_{i,j}^{(k)} \geq L_{i,j}^{(k+1)}, \forall i,j,k$. Since $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$, we have

$$\begin{aligned} & g(W^{(k)}, B^{(k)}) = \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}^{(k)} \mathbb{1}_{\{W_{i,j}^{(k)} \neq 0\}} \right\} \qquad (23) \\ & \geq \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}^{(k+1)} \mathbb{1}_{\{W_{i,j}^{(k+1)} \neq 0\}} \right\} = g(W^{(k+1)}, B^{(k+1)}), \ \forall k. \end{aligned}$$

For $k > k_0$, since we also have $\rho(W^{(k)}) < 1$, $\frac{1}{1-\rho(W^{(k)})}g(W^{(k)}, B^{(k)})$ is non-increasing. □

Theorem 8. *Under Assumptions 1–5, if $K > k_0$, decentralized ML converges when the output of CENT is applied.*

Proof. The convergence of decentralized ML is guaranteed, if the consensus weight matrix W satisfies Assumption 6, i.e., $W\mathbf{1} = \mathbf{1}$, $W = W^\top$, and $\rho(W) < 1$ [48]. From Lemma 5, we have $\rho(W^{(k)}) < 1$ for $k > k_0$. Therefore, $\rho(W^{(K)}) < 1$. By solving problem (20), we further have $\rho(\hat{W}) \leq \rho(W^{(K)})$. We conclude that $\rho(\hat{W}) < 1$. Furthermore, from (21), if there exists a link (i,j) such that $A_{i,j}^{(K)} \neq 0$ and $\hat{B}_{i,j} = 0$, the objective function goes to infinity, which is obviously not optimal. Therefore, we have $\hat{B}_{i,j} \neq 0$ for all i and j such that $A_{i,j}^{(K)} \neq 0$. The selected subgraph is connected and each link has a finite delay, which concludes the proof. □

5 CENT WITH ADAPTIVE STEP SIZE (CENT-A)

In CENT, we enforce a linear growth to update the tradeoff factor $\lambda^{(k)}$, which is designed to balance the ML convergence rate and the weighted graph sparsity. To reduce the parameter tuning complexity of CENT, we aim to adaptively tune the tradeoff factor as the algorithm progresses. In this section, we propose CENT-A, short for CENT with adaptive

Algorithm 2 CENT with Adaptive Step Size (CENT-A)

Input: Number of rounds K, set of workers $\mathcal{N}$, physical network topology A, global network conditions $D_i, l_i^{\mathrm{P}}, \eta_{i,j}, \forall i,j$, and bandwidth budget $\bar{B}$.
Output: consensus weight matrix $\hat{W}$, bandwidth allocation $\hat{B}$.

1: $\gamma^{(0)} \leftarrow 0, A^{(0)} \leftarrow A$;
2: **for** $k = 0, ..., K-1$ **do**
3: $B_{i,j}^{(k)} \leftarrow \frac{\bar{B}}{||A^{(k)}||_{1,1}}, \forall (i,j)$; ▷ Equal allocation among edges
4: Compute matrix $L^{(k)}$ with (16); ▷ Capture the inherent goodness of the links
5: Compute $W^{(k)}$ with (24); ▷ Enforce sparsity and avoid bad links
6: Update the adjacency matrix $A^{(k+1)}$ with (19); ▷ Update adjacency matrix
7: **if** $A^{(k+1)} = A^{(k)}$ **then**
8: $\gamma^{(k+1)} \leftarrow \frac{1-\rho(W^{(k)})}{||L^{(k)} \odot W^{(k)}||_{1,1}}$; ▷ Enforce consensus
9: **else**
10: $\gamma^{(k+1)} \leftarrow \gamma^{(k)}$;
11: **end if**
12: **end for**
13: $\hat{W} \leftarrow \text{FDLA}(A^{(K)})$;
14: $\hat{B} \leftarrow \text{Min-Max-RA}(A^{(K)})$;
15: **return** $\hat{W}, \hat{B}$

step size. A proxy for $\lambda^{(k)}$, which is denoted by $\gamma^{(k)}$, is updated directly with the intermediate values of the objective function, i.e., $\rho(W^{(k)})$ and $||L^{(k)} \odot W^{(k)}||_{1,1}$, without introducing an increase in the computation complexity.

5.1 Dependence of Convergence on Step Sizes Δ_λ

CENT is designed to iteratively increase the trade-off factor $\lambda^{(k)}$ in (17) by a user-defined fixed step size Δ_λ. As shown in Theorem 8, CENT requires the number of iterations K to be larger than a certain unknown integer k_0 so as to guarantee the convergence of decentralized ML. The hyperparameter Δ_λ affects the performance of the solutions and needs to be finely tuned. Fig. 3 illustrates the performance of CENT and its dependence on the prescribed fixed step size Δ_λ. CENT with a large Δ_λ might be advantageous at the beginning by surpassing λ_0 quickly but run the risk of overstepping the minimum. However, when the step size Δ_λ is small, the convergence of CENT with respect to the value of the objective function in (8) can be slow.

Such trade-off between some performance metric and the convergence rate has been frequently witnessed in literature, e.g., gradient descent algorithm [52], [53], neural networks training [54], and Kalman filtering [55]. We emphasize here that different from state-of-the-art where the step size may act on the consecutive updates of the decision variables, the step size in this work refers to the update on the tradeoff factor $\lambda^{(k)}$ rather than the decision variables.

5.2 Design of CENT-A

Inspired by the application of Newton's method to fractional programming [56], we observe that the ratio between the convergence factor $1-\rho(W^{(k)})$ and the sparsity of the topology $||L^{(k)} \odot W^{(k)}||_{1,1}$ at each step k is an important quantity that can guide the selection of the trade-off factor in our design.[1] Rather than updating the tradeoff factor with a fixed step size as in CENT, we propose to adaptively choose the trade-off factor based on this quantity.

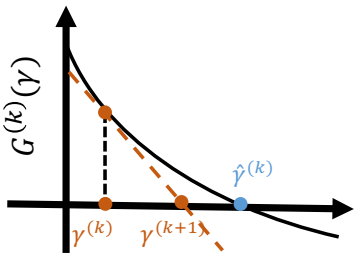

Fig. 4. Illustration of the update on the tradeoff factor $\gamma^{(k)}$ at step k.

The pseudocode of CENT-A is given in Algorithm 2, where lines 2-12 extract a sparse graph $A^{(K)}$ and lines 13-14 compute the solution of consensus weight matrix design $\hat{W}$ and bandwidth allocation $\hat{B}$ based on the extracted graph. At step k, we revisit the intermediate optimization problem (17). By substituting $\lambda^{(k)}$ with $\frac{1}{\gamma^{(k)}}$, we shift the tradeoff weight from the convergence factor $\rho(W)$ to the weighted graph sparsity $||L^{(k)} \odot W||_{1,1}$. Problem (17) can be rewritten as

$$\begin{aligned} \max_{W} \quad & 1-\rho(W)-\gamma^{(k)}||L^{(k)} \odot W||_{1,1}, \\ \text{s.t.} \quad & (12), (13), (18). \end{aligned} \tag{24}$$

We emphasize here that, same as in CENT, constraint (11) is not included in problem (24) but will be naturally guaranteed by our design, as shown in the next section.

