# Joint Consensus Matrix Design and Resource Allocation for Decentralized Learning

Jingrong Wang*, Ben Liang*, Zhongwen Zhu†, Emmanuel Thepie Fapi†, Hardik Dalal†
*Department of Electrical and Computer Engineering, University of Toronto, Canada
†Ericsson Global AI Accelerator Montréal, Canada

***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 preserves the training convergence rate while enforcing communication graph sparsity and avoiding selecting poor communication links. Numerical study with real-world machine learning data demonstrates the efficacy of the proposed solution and its performance advantage over state-of-the-art algorithms.**

## I. Introduction

Large-scale machine learning (ML) often requires distributed storage and computing. While most well-known distributed ML algorithms and systems are built in a centralized fashion (e.g., with a dedicated parameter server) [1]–[3], 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 [4]. 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 [4]. Other variations of FDLA have also been proposed in the literature [5]–[13]. 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 weight [14].

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 [15]. 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 ring, has been investigated to reduce the communication complexity [15]–[19]. 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 [20]–[25], or finding an optimal network topology that minimizes the communication cost subject to a prescribed convergence rate [4], [26], or a connected graph [27]. 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 paper, we aim to accelerate the training process of decentralized ML via joint sparse consensus weight matrix design and communication resource allocation. We propose a novel algorithm named Communication-Efficient Network Topology (CENT), which reduces 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:

This work has been funded in part by Ericsson and by Mitacs.

ISBN 978-3-903176-48-5©2022 IFIP

- 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 in order 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 convergent in decentralized ML. Experimental results on decentralized training of neural networks further demonstrate the performance advantage of CENT over state-of-the-art algorithms, in terms of significantly faster wall-clock training time.

## II. Related Work

### A. Consensus Weight Matrix Design without Considering Communication

To find the optimal consensus weight matrix that leads to the *fastest* convergence rate in terms of the training iterations, Xiao and Boyd [4] proposed FDLA that minimizes the spectral norm of the consensus weight matrix. Further studies on the variations of FDLA have been investigated [5]–[13] Some heuristics to construct the consensus weight matrix have also been proposed to guarantee the convergence of decentralized ML. A naive idea 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 weights [14]. A common disadvantage of the above methods is that they use all physical links of a given underlying network.

### B. 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 [15], [16], path [17], regular expander graphs [18], [19]. However, such sparse graphs are sub-optimal in general.

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

*1) Maximizing the convergence rate subject to limited communication budget:* Dai and Mesbahi [20] 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. [21] and Kempton et al. [22] 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. [23] maximized the algebraic connectivity subject to constraints on the number of workers and communication links. Similarly, Gusrialdi et al. [24] proposed to remove a prescribed number of links such that the largest eigenvalue of the adjacency matrix is minimized. Meng et al. [25] solved the link selection problem with reinforcement learning subject to communication resource and energy consumption constraints.

*2) Minimizing the communication cost subject to a required convergence rate:* Given the underlying connected graph, Xiao and Boyd [4] 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 [26] minimized the number of communication links subject to the constraint that the algebraic connectivity is larger than a prescribed positive value. Marfoq et al. [27] 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.

### C. 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 [28]–[30], gossip and push-sum algorithms [31], [32], and efficient scheduling [33]–[36]. Most of these techniques are orthogonal to our work and can be combined with the proposed solution.

## III. 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.

### A. Decentralized ML

As shown in Fig. 1, we consider a network of workers $\mathcal{N} = \{1, 2, ..., N\}$ that cooperatively train a shared model.

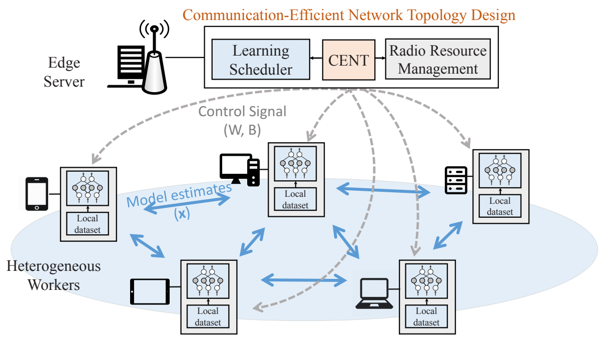

Fig. 1. CENT for decentralized ML.