After solving problem (24) and obtaining $W^{(k)}$, the adjacency matrix $A^{(k+1)}$ is then updated by replacing nonzero elements of the weight matrix with ones and diagonal elements as zeros. We enforce graph sparsity by updating $\gamma^{(k+1)}$ based on the ratio between the convergence factor and the sparsity of the consensus weight matrix, i.e.,

$$\gamma^{(k+1)} = \frac{1-\rho(W^{(k)})}{||L^{(k)} \odot W^{(k)}||_{1,1}}. \tag{25}$$

The above procedure is repeated for K iterations. Then, with the sparsity pattern $A^{(K)}$, the final solutions $\hat{W}$ and $\hat{B}$ are obtained by solving the convex optimization problems (20) and (21).

Remark 4. As illustrated in Fig. 4, the rationale behind $\gamma^{(k+1)}$ is to obtain a better approximation than $\gamma^{(k)}$ to the root $\hat{\gamma}^{(k)}$ of the function

$$G^{(k)}(\gamma) := \max_{W \in \tilde{\mathcal{W}}^{(k)}} \left\{ 1-\rho(W)-\gamma||L^{(k)} \odot W||_{1,1} \right\}, \tag{26}$$

where $\tilde{\mathcal{W}}^{(k)} = \mathcal{W}^{(k)} \setminus \{I\}$. Note that $\hat{\gamma}^{(k)}$ represents the optimal objective function value of $\max_{W \in \mathcal{W}^{(k)}} \frac{1-\rho(W)}{||L^{(k)} \odot W||_{1,1}}$. By approximating this root, we are enforcing graph sparsity with reduced communication latency and, meanwhile,

1. We remark that existing algorithms for fractional programming, e.g., Dinkelbachs algorithm [56], require a concave-convex problem for convergence. However, problem (8) does not satisfy this condition with respect to the decision variables W and B.

retain a comparable convergence rate. We will show some important properties of $G^{(k)}(\gamma)$ and state in Lemma 13 that $-||L^{(k)} \odot W^{(k)}||_{1,1}$ is a subgradient of $G^{(k)}(\gamma)$ at $\gamma^{(k)}$, $\forall k$. Therefore, the unique root of the linear approximation of $G^{(k)}(\gamma)$ at $\gamma^{(k)}$ is

$$\begin{aligned}\gamma^{(k+1)} &= \gamma^{(k)} - \frac{G^{(k)}(\gamma^{(k)})}{-||L^{(k)} \odot W^{(k)}||_{1,1}} \\ &= \gamma^{(k)} - \frac{\max_{W \in \mathcal{W}^{(k)}} \left\{1 - \rho(W) - \gamma^{(k)}||L^{(k)} \odot W||_{1,1}\right\}}{-||L^{(k)} \odot W^{(k)}||_{1,1}} \\ &= \gamma^{(k)} - \frac{1 - \rho(W^{(k)}) - \gamma^{(k)}||L^{(k)} \odot W^{(k)}||_{1,1}}{-||L^{(k)} \odot W^{(k)}||_{1,1}} \\ &= \frac{1 - \rho(W^{(k)})}{||L^{(k)} \odot W^{(k)}||_{1,1}}.\end{aligned}$$

We note that the intermediate problems (17) and (24) are equivalent when the tradeoff parameter satisfies $\lambda^{(k)} = \frac{1}{\gamma^{(k)}}$. However, CENT and CENT-A invoke two different ways to update these parameters: CENT-A progressively amplifies the importance of the weighted graph sparsity $||L^{(k)} \odot W||_{1,1}$ over steps k, whereas CENT takes the opposite approach. When compared with CENT, CENT-A iteratively updates the altered tradeoff factor $\gamma^{(k)}$ directly based on the value of the objective function in (24). CENT-A enjoys the convenience of eliminating the initialization of the step size Δ_λ, as well as removing the requirement of $K \geq k_0$ in the convergence analysis, as shown in the next section. Furthermore, an outstanding feature of CENT-A is that by adjusting the tradeoff parameter in the opposite manner to CENT, CENT-A avoids generating a disconnected network, ensuring $\rho(W^{(k)}) < 1$, $\forall k$, and $\rho(\hat{W}) < 1$. Consequently, constraint (11) is satisfied for any user-defined maximum number of steps K.

5.3 Convergence Analysis of CENT-A

To guarantee the convergence of decentralized ML, we will show that $\rho(W^{(k)}) < 1, \forall k$ and then we conclude that $\rho(\hat{W}) < 1, \forall K$.

We note that $L^{(k)}$ and $\mathcal{W}^{(k)}$ are the intermediate variables when running CENT-A at the k-th step. For any γ, let $\tilde{W}_\gamma^{(k)}$ denote a maximizer that gives $G^{(k)}(\gamma)$. We first present some properties of the function $G^{(k)}(\gamma)$.

Lemma 9. *$G^{(k)}(\gamma)$ is convex with respect to γ, $\forall k$.*

This conclusion follows directly from the definition of $G^{(k)}(\gamma)$ as the maximum of functions affine in γ.

Lemma 10. *$G^{(k)}(\gamma^{(k)}) \leq G^{(k+1)}(\gamma^{(k)}), \forall k$.*

Proof. For any k, if $A^{(k+1)} = A^{(k)}$, then $L^{(k+1)} = L^{(k)}$. By the definition of $G^{(k)}(\gamma)$, we directly conclude that $G^{(k)}(\gamma^{(k)}) = G^{(k+1)}(\gamma^{(k)})$. If $A^{(k+1)} \neq A^{(k)}$, from Lemma 3, we have $L_{i,j}^{(k)} > L_{i,j}^{(k+1)}$ for any i and j such that $L_{i,j}^{(k)} \neq 0$. We further have

$$\begin{aligned}G^{(k)}(\gamma^{(k)}) &= \max_{W \in \mathcal{W}^{(k)}} \left\{1 - \rho(W) - \gamma^{(k)}||L^{(k)} \odot W||_{1,1}\right\} \\ &= 1 - \rho(W^{(k)}) - \gamma^{(k)}||L^{(k)} \odot W^{(k)}||_{1,1} \\ &\overset{(a)}{\leq} 1 - \rho(W^{(k)}) - \gamma^{(k)}||L^{(k+1)} \odot W^{(k)}||_{1,1} \\ &\overset{(b)}{\leq} \max_{W \in \mathcal{W}^{(k+1)}} \left\{1 - \rho(W) - \gamma^{(k)}||L^{(k+1)} \odot W||_{1,1}\right\} \\ &= G^{(k+1)}(\gamma^{(k)}),\end{aligned}$$

where (a) holds since $||L^{(k)} \odot W^{(k)}||_{1,1} \geq ||L^{(k+1)} \odot W^{(k)}||_{1,1}$ and (b) holds since $W^{(k)} \in \mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$. □

Lemma 11. *$G^{(k)}(\gamma)$ is strictly decreasing with respect to γ, $\forall k$. $G^{(k)}(\gamma) = 0$ has a unique and positive solution, $\forall k$.*

Proof. For any k, γ_1, and γ_2 such that $\gamma_1 < \gamma_2$, we have

$$\begin{aligned}G^{(k)}(\gamma_2) &= \max_{W \in \mathcal{W}^{(k)}} \left\{1 - \rho(W) - \gamma_2||L^{(k)} \odot W||_{1,1}\right\} \\ &= 1 - \rho(\tilde{W}_{\gamma_2}^{(k)}) - \gamma_2||L^{(k)} \odot \tilde{W}_{\gamma_2}^{(k)}||_{1,1} \\ &< 1 - \rho(\tilde{W}_{\gamma_2}^{(k)}) - \gamma_1||L^{(k)} \odot \tilde{W}_{\gamma_2}^{(k)}||_{1,1} \\ &\leq 1 - \rho(\tilde{W}_{\gamma_1}^{(k)}) - \gamma_1||L^{(k)} \odot \tilde{W}_{\gamma_1}^{(k)}||_{1,1} \\ &= G^{(k)}(\gamma_1).\end{aligned}$$

As γ approaches infinity, we have $\lim_{\gamma \to +\infty} G^{(k)}(\gamma) = -\infty, \forall k$. Since $G^{(k)}(\gamma)$ is convex, we conlcude that $G^{(k)}(\gamma)$ is strictly decreasing with respect to $\gamma, \forall k$.

For $k = 0$ and $\gamma = 0$, we have $G^{(0)}(0) = \max_{W \in \mathcal{W}^{(0)}} 1 - \rho(W) > 0$. Since the initial topology $A^{(0)}$ is connected, $\mathcal{W}_2^{(0)} \neq \emptyset$ and we have $G^{(0)}(0) > 0$. From Lemma 10, we further have $G^{(k)}(0) > 0, \forall k$. Since $\lim_{\gamma \to +\infty} G^{(k)}(\gamma) = -\infty, \forall k$, we conclude that the strictly decreasing $G^{(0)}(\gamma) = 0$ has a unique and positive solution. □

As illustrated in Fig. 4, let $\hat{\gamma}^{(k)}$ denote the unique solution such that $G^{(k)}(\gamma) = 0, \forall k$.

Lemma 12. *For any γ such that $\gamma < \hat{\gamma}^{(k)}$, we have $\tilde{W}_\gamma^{(k)} \in \mathcal{W}_2^{(k)}$ and thus $\rho(\tilde{W}_\gamma^{(k)}) < 1, \forall k$.*

Proof. For any k and γ such that $\gamma < \hat{\gamma}^{(k)}$, from Lemma 11, we have $G^{(k)}(\gamma) > 0$.

We proceed to prove the lemma by contradiction. Assume that $\tilde{W}_\gamma^{(k)} \notin \mathcal{W}_2^{(k)}$, i.e., $\tilde{W}_\gamma^{(k)} \in \mathcal{W}_1^{(k)}$, for any γ such that $0 \leq \gamma < \hat{\gamma}^{(k)}$. Let $F^{(k)}(W) = 1 - \rho(W) - \gamma||L^{(k)} \odot W||_{1,1}, \forall k$. We have $F^{(k)}(I) = 0$ and we note that $I \in \mathcal{W}_1^{(k)}$. For any $W \in \mathcal{W}_1^{(k)} \setminus I$, by the definition of $\mathcal{W}_1^{(k)}$, we have $F^{(k)}(W) < F^{(k)}(I) = 0$. This leads to a contradiction with $G^{(k)}(\gamma) = F^{(k)}(\tilde{W}_\gamma^{(k)}) > 0$ and thus we must have $\tilde{W}_\gamma^{(k)} \in \mathcal{W}_2^{(k)}$. We conclude that $\rho(\tilde{W}_\gamma^{(k)}) < 1, \forall k$. □

From the properties of $G^{(k)}(\gamma)$ above, we can infer the following properties of $\gamma^{(k)}$ and $W^{(k)}$.

Lemma 13. *$-||L^{(k)} \odot W^{(k)}||_{1,1}$ is a subgradient of $G^{(k)}(\gamma)$ at $\gamma^{(k)}$, $\forall k$.*

Proof. For any k and γ, we have

$$\begin{aligned}G^{(k)}(\gamma) &= \max_{W \in \mathcal{W}^{(k)}} \left\{1 - \rho(W) - \gamma||L^{(k)} \odot W||_{1,1}\right\} \\ &= 1 - \rho(\tilde{W}_\gamma^{(k)}) - \gamma||L^{(k)} \odot \tilde{W}_\gamma^{(k)}||_{1,1} \\ &\geq 1 - \rho(W^{(k)}) - \gamma||L^{(k)} \odot W^{(k)}||_{1,1} \\ &= 1 - \rho(W^{(k)}) - \gamma^{(k)}||L^{(k)} \odot W^{(k)}||_{1,1} \\ &\quad - (\gamma - \gamma^{(k)})||L^{(k)} \odot W^{(k)}||_{1,1} \\ &= G^{(k)}(\gamma^{(k)}) - (\gamma - \gamma^{(k)})||L^{(k)} \odot W^{(k)}||_{1,1}.\end{aligned}$$ □

Lemma 14. *$\gamma^{(k)} \leq \gamma^{(k+1)} < \hat{\gamma}^{(k)}$ and $\rho(W^{(k)}) < 1, \forall k$.*

Proof. We give a proof by induction. When $k = 0$, from Lemma 11, we have $\hat{\gamma}^{(k)} > 0$. Since $\gamma^{(0)} = 0$, $\gamma^{(0)} < \hat{\gamma}^{(0)}$ is clearly true. Assume the induction hypothesis that for a particular k, $\gamma^{(k)} < \hat{\gamma}^{(k)}$. We have two cases depending on $A^{(k+1)}$ (lines 7-11 of Algorithm 2). If $A^{(k+1)} \neq A^{(k)}$, then $\gamma^{(k)} = \gamma^{(k+1)}$. If $A^{(k+1)} = A^{(k)}$, from Lemmas 9 and 13, we have $\gamma^{(k)} < \gamma^{(k+1)} < \hat{\gamma}^{(k)}$, as illustrated in Fig. 4. From Lemmas 10 and 11, we further have $\gamma^{(k+1)} < \hat{\gamma}^{(k+1)}$. Since both the base case and the induction step have been proved as true, we conclude that $\gamma^{(k)} < \hat{\gamma}^{(k)}, \forall k$. From Lemma 12, we further have $\rho(W^{(k)}) = \rho(\tilde{W}^{(k)}_{\gamma^{(k)}}) < 1, \forall k$. □

Theorem 15. *CENT-A converges as k approaches infinity.*

Proof. Since the number of selected links $||A^{(k)}||_{1,1}$ is decreasing over steps k and bounded below by $N - 1$, the Monotone Convergence Theorem implies the convergence of CENT-A. □

Theorem 16. *Under Assumptions 1–5, for any K, decentralized ML converges when the output of CENT-A is applied.*