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 $F_i(\mathbf{x}) = \sum_{\pi \in \xi_i} f(\mathbf{x}; \pi)$ denote the local training loss function of worker $i$ with model parameter $\mathbf{x}$, which is the sum of training losses on the set of local data samples $\xi_i$. 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.

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

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

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 [3].

## B. Communication Model

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 workers are connected to the edge server via an access network, e.g., LTE or 5G access. To facilitate local communication, the workers use orthogonal channels to communicate with neighbors. The edge server communicates with the workers via a dedicated control channel.

Let $B_{i,j}$ denote the bandwidth allocated to the link from worker $i$ to worker $j$, and let $B = [B_{i,j}]$. The communication latency from $i$ to $j$ is

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

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

$$L_{i,j}(B_{i,j}) = l^{\mathrm{P}}_i + l^{\mathrm{C}}_{i,j}(B_{i,j}), \tag{5}$$

where $l^{\mathrm{P}}_i$ denotes the processing latency of worker $i$ for gradient calculation. The latency in each training latency 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 $l^{\mathrm{P}}_i, \forall i$, and channel information $\eta_{i,j}, \forall i, j$, are known and constant over the training iterations.

## C. 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 $T_\epsilon$ denote the number of training iterations required to approximate the ideal training model by a desired error $\epsilon$. It is known that $T_\epsilon$ is inversely proportional to $1 - \rho(W)$ [37], i.e.,

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

Smaller $\rho(W)$ suggests faster network consensus. We note that $\rho(W) < 1$ guarantees the convergence of decentralized ML.

The wall-clock time for training $T_\epsilon$ 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 $\bar{B}$ is the resource budget and

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

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 doubly-stochastic matrix, since using symmetric weights leads to only small decrease in the convergence rate but substantial reduction in computation [4]. 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.

## IV. 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.

### *A. 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., $||A^{(k)}||_1$, and *hypothetically* allocate equal resource to each link, i.e., setting $B^{(k)}_{i,j} = \frac{\bar{B}}{||A^{(k)}||_1}$, 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:

$$L^{(k)}_{i,j} = \begin{cases} L_{i,j}(B^{(k)}_{i,j}), & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases} \tag{15}$$

**Remark 2.** 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 differential 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 $||L^{(k)} \otimes W||_1$ at step $k$, which will be updated over time. We solve the following optimization problem:

$$\begin{aligned} \min_{W} \quad & \lambda^{(k)}\rho(W) + ||L^{(k)} \otimes W||_1, \\ \text{s.t.} \quad & W \in S_{A^{(k)}}, \\ & (12) - (13), \end{aligned} \tag{16}$$

where $\otimes$ denotes the Hadamard product of matrices and $||L^{(k)} \otimes W||_1 = \sum_{i,j\in\mathcal{N}} |L_{i,j}W_{i,j}|$. Since $W \in S_{A^{(k)}}$, we further have $||L^{(k)} \otimes W||_1 = \sum_{\{(i,j)|A^{(k)}_{i,j}=1\}} |L_{i,j}W_{i,j}|$. Placing this $L_1$ 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 $L^{(k)}_{i,j}$. We note that constraint (11) is not considered in problem (16) but will be naturally guaranteed by our design, as shown in the next section.

Problem (16) can be efficiently solved through semidefinite programming as follows:

$$\min_{W,s_1,s_2} \quad \lambda^{(k)}s_1 + s_2,$$

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

where $\preceq$ denotes matrix inequality. Let $W^{(k)} = [W^{(k)}_{i,j}]$ be the solution.