Proof. In Lemma 14, we have $\rho(W^{(k)}) < 1, \forall k$. Therefore, for any K, we have $\rho(W^{(K)}) < 1$. By solving problem (20), we further have $\rho(\hat{W}) \leq \rho(W^{(K)}) < 1$. We conclude the proof by noting that the communication latency in each training iteration is finite by solving problem (21). □

Remark 5. If we consider the individual bandwidth constraint in (15), the hypothetical bandwidth allocation in CENT (line 4 of Algorithm 1) and CENT-A (line 3 of Algorithm 2) can be changed to $B^{(k)}_{i,j} \leftarrow \frac{\bar{B}_i}{||A^{(k)}_i||_1}, \forall (i,j)$, where $A^{(k)}_i$ is the i-th row of matrix $A^{(k)}$ and $||\cdot||_1$ denotes the L_1 norm of a vector. As we progressively prune the network, fewer links are selected and more bandwidth is hypothetically allocated to the links. In the extended system settings in Remark 2, since the latency function decreases with increased bandwidth usage in general, Lemma 3 remains valid. Therefore, the convergence analysis of CENT and CENT-A, as well as the convergence of decentralized ML when applying the output of CENT and CENT-A, still holds.

6 Numerical Performance Evaluation

6.1 Decentralized Machine Learning Setup

In this section, we evaluate the performance of CENT and CENT-A on decentralized convolutional neural network (CNN) training over a decentralized network. Unless otherwise specified, we place $N = 50$ workers randomly in a 100 m $\times$ 100 m area. Two workers are connected by an edge if the distance between them is within the communication range 60 m. We evaluate the performance of CENT and CENT-A with 95% confidence intervals, over 100 realizations of the physical graphs. The total bandwidth budget $\bar{B}$ is 20 MHz. The transmission power of each worker is 1 W. The channel power gain $h^2_{i,j}(d_{i,j}) = \gamma_0(\frac{\hat{d}}{d_{i,j}})^4$, where $\gamma_0 = -40$ dB is the path loss at the reference distance $\hat{d} = 1$ m [57].

We consider two ML tasks where N workers collectively train CNNs on the MNIST dataset [58] and the CIFAR10 dataset [59]. The MNIST dataset consists of $60{,}000$ training images and $10{,}000$ testing images, each of which has 28×28 pixels and is labeled between 0 and 9. The CIFAR10 dataset consists of $50{,}000$ training images and $10{,}000$ testing images, each of which contains 32×32 colour images in 10 classes. We consider data heterogeneity among $N = 50$ workers. The training sets of the two datasets are divided into N segments with comparable sizes, each consisting of only p, where $1 \leq p \leq 10$, randomly selected labels. The segments are then randomly allocated to workers. By default, we set $p = 6$. The subsets are randomly distributed to workers. By default, we set $p = 6$. We consider LeNet [60] as a representative of CNN, which is implemented with PyTorch. It is composed of two convolutional layers, followed by two fully connected layers and a softmax classifier. It has $61{,}706$ trainable parameters and the size in memory is 0.35 MB. It is trained with the cross-entropy loss and Adam optimizer with the default learning rate of 0.001. The minibatch size is set to 256 and 8 when training over MNIST and CIFAR10, respectively. In each training iteration, each worker processes a minibatch of the local dataset and then communicates with the adjacent neighbors.

The decentralized CNN training is implemented with torch.distributed and torch.multiprocessing in PyTorch. To represent the heterogeneous computation capacities among workers, we model the processing time of the workers as $l^{\mathrm{P}}_i = \bar{l}^{\mathrm{P}}_i((1-v) + v\phi), \forall i$, where ϕ follows the uniform distribution over $[0, 2]$ and $v = 0.8$ [13]. The average processing time for MNIST is $\bar{l}^{\mathrm{P}}_i = 5.231$ s, which is measured from processing a minibatch of MNIST data samples on a 2.9 GHz Intel Core i5 processor and 8 GB of memory. The average processing time for CIFAR10 is $\bar{l}^{\mathrm{P}}_i = 0.494$ s, which is measured from processing a minibatch of CIFAR10 data samples on an M2 Pro chip and 16 GB of memory.

We compare the performance of CENT and CENT-A with that of the following benchmarks:

- **FDLA** [5]: The weights are calculated by solving the spectral norm minimization problem. It gives the fastest convergence rate in terms of the number of training iterations.
- **Max-degree (MD)** [61]: Assign all edges the same weight based on the maximum degree of the graph.
- **MetropolisHastings (MH)** [62]: Assign each edge a weight based on the maximum degree of its two adjacent workers.
- **Best-constant (BC)** [63]: Assign all edges the same optimal constant weight based on the eigenvalues of the Laplacian matrix of the graph.
- **MATCHA** [23]: Disjoint pairs of workers are randomly selected based on the matchings' selection probabilities which optimize the algebraic connectivity of the expected topology. Assign all links the same constant weight by solving (15).
- **MST** [26]: First hypothetically distribute bandwidth equally to all physical links, and run the minimum spanning tree algorithm with minimal cycle time. Then, assign consensus weight to the links in the tree by running FDLA.

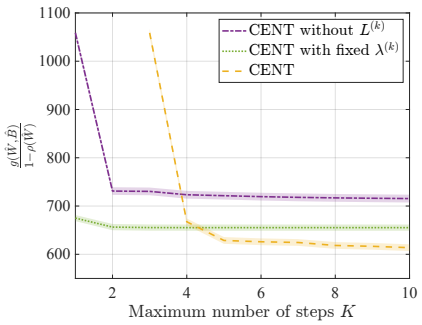

Fig. 5. Objective value over the maximum number of steps K.

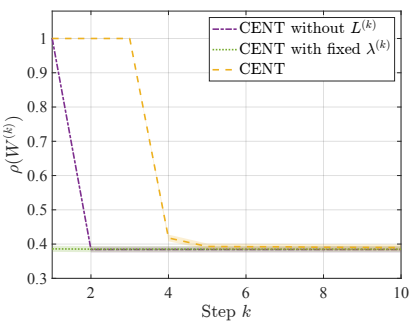

Fig. 6. Convergence factor $\rho^{(k)}$ over steps k.

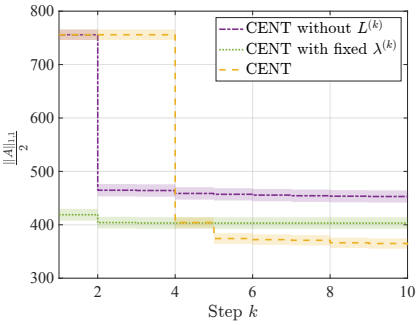

Fig. 7. Number of selected links $||A^{(k)}||_{1,1}$ over steps k.

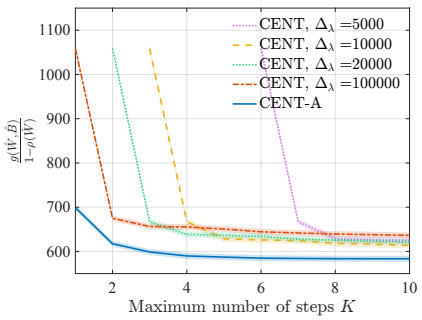

Fig. 8. Objective value over the maximum number of steps K.

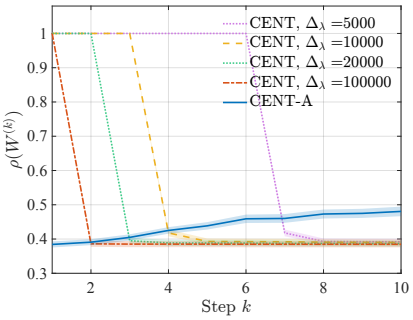

Fig. 9. Convergence factor $\rho^{(k)}$ over steps k.

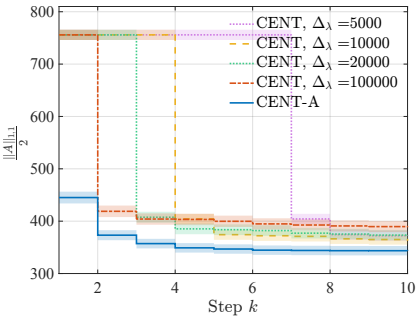

Fig. 10. Number of selected links $\frac{||A^{(k)}||_{1,1}}{2}$ over steps k.

We do not compare with the methods in [10]–[14], [16]–[22], [24], [25], since they are incompatible with the wall-clock training time and thus do not solve our problem. For the above benchmarks, the corresponding bandwidth allocation is calculated by solving Min-Max-RA subject to the constraint of the corresponding network topology. For CENT, by default, Δ_λ is set to 10000.

6.2 Importance of $L^{(k)}$ and $\lambda^{(k)}$ in the Design of CENT

Figs. 5–7 show the impact of using $L^{(k)}$ and $\lambda^{(k)}$ in the design of CENT. We compare CENT with two naive variants: 1) CENT w/o $L^{(k)}$ refers to eliminating $L^{(k)}$ in Algorithm 1, i.e., $L^{(k)}$ is substituted by an all-ones $N \times N$ matrix, and 2) CENT with fixed $\lambda^{(k)}$ refers to fixing $\lambda^{(k)} = 100000, \forall k$ as the algorithm processes. Fig. 5 shows the value of the objective function $\frac{g(\hat{W},\hat{B})}{1-\rho(\hat{W})}$ obtained by CENT under various maximum number of steps K. The selection of K in CENT is restricted by k_0, which, in this case, equals 3. When $K \leq k_0$, we obtain $\rho(\hat{W}) = 1$ indicating that the resulting network topology is disconnected. This accords with the convergence analysis in Section 4.2. When $K > k_0$, we observe that both variants result in an increase in the training time when compared with CENT.

Figs. 6 and 7 show the convergence factor $\rho(W^{(k)})$ and the number of selected links $||A^{(k)}||_{1,1}$ over steps k with $K = 10$. When compared with CENT with fixed $\lambda^{(k)}$, CENT enjoys a faster ML convergence rate, i.e., a smaller convergence factor. When compared with CENT w/o $L^{(k)}$, CENT achieves the same ML convergence rate with fewer selected links, which significantly reduces the per-iteration training latency. This is because the weights $L^{(k)}$ in CENT help reveal the links that dominate the latency in each training iteration, while the increasing sequence of $\lambda^{(k)}$ searches for an appropriate weight on the convergence factor $\rho(W^{(k)})$ to improve the consensus among workers. By jointly considering the design of $L^{(k)}$ and $\lambda^{(k)}$, CENT accelerates the wall-clock training time by removing congested communication links while retaining a high convergence rate.

6.3 Impact of Step Size Δ_λ

Figs. 8–10 compares the performance of CENT-A and CENT under various step size Δ_λ values. When Δ_λ is small, the convergence of CENT can be slow. When Δ_λ is large, CENT runs the risk of overstepping the minimum, resulting in a higher objective function value. As shown in Fig. 8, CENT-A converges faster and obtains a smaller objective function value. Moreover, CENT-A not only enjoys a minimal need for tuning the step sizes but also eliminates the requirement on the selection on K, making it more robust under various system setups.

CENT-A proceeds differently from CENT. As shown in Figs. 9 and 10, CENT starts with an identity matrix and stucks at $\rho(W^{(k)}) = 1$ when $k \leq k_0$. However, CENT-A starts with selecting all physical links. By increasing the weight $\gamma^{(k)}$ on the graph sparsity, poor links are eliminated step by step and CENT-A iteratively makes the network topology sparser. With adaptive step sizes, CENT-A takes larger steps to update the tradeoff factor when it is far away from the optimum. When CENT-A approaches the minimum, since the tradeoff factor is weighted on the graph sparsity, CENT-A takes small steps to increase the tradeoff factor and stops being aggressive on graph sparsification. In our experiment, we observe that CENT-A outperforms

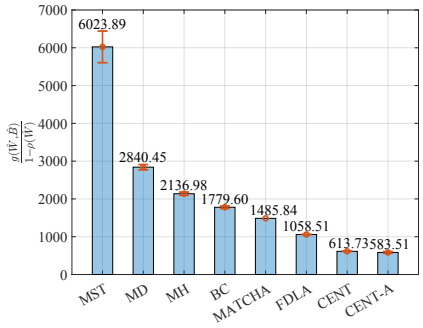

Fig. 11. Comparison on objective value with communication range 60 m.

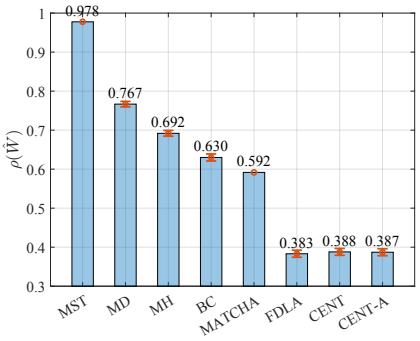

Fig. 12. Comparison on convergence factor with communication range 60 m.

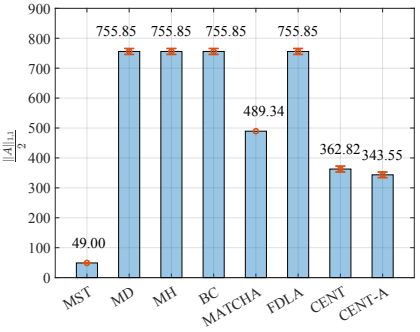

Fig. 13. Comparison on selected links with communication range 60 m.

Algorithm	$\rho(\hat{W})$	$\frac{g(\hat{W},\hat{B})}{1-\rho(\hat{W})}$
MST	0.8821	122.3
MD	0.9252	185.9
MH	0.9191	171.9
BC	0.8852	121.1
MATCHA	0.8895	125.9
FDLA	0.8563	96.7
CENT	0.8563	74.1
CENT-A	0.8563	74.1
OPT	0.8563	74.1

TABLE 1
Comparison with optimal solution.

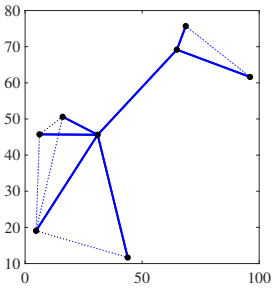

Fig. 14. Topology obtained by CENT.