Then, the adjacency matrix is updated by replacing nonzero elements of the weight matrix with ones, i.e.,

$$A^{(k+1)}_{i,j} = \begin{cases} \mathbb{1}_{\{W^{(k)}_{i,j} \neq 0\}}, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases} \tag{17}$$

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 $\Delta_\lambda$ 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 $\rho(W^{(K)}) < 1$, i.e., constraint (11) is satisfied. 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

$$\begin{aligned} \min_{W} \quad & \rho(W), \\ \text{s.t.} \quad & W \in S_{A^{(K)}}, \\ & (12) - (13), \end{aligned} \tag{18}$$

and the Min-Max-RA problem

$$\begin{aligned} \min_{B} \quad & \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}(B_{i,j}) \mathbb{1}_{\{A^{(K)}_{i,j} \neq 0\}} \right\}, \\ \text{s.t.} \quad & (9) - (10). \end{aligned} \tag{19}$$

Both are convex problems with known solutions, e.g., FDLA [4] and the primal-dual Lagrangian approach [38].

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.

## B. Convergence Analysis

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. To show the convergence of CENT, 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 $\rho(\hat{W}) < 1$.

**Algorithm 1** Communication-Efficient Network Topology (CENT)

**Input:** Number of rounds $K$, set of workers $\mathcal{N}$, physical network topology $A$, global network conditions $l^{\text{P}}_i, \eta_{i,j}, \forall i, j$, bandwidth budget $\bar{B}$, and step size $\Delta_\lambda > 0$.

**Output:** consensus weight matrix $\hat{W}$, bandwidth allocation $\hat{B}$.

```
1:  $\lambda^{(0)} \leftarrow 0, A^{(0)} \leftarrow A$;
2:  for $k = 0, ..., K-1$ do
3:      $A^{(k)} \leftarrow A$ if $A^{(k)} = \mathbf{0}$;
4:      $B^{(k)}_{i,j} \leftarrow \frac{\bar{B}}{||A^{(k)}||_1}, \forall (i,j)$;
                                    ▷ Equal allocation among edges
5:      Compute matrix $L^{(k)}$ with (15);
                           ▷ Capture the inherent goodness of the links
6:      Compute $W^{(k)}$ with (16);
                              ▷ Enforce sparsity and avoid bad links
7:      Update the adjacency matrix $A^{(k+1)}$ with (17);
                                       ▷ Update adjacency matrix
8:      if $A^{(k+1)} = A^{(k)}$ 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 $\hat{W}, \hat{B}_K$
```

Let $\mathcal{W}^{(k)} = \{W | W\mathbf{1} = \mathbf{1}, W = W^\top, W \in S_{A^{(k)}}\}, \forall k$. This is the feasible set at step $k$ of CENT. We partition $\mathcal{W}^{(k)}$ into two subsets, i.e.,

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

Since for any connected graph $\mathcal{G}$, we can always design some $W$ such that $\rho(W) < 1$ by treating each link equally, e.g., a constant edge weight based on the Laplacian matrix of the graph [4], we have the following lemma. The proof details are omitted due to the page limit.

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

Furthermore, we make the following observation:

**Lemma 2.** *For any $W \in \mathcal{W}^{(k)}, \forall k$, we have $||L^{(k)} \otimes W||_1 = 0$ if and only if $W$ is the identity matrix $I$.*

*Proof.* If $W = I$, we have $||L^{(k)} \otimes W||_1 = \sum_{i,j} |L^{(k)}_{i,j} W_{i,j}| = \sum_{i=j} |0 \cdot W_{i,j}| + \sum_{i \neq j} |L^{(k)}_{i,j} \cdot 0| = 0$.