CENT on average, but there exist realizations of the system where CENT performs better.

6.4 Comparison on Wall-Clock Training Time and Convergence Factor $\rho(\hat{W})$

6.4.1 Comparison with Optimal Solution in Small-Scale Networks

As shown in Fig. 14, we consider a small-scale network which has $N = 8$ workers and 12 edges. The optimal solution to Problem (8), denoted by OPT, is calculated through exhaustive search. Table. 1 compares the performance of various algorithms in terms of the convergence factor $\rho(\hat{W})$ and the theoretical upper bound of the training time $\frac{g(\hat{W},\hat{B})}{1-\rho(\hat{W})}$. We observe that CENT and CENT-A achieve the optimal solution as OPT and obtain the same (fastest) convergence rate as FDLA. With fewer edges selected for communications, CENT and CENT-A significantly reduce the latency in each training iteration.

6.4.2 Comparison in Large-Scale Networks

Figs. 11–13 show the theoretical upper bound of wall-clock training time, the convergence factor $\rho(\hat{W})$, and the number of selected links obtained by various methods, respectively. Note that FDLA by design gives the minimum possible $\rho(\hat{W})$. Since MATCHA randomly samples overlays that could be sparse or even disconnected in each training iteration, we present the value of the convergence factor on the expected activated topology. MST reduces the communication latency in each training iteration by only obtaining $N-1$ links at the expense of a high $\rho(\hat{W})$, resulting in a slow ML convergence rate. CENT and CENT-A require significantly shorter wall-clock training time than the other methods, while retaining similar $\rho(\hat{W})$ as FDLA. CENT reduces the wall-clock training time by 89.8%, 78.4%, 71.3%, 65.5%, 58.7%, and 42%, respectively, when compared with MST, MD, MH, BC, MATCHA, and FDLA. Furthermore, CENT-A achieves shorter wall-clock training time than CENT while maintaining a similar $\rho(\hat{W})$ value, i.e., a slightly faster ML convergence rate in terms of the training iteration.

Figs. 15–17 show the impact of the physical topology density by varying the communication range from 30 m to 60 m. The shorter the communication range is, the sparser the original physical network topology is. Under different settings, CENT and CENT-A consistently attain the lowest objective function value while tracking closely with FDLA in terms of the convergence factor $\rho(\hat{W})$. This suggests that the communication links eliminated by our proposed algorithms not only reduce communication costs but also do not adversely affect the convergence of distributed training. We observe that, as expected, more links are removed when the initial network topology is dense. Overall, CENT and CENT-A efficiently enforce graph sparsity with reduced communication latency and, meanwhile, retain a comparable convergence rate with FDLA.

6.5 Learning Accuracy over Wall-Clock Time

We consider the same graph model as in [5], which has 50 workers and 200 edges. The graph is generated by uniformly distributing 50 workers and increasing the communication range until the total number of edges reaches 200. Fig. 18 shows one realization of the random graphs as well as the corresponding sparse graph extracted by CENT.

Figs. 19 and 20 show the test accuracy of the global model, which is the average of all local CNN models, on the MNIST dataset and CIFAR10 dataset over wall-clock time, respectively. The experimental test accuracy measures the generalization performance of the global model on the data samples it has not seen before. It reflects the convergence of the global averaged model to the optimal ML model as well as the consensus achieved by all workers over a decentralized topology. These figures confirm that even though FDLA by design gives the fastest convergence rate in terms of the training iteration, it does not lead to the fastest convergence speed in terms of the wall-clock time.

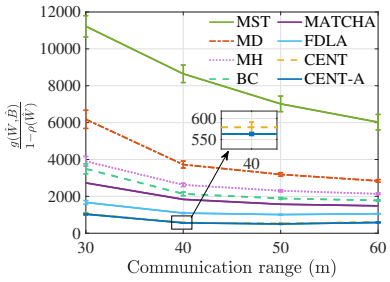

Fig. 15. Comparison on objective value over communication range.

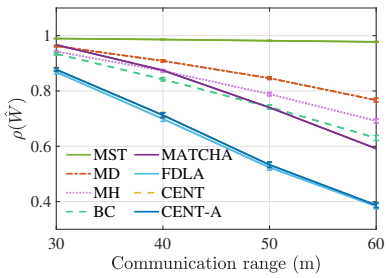

Fig. 16. Comparison on convergence factor over communication range.

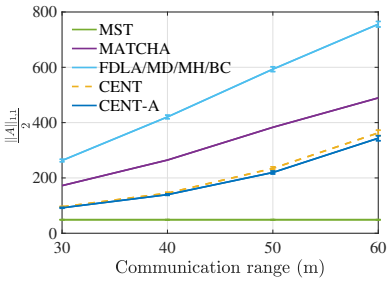

Fig. 17. Comparison on selected links over communication range.

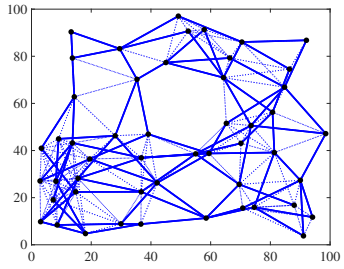

Fig. 18. One realization of sparse graph obtained by CENT.

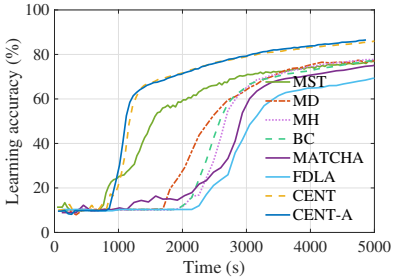

Fig. 19. Learning accuracy on MNIST in wall-clock time with $N = 50$.

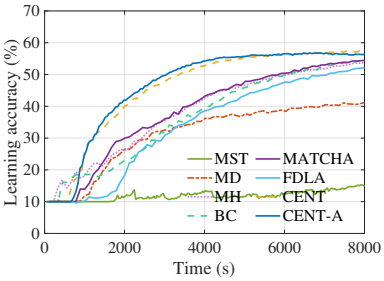

Fig. 20. Learning accuracy on CIFAR10 in wall-clock time with $N = 50$.

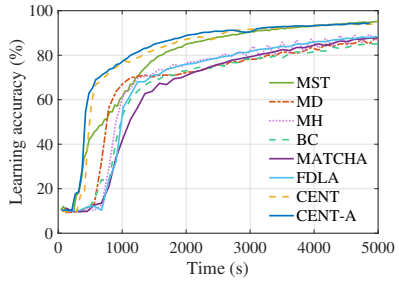

Fig. 21. Learning accuracy in wall-clock time with $N = 20$.

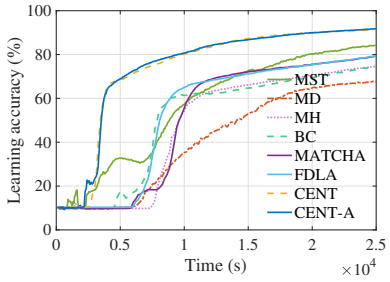

Fig. 22. Learning accuracy in wall-clock time with $N = 100$.