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

$$||L^{(k)} \otimes W||_1 = \sum_{\{(i,j)|A^{(k)}_{i,j}=1\}} |L^{(k)}_{i,j} W_{i,j}| \overset{(a)}{>} 0,$$

where $(a)$ holds since $L^{(k)}_{i,j} \neq 0$ for all $i$ and $j$ such that $A^{(k)}_{i,j} = 1$. Therefore, if $||L^{(k)} \otimes W||_1 = 0$, we must have

$W_{i,j} = 0$ for all $i$ and $j$ such that $A^{(k)}_{i,j} = 1$. Since $W \in \mathcal{W}^{(k)}$, we further have $W = I$. $\square$

To facilitate the rest of our analysis, we define

$$\lambda_0 = \min_{W \in \mathcal{W}_2^{(0)}} \frac{||L^{(0)} \otimes W||_1}{1 - \rho(W)}.$$

**Lemma 3.** $\lambda_0$ *is positive and finite.*

*Proof.* Since $\rho(I) = 1$, we know that $I \notin \mathcal{W}_2^{(0)}$. Then, from Lemma 2, we know that for any $W \in \mathcal{W}_2^{(0)}$, $||L^{(0)} \otimes W||_1 > 0$. Further combining the fact that for any $W \in \mathcal{W}_2^{(0)}$, $1 - \rho(W) > 0$, we have $\lambda_0 > 0$. Finally, $\lambda_0$ is finite since $1 - \rho(W) \neq 0$. $\square$

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 4.** *For $0 \leq k < k_0$, the identity matrix $I$ is the unique minimizer to problem (16), so $\rho(W^{(k)}) = 1$. For $k = k_0$, $\rho(W^{(k)}) \leq 1$. For $k > k_0$, $\rho(W^{(k)}) < 1$.*

*Proof.* Let $f^{(k)}(W) = \lambda^{(k)}\rho(W) + ||L^{(k)} \otimes W||_1$.

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

$$\begin{aligned} & f^{(0)}(I) - f^{(0)}(W) \\ &= \lambda^{(0)} - (\lambda^{(0)}\rho(W) + ||L^{(0)} \otimes W||_1) \\ &= \lambda^{(0)}(1 - \rho(W)) - ||L^{(0)} \otimes W||_1. \end{aligned}$$

If $W \in \mathcal{W}_1^{(0)} \backslash I$, from Lemma 2, we know that $||L^{(0)} \otimes W||_1 > 0$. 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

$$f^{(0)}(I) - f^{(0)}(W) < \lambda_0(1 - \rho(W)) - ||L^{(0)} \otimes W||_1.$$

Since by definition $\lambda_0 \leq \frac{||L^{(0)} \otimes W||_1}{1-\rho(W)}, \forall W \in \mathcal{W}_2^{(0)}$, we have $\lambda_0(1 - \rho(W)) - ||L^{(0)} \otimes W||_1 \leq 0$. 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 $\mathcal{W}_2^{(k_0)}$. Let $\bar{W} \in \mathcal{W}_2^{(0)}$ denote the consensus weight matrix such that

$$\frac{||L^{(0)} \otimes \bar{W}||_1}{1 - \rho(\bar{W})} = \lambda_0.$$

Since $\mathcal{W}^{(k_0)} = \mathcal{W}^{(0)}$, we have $\mathcal{W}_2^{(k_0)} = \mathcal{W}_2^{(0)}$ and thus $\bar{W} \in \mathcal{W}_2^{(k_0)}$. Since $\rho(\bar{W}) < 1, L^{(k_0)} = L^{(0)}$, and $\lambda^{(k_0)} = \lambda_0$, we have

$$\begin{aligned} & f^{(k_0)}(I) - f^{(k_0)}(\bar{W}) \\ &= \lambda_0(1 - \rho(\bar{W})) - ||L^{(0)} \otimes \bar{W}||_1 = 0. \end{aligned}$$

We conclude that $\bar{W}$ is no worse than any solution in $\mathcal{W}_1^{(k_0)}$, so a minimizer must exist in $\mathcal{W}_2^{(k_0)}$. Therefore, $\rho(W^{(k_0)}) \leq 1$.

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 $\mathcal{W}^{(k_0+1)}$. Let $\bar{W}$ be as defined above. In this case, we also have $\bar{W} \in \mathcal{W}_2^{(k_0+1)}$. We observe that

$$\begin{aligned} & f^{(k_0+1)}(I) - f^{(k_0+1)}(\bar{W}) \\ &= \lambda^{(k_0+1)}(1 - \rho(\bar{W})) - ||L^{(k_0+1)} \otimes \bar{W}||_1 \\ &= \lambda^{(k_0+1)}(1 - \rho(\bar{W})) - ||L^{(0)} \otimes \bar{W}||_1 \\ &> \lambda_0(1 - \rho(\bar{W})) - ||L^{(0)} \otimes \bar{W}||_1 \\ &= 0. \end{aligned}$$

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

ii) If $W^{(k_0)} \neq I$, from the analysis of the case $k = k_0$, we must have $W^{(k_0)} \in \mathcal{W}_2^{(k_0)}$. 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 $A^{(k_0+1)} \neq A^{(k_0)}$, then we have $\lambda^{(k_0+1)} = \lambda^{(k_0)}$ and $\mathcal{W}^{(k_0+1)} \subset \mathcal{W}^{(k_0)}$. Therefore, $||A^{(k_0+1)}||_1 < ||A^{(k_0)}||_1$. By equally allocating resource to each link, we have $B_{i,j}^{(k_0+1)} > B_{i,j}^{(k_0)}, \forall i, j$ (line 4 of Algorithm 1). Since $L_{i,j}$ is strictly decreasing with $B_{i,j}$, we further have $L_{i,j}^{(k_0)} > L_{i,j}^{(k_0+1)}, \forall i, j$. This implies that

$$\begin{aligned} & f^{(k_0+1)}(I) - f^{(k_0+1)}(W^{(k_0)}) \\ &= \lambda^{(k_0+1)}(1 - \rho(W^{(k_0)})) - ||L^{(k_0+1)} \otimes W^{(k_0)}||_1 \\ &= \lambda^{(k_0)}(1 - \rho(W^{(k_0)})) - ||L^{(k_0+1)} \otimes W^{(k_0)}||_1 \\ &> \lambda^{(k_0)}(1 - \rho(W^{(k_0)})) - ||L^{(k_0)} \otimes W^{(k_0)}||_1. \end{aligned}$$

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 (17), 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 $\rho(W^{(k)}) < 1$. $\square$

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

**Lemma 5.** $\rho(W^{(k)}) \geq \rho(W^{(k+1)}), \forall k > k_0$.

*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 (17), we also have $W^{(k)} \in \mathcal{W}^{(k+1)}$. Let $\eta^{(k)} = ||L^{(k)} \otimes W^{(k)}||_1$. Since $W^{(k)}$ and $W^{(k+1)}$ are minimizers of problem (16) in steps $k$ and $k+1$, respectively, we have

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

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 $||L^{(k+1)} \otimes W||_1$ and $||L^{(k)} \otimes W||_1$. As explained in the proof of Lemma 4, 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 4, 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 (16) in step $k+1$ can be equivalently written as

$$\min_{W \in \mathcal{W}^{(k+1)}} \frac{\lambda^{(k)}}{\psi^{(k)}(W)}\rho(W) + ||L^{(k)} \otimes W||_1.$$

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 6.** *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 5, 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 4, $L_{i,j}^{(k)} \geq L_{i,j}^{(k+1)}, \forall i,j,k$. Since $\mathcal{W}^{(k+1)} \subseteq \mathcal{W}^{(k)}$, we have

$$g(W^{(k)}, B^{(k)}) = \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}^{(k)} \mathbb{1}_{\{W_{i,j}^{(k)} \neq 0\}} \right\} \tag{20}$$
$$\geq \max_{i,j \in \mathcal{N}} \left\{ L_{i,j}^{(k+1)} \mathbb{1}_{\{W_{i,j}^{(k+1)} \neq 0\}} \right\} = g(W^{(k+1)}, B^{(k+1)}), \ \forall k.$$