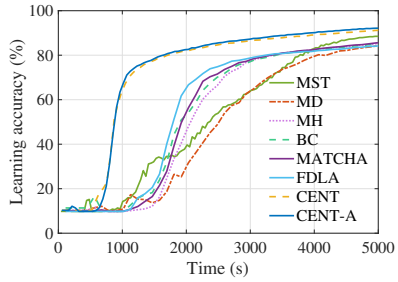

Fig. 23. Learning accuracy in wall-clock time with $p = 10$.

Moreover, the final test accuracy achieved by CENT and CENT-A is comparable to the benchmarks that utilize all physical links of the underlying network, demonstrating the efficacy of the design network topology that retains a favorable convergence rate.

6.5.1 *Impact of Network Size*

Figs. 21 and 22 show the test accuracy of the global model on MNIST when the initial graph has $N = 20$ workers with 80 edges and $N = 100$ with 400 edges, respectively. With more workers sharing limited bandwidth, the communication latency in each training iteration increases and gradually overweighs the computation cost. Therefore, with more workers engaged in decentralized training, the communication bandwidth is emerging as a bottleneck. Blindly increasing the number of workers and utilizing all physical links among them can aggravate network congestion, canceling out the benefits of parallel computing. With fewer edges selected for communications and the reduced latency in each training iteration, CENT and CENT-A demonstrate their performance advantage over the benchmarks in terms of the wall-clock time.

6.5.2 *Impact of Data Heterogeneity*

Fig. 19 in the previous subsection gives the learning accuracy with $p = 6$, and we further present Figs. 23–26 for $p = 10$, 8, 4, and 2, respectively. The smaller the p is, the larger the variation is among the distributions of the local training data. Highly heterogeneous data is known to create more challenges for reaching consensus among workers. A more heterogeneous data distribution aggravates the final test accuracy achieved by all benchmarks. It requires more iterations to reach consensus among workers and thus takes a longer time to converge to the same level of test accuracy. Therefore, in the presence of data heterogeneity, the learning process of all methods inevitably slows down. However, such slow-down can be compensated, to some extent, by significantly reduced latency in each training iteration. We observe the performance advantages of CENT and CENT-A over the benchmarks, in terms of the wall-clock time,

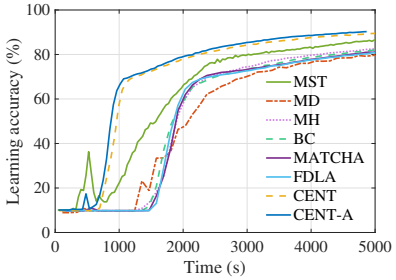

Fig. 24. Learning accuracy in wall-clock time with $p = 8$.

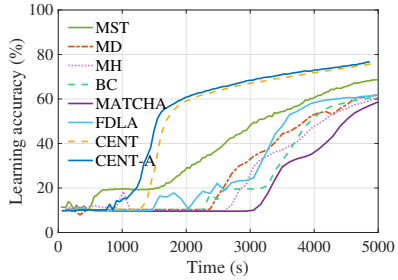

Fig. 25. Learning accuracy in wall-clock time with $p = 4$.

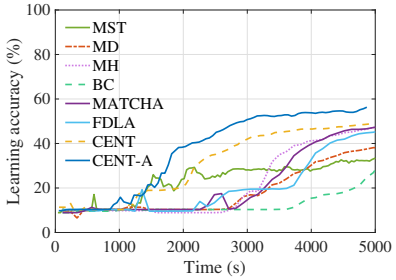

Fig. 26. Learning accuracy in wall-clock time with $p = 2$.

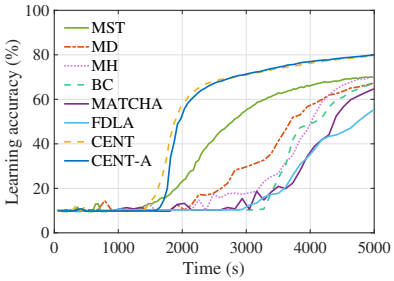

Fig. 27. Learning accuracy in wall-clock time with learning rate 0.0005.

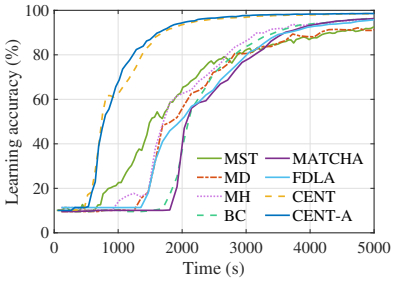

Fig. 28. Learning accuracy in wall-clock time with learning rate 0.01.

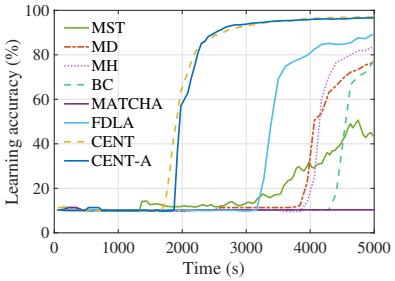

Fig. 29. Learning accuracy in wall-clock time with learning rate 0.02.

become even more substantial under higher data heterogeneity.

6.5.3 *Impact of Learning Rate*

We further conduct comparisons of different design algorithms across additional learning rate values. As shown in Figs. 27–29, we have considered learning rates of 0.005, 0.01, and 0.02. In further experiments we have observed that for learning rates greater than 0.02, all methods start to experience difficulty to converge. These figures show that the proposed CENT and CENT-A consistently outperform the other methods in all scenarios.

7 Conclusion and Future Work

In this paper, we have explored communication design for decentralized ML with bandwidth constraints and heterogeneous workers. By jointly designing a consensus matrix and allocating bandwidth allocation to the communication links, we propose a novel algorithm, termed CENT, as well as a more adaptive variant that requires less parameter tuning, termed CENT-A. Both algorithms speed up the training process by eliminating unnecessary communication links for more efficient bandwidth allocation. Numerical studies on real-world decentralized CNN training show that CENT and CENT-A require significantly shorter wall-clock training time than the state of the art, while retaining similar value for $\rho(\hat{W})$ as FDLA. For future work, efficient consensus weight matrix design for dynamic networks can be explored when further considering worker mobility and heterogeneous resources.

References

[1] J. Wang, B. Liang, Z. Zhu, E. T. Fapi, and H. Dalal, "Joint consensus matrix design and resource allocation for decentralized learning," in *Proc. IFIP Networking*, 2022.

[2] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. NeurIPS*, 2014.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017.

[4] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1–33, 2020.

[5] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[6] M. Jameel, S. Jawed, and L. Schmidt-Thieme, "Optimal topology search for fast model averaging in decentralized parallel SGD," in *Proc. PAKDD*, 2020.

[7] Y. Dandi, A. Koloskova, M. Jaggi, and S. U. Stich, "Data-heterogeneity-aware mixing for decentralized learning," *arXiv preprint arXiv:2204.06477*, 2022.

[8] B. Le Bars, A. Bellet, M. Tommasi, E. Lavoie, and A. Kermarrec, "Refined convergence and topology learning for decentralized optimization with heterogeneous data," *Proc. AISTATS*, 2023.