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 7.** *If $K > k_0$, decentralized ML converges.*

*Proof.* From Lemma 4, we have $\rho(W^{(k)}) < 1$ for $k > k_0$. Therefore, $\rho(W^{(K)}) < 1$. By solving problem (18), we further have $\rho(\hat{W}) \leq \rho(W^{(K)})$. We conclude that $\rho(\hat{W}) < 1$. Furthermore, from (19), 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 $A_{i,j}^{(K)} \neq 0$. This means that the communication latency in each training iteration is finite, which concludes the proof. □

## V. Numerical Performance Evaluation

In this section, we evaluate the performance of CENT on decentralized convolutional neural network (CNN) training over a decentralized network. Unless otherwise specified, we place $N = 50$ workers randomly in a 100 m × 100 m area. We generate the network topology in a similar way as [4], with 200 edges among the workers. Two workers are connected by an edge if the distance between them is within a specified threshold, and we increase the threshold until the total number of edges reaches 200. This procedure is repeated until a connected graph is generated. The total bandwidth budget $\bar{B}$ is 20 MHz. The transmission power of each worker is 1 W. The channel power gain $h_{i,j}(d_{i,j}) = \gamma_0(\frac{\hat{d}}{d_{i,j}})^4$, where $\gamma_0 = -40$ dB is the path loss at the reference distance $\hat{d} = 1$ m [39].

The 50 workers collectively train a CNN for handwritten-digit recognition on the MNIST dataset [40]. 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 training set is evenly divided among the workers at random. We consider LeNet [41] 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. The learning rate is set to 0.001. The minibatch size is set to 256. In each training iteration, each worker processes a minibatch of the local dataset and then communicates with the adjacent neighbors.

To represent the heterogeneous computation capacities among workers, we model the processing time of the workers as $l_i^{\mathrm{P}} = \bar{l}_i^{\mathrm{P}}((1-v) + v\phi), \forall i$, where $\phi$ follows the uniform distribution over $[0, 2]$ and $v = 0.8$ [18], and the average processing time $\bar{l}_i^{\mathrm{P}} = 5.231$ s is measured from processing a minibatch of data samples on a 2.9 GHz Intel Core i5 processor and 8 GB of memory.

We compare the performance of CENT with that of the following benchmarks:

- **FDLA** [4]: 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** [5]: Assign all edges the same weight based on the maximum degree of the graph.
- **Metropolis** [6]: Assign each edge a weight based on the maximum degree of its two adjacent workers.

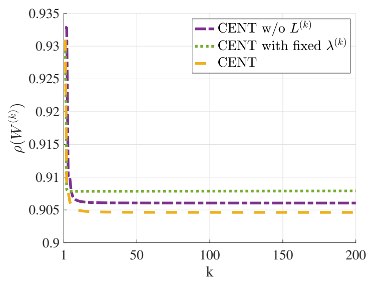

Fig. 2. Convergence factor $\rho(W)$ vs. steps.

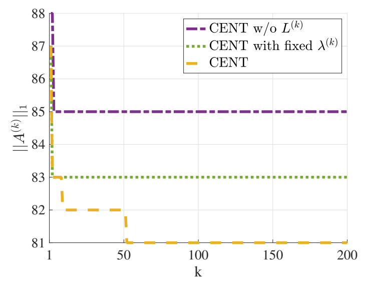

Fig. 3. Number of selected links vs. steps.

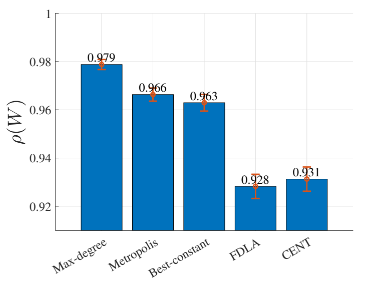

Fig. 4. Convergence factor $\rho(W)$.