[9] A. Nedić, 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.

[10] X. Lian, C. Zhang, H. Zhang, C. J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. NeurIPS*, 2017.

[11] W. Zhang, X. Cui, A. Kayi, M. Liu, U. Finkler, B. Kingsbury, G. Saon, Y. Mroueh, A. Buyuktosunoglu, P. Das, D. Kung, and M. Picheny, "Improving efficiency in large-scale decentralized distributed training," in *Proc. IEEE ICASSP*, 2020.

[12] H. Wang, Y. Gao, Y. Shi, and R. Wang, "Group-based alternating direction method of multipliers for distributed linear classification," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3568–3582, 2017.

[13] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *Proc. IEEE INFOCOM*, 2019.

[14] Y. Hua, K. Miller, A. L. Bertozzi, C. Qian, and B. Wang, "Efficient and reliable overlay networks for decentralized federated learning," *SIAM Journal on Applied Mathematics*, vol. 82, no. 4, pp. 1558–1586, 2022.

[15] Z. Song, W. Li, K. Jin, L. Shi, M. Yan, W. Yin, and K. Yuan, "Communication-efficient topologies for decentralized learning with $O(1)$ consensus rate," in *Proc. NeurIPS*, 2022.

[16] R. Dai and M. Mesbahi, "Optimal topology design for dynamic networks," in *Proc. IEEE CDC-ECC*, 2011.

[17] J. C. Delvenne, R. Carli, and S. Zampieri, "Optimal strategies in the average consensus problem," *Systems & Control Letters*, vol. 58, no. 10-11, pp. 759–765, 2009.

[18] L. Kempton, G. Herrmann, and M. di Bernardo, "Adaptive weight selection for optimal consensus performance," in *Proc. IEEE CDC*, 2014.

[19] K. Ogiwara, T. Fukami, and N. Takahashi, "Maximizing algebraic connectivity in the space of graphs with a fixed number of vertices and edges," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 359–368, 2017.

[20] A. Gusrialdi, Z. Qu, and S. Hirche, "Distributed link removal using local estimation of network topology," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 280–292, 2019.

[21] Z. Tang, S. Shi, and X. Chu, "Communication-efficient decentralized learning with sparsification and adaptive peer selection," in *Proc. ICDCS*, 2020.

[22] Z. Meng, H. Xu, M. Chen, X. Yang, Y. Zhao, and C. Qiao, "Learning-driven decentralized machine learning in resource-constrained wireless edge computing," in *Proc. IEEE INFOCOM*, 2021.

[23] J. Wang, A. K. Sahu, G. Joshi, and S. Kar, "MATCHA: A matching-based link scheduling strategy to speed up distributed optimization," *IEEE Transactions on Signal Processing*, vol. 70, pp. 5208–5221, 2022.

[24] M. Rafiee and A. M. Bayen, "Optimal network topology design in multi-agent systems for efficient average consensus," in *Proc. IEEE CDC*, 2010.

[25] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, "Communication-efficient decentralized machine learning over heterogeneous networks," in *Proc. IEEE ICDE*, 2021.

[26] O. Marfoq, C. Xu, G. Neglia, and R. Vidal, "Throughput-optimal topology design for cross-silo federated learning," in *Proc. NeurIPS*, 2020.

[27] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[28] A. Nedić and J. Liu, "On convergence rate of weighted-averaging dynamics for consensus problems," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 766–781, 2017.

[29] S. Pu, W. Shi, J. Xu, and A. Nedić, "Push-pull gradient methods for distributed optimization in networks," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 1–16, 2021.

[30] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. ICML*, 2019.

[31] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *Proc. ICLR*, 2020.

[32] H. Zhao, B. Li, Z. Li, P. Richtárik, and Y. Chi, "BEER: Fast $O(1/t)$ rate for decentralized nonconvex optimization with communication compression," in *Proc. NeurIPS*, 2022.

[33] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. ICLR*, 2017.

[34] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. NeurIPS*, 2017.

[35] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. NeurIPS*, 2018.

[36] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *Proc. IEEE INFOCOM*, 2005.

[37] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *Proc. IEEE CDC*, 2012.

[38] S. Shi, X. Chu, and B. Li, "MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms," in *Proc. IEEE INFOCOM*, 2019.

[39] N. Eshraghi and B. Liang, "Distributed online optimization over a heterogeneous network with any-batch mirror descent," in *Proc. ICML*, 2020.

[40] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. ICLR*, 2019.

[41] A. Dieuleveut and K. K. Patel, "Communication trade-offs for Local-SGD with large step size," in *Proc. NeurIPS*, 2019.

[42] X. Zhou, W. Liang, K. I.-K. Wang, Z. Yan, L. T. Yang, W. Wei, J. Ma, and Q. Jin, "Decentralized P2P federated learning for privacy-preserving and resilient mobile robotic systems," *IEEE Wireless Communications*, vol. 30, no. 2, pp. 82–89, 2023.

[43] Y. Qu, H. Dai, Y. Zhuang, J. Chen, C. Dong, F. Wu, and S. Guo, "Decentralized federated learning for UAV networks: Architecture, challenges, and opportunities," *IEEE Network*, vol. 35, no. 6, pp. 156–162, 2021.

[44] J. Posner, L. Tseng, M. Aloqaily, and Y. Jararweh, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Network*, vol. 35, no. 2, pp. 152–159, 2021.

[45] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[46] 3GPP, "Study on 5G system support for AI/ML-based services," V18.0.0, Tech. Rep. TR 23.700-80, 2022.

[47] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?" in *Proc. AISTATS*, 2020.

[48] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms," *Journal of Machine Learning Research*, vol. 22, no. 213, pp. 1–50, 2021.

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

[50] A. L. Peressini, F. E. Sullivan, and J. J. Uhl, *The mathematics of nonlinear programming*. Springer, 1988.

[51] K. Srivastava, A. Nedić, and D. Stipanović, "Distributed Bregman-distance algorithms for min-max optimization," in *Agent-Based Optimization*, 2013, pp. 143–174.

[52] M. Rolinek and G. Martius, "L4: Practical loss-based stepsize adaptation for deep learning," in *Proc. NeurIPS*, 2018.

[53] K. Nar and S. Sastry, "Step size matters in deep learning," in *Proc. NeurIPS*, 2018.

[54] C. Daniel, J. Taylor, and S. Nowozin, "Learning step size controllers for robust neural network training," in *Proc. AAAI*, 2016.

[55] B. Or, B. Z. Bobrovsky, and I. Klein, "Kalman filtering with adaptive step size using a covariance-based criterion," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–10, 2021.

[56] J. P. Crouzeix and J. A. Ferland, "Algorithms for generalized fractional programming," *Mathematical Programming*, vol. 52, no. 1, pp. 191–207, 1991.

[57] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[58] "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/.

[59] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[61] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[62] L. Xiao, S. Boyd, and S. J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[63] S. Boyd, P. Diaconis, P. Parrilo, and L. Xiao, "Fastest mixing Markov chain on graphs with symmetries," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 792–819, 2009.