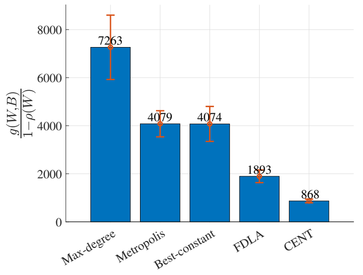

Fig. 5. Training time objective $\frac{g(W,B)}{1-\rho(W)}$.

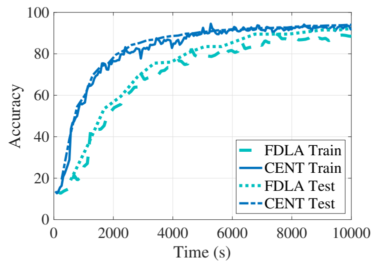

Fig. 6. Accuracy vs. wall-clock time.

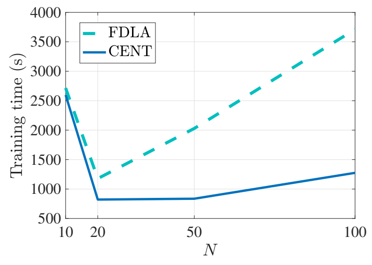

Fig. 7. Wall-clock training time over $N$.

- **Best-constant** [7]: Assign all edges the same optimal constant weight based on the eigenvalues of the Laplacian matrix of the graph.

We do not compare with the methods in [15]–[27], since they are incompatible with the wall-clock training time and thus do not solve our problem. For the above four benchmarks, the corresponding bandwidth allocation is calculated by solving Min-Max-RA subject to the physical network topology. For CENT, by default, $\Delta_\lambda$ is set to 2000.

### *A. Impact of $L^{(k)}$ and $\lambda^{(k)}$ in CENT*

Figs. 2 and 3 illustrate the impact of using $L^{(k)}$ and $\lambda^{(k)}$ in the design of CENT. We compare CENT with two naive variants: CENT w/o $L^{(k)}$ refers to eliminating $L^{(k)}$ in Algorithm 1, i.e., $L^{(k)}$ is substituted by an all-ones matrix; CENT with fixed $\lambda^{(k)}$ refers to fixing $\lambda^{(k)} = 2000$ over all $K$ steps. We observe that both variants result in an increase in the training time when compared with CENT. 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 $\rho(W)$ 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.

### *B. Convergence Factor $\rho(W)$ and Wall-Clock Training Time*

Figs. 4 and 5 show the convergence factor $\rho(W)$ and upper bound of wall-clock training time with 95% confidence intervals, over 100 realizations of the physical network topology. CENT requires significantly shorter wall-clock training time than the other methods, while retaining $\rho(W)$ as FDLA. Note that FDLA by design gives the minimum possible $\rho(W)$. Furthermore, when compared with the benchmarks, CENT reduces the wall-clock training time by 88%, 78.7%, 78.6%, and 54%.

### *C. Training and Test Accuracy over Wall-clock Time*

Fig. 6 shows the training and test accuracy over wall-clock time. We observe that even though FDLA optimizes the convergence factor, it does not lead to the fastest convergence speed in terms of the wall-clock time. With fewer edges selected for communications and reduced latency in each training iteration, CENT spends less time achieving the same level of training accuracy.

### *D. Impact of the Network Scale*

We investigate the performance of CENT over different network sizes $N$. We generate the network topology in a similar way as mentioned above, with $\frac{200N}{50}$ edges among $N$ workers. The minibatch size is set to $\frac{256\times 50}{N}$. Fig. 7 shows the wall-clock training time to achieve the same 60% training accuracy with the same 20 MHz bandwidth. For both FDLA and CENT, the wall-clock training time has a significant drop when $N$ grows from 10 to 20 due to the reduced computational workload at each worker. However, when $N$ is further increased, the wall-clock training time increases considerably due to the scarcity of the network bandwidth. With more workers sharing limited bandwidth, the communication latency in each training iteration increases and gradually overweighs the computation cost. Thus, we observe a tradeoff between computation and communication under different network scales. With limited bandwidth, blindly increasing the number of workers and the edges among them aggravates network congestion, canceling out the benefits of parallel computing. However, when compared with FDLA, CENT excels in robustness to accelerate the wall-clock training time with efficient sparse graph design and bandwidth allocation.

## VI. Conclusion and Future Work

In this paper, we have explored communication design for decentralized ML with bandwidth constraints and heterogeneous workers. By jointly considering the consensus matrix and bandwidth allocation, we have proposed a novel algorithm, termed CENT, aiming to speed up the training process by eliminating unnecessary communication links for more efficient bandwidth allocation. Numerical studies on real-world decentralized CNN training show the efficacy of the proposed solution, in terms of significantly faster wall-clock training time. For future work, efficient consensus weight matrix design for dynamic networks can be explored when further considering worker mobility and heterogeneous resources.

## References

[1] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. NeurIPS*, vol. 27, 2014, pp. 19–27.

[2] 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, pp. 1273–1282.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 116–120, 2006.

[9] A. Ghosh and S. Boyd, "Upper bounds on algebraic connectivity via convex optimization," *Linear algebra and its applications*, vol. 418, no. 2-3, pp. 693–707, 2006.

[10] M. C. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Proc. IEEE CDC*, 2006, pp. 3628–3633.

[11] M. Cao and C. W. Wu, "Topology design for fast convergence of network consensus algorithms," in *Proc. IEEE ISCAS*, 2007, pp. 1029–1032.

[12] P. Mukherjee, M. Santilli, A. Gasparri, and R. K. Williams, "Optimal topology selection for stable coordination of asymmetrically interacting multi-robot systems," in *Proc. IEEE ICRA*, 2020, pp. 6668–6674.

[13] M. Jameel, S. Jawed, and L. Schmidt-Thieme, "Optimal topology search for fast model averaging in decentralized parallel SGD," in *Proc. PAKDD*, 2020, pp. 894–905.

[14] 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.

[15] 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, pp. 5336–5346.

[16] 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, pp. 3022–3026.

[17] 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.

[18] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *Proc. IEEE INFOCOM*, 2019, pp. 2350–2358.

[19] Y. Hua, K. Miller, A. L. Bertozzi, C. Qian, and B. Wang, "Efficient and reliable overlay networks for decentralized federated learning," *arXiv preprint arXiv:2112.15486*, 2021.

[20] R. Dai and M. Mesbahi, "Optimal topology design for dynamic networks," in *Proc. IEEE CDC-ECC*, 2011, pp. 1280–1285.

[21] 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.

[22] L. Kempton, G. Herrmann, and M. di Bernardo, "Adaptive weight selection for optimal consensus performance," in *Proc. IEEE CDC*, 2014, pp. 2234–2239.

[23] 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.

[24] 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.

[25] 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.

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

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

[28] 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.

[29] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. NeurIPS*, vol. 30, 2017, pp. 1709–1720.

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

[31] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *Proc. IEEE INFOCOM*, vol. 3, 2005, pp. 1653–1664.

[32] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *Proc. IEEE CDC*, 2012, pp. 5453–5458.

[33] S. Shi, X. Chu, and B. Li, "MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms," in *Proc. IEEE INFOCOM*, 2019, pp. 172–180.

[34] N. Eshraghi and B. Liang, "Distributed online optimization over a heterogeneous network with any-batch mirror descent," in *Proc. ICML*, 2020, pp. 2933–2942.

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

[36] A. Dieuleveut and K. K. Patel, "Communication trade-offs for Local-SGD with large step size," in *Proc. NeurIPS*, vol. 32, 2019, pp. 13 601–13 612.

[37] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?" in *Proc. AISTATS*, vol. 108, 2020, pp. 2348–2358.

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

[39] 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.

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

[41] 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.