Laplacian Matrix Sampling for Communication-efficient Decentralized Learning

Cho-Chun Chiu, *Student Member, IEEE,* Xusheng Zhang, *Student Member, IEEE,*
Ting He, *Senior Member, IEEE,* Shiqiang Wang, *Member, IEEE,* and Ananthram Swami, *Fellow, IEEE*

***Abstract*—We consider the problem of training a given machine learning model by decentralized parallel stochastic gradient descent over training data distributed across multiple nodes, which arises in many application scenarios. Although extensive studies have been conducted on improving the communication efficiency by optimizing what to communicate between nodes (e.g., model compression) and how often to communicate, recent studies have shown that it is also important to customize the communication patterns between each pair of nodes, which is the focus of this work. To this end, we propose a framework and efficient algorithms to design the communication patterns through Laplacian matrix sampling (LMS), which governs not only which nodes should communicate with each other but also what weights the communicated parameters should carry during parameter aggregation. Our framework is designed to minimize the total cost incurred until convergence based on any given cost model that is additive over iterations, with focus on minimizing the communication cost. Besides achieving a theoretically guaranteed performance in the special case of additive homogeneous communication costs, our solution also achieves superior performance under a variety of network settings and cost models in experiments based on real datasets and topologies, saving 24–50% of the cost compared to the state-of-the-art design without compromising the quality of the trained model.**

***Index Terms*—Decentralized learning, D-PSGD, convergence analysis, Laplacian matrix sampling, communication cost.**

I. Introduction

Learning from decentralized data, first introduced by [1], is a machine learning paradigm in which multiple nodes collaboratively learn a shared machine learning model over the union of their local data without directly sharing the data [2]. Instead, the nodes exchange updates to the shared model or updated local models, either through a central parameter server (as in federated learning [1]) or through peer-to-peer links between neighboring nodes (as in decentralized learning [2]), which are then aggregated to update the shared model in an iterative manner. Due to its potential in reducing communication cost and preserving user privacy, this learning paradigm has found many applications, including both mobile applications (e.g., for Google Keyboard [3] and Google Assistant [4]) and desktop applications (e.g., for Chrome [5] and Brave [6]).

In all these use cases, although the raw data stay local, the nodes still need to communicate repeatedly to update the shared model, which incurs a nontrivial communication cost, e.g., in terms of communication time, bandwidth consumption, and energy consumption. In many application scenarios, the communication cost dominates the total operation cost [1]. This has motivated a number of ideas to reduce the communication cost, mainly by (i) reducing the cost of each communication and (ii) reducing the total number of communications. The former is usually achieved through compressing the communicated models [7], [8], [9], and the latter is usually achieved through tuning the frequency of communications [1], [10], [11], [12]. The two approaches address orthogonal aspects of the communication cost and thus can be applied jointly. In this work, we focus on the second approach.

We consider learning in a (fully) decentralized setting. In contrast to the original setting in [1] where the communication topology is a star with the parameter server at the center, nodes in the decentralized setting can communicate according to an arbitrary topology (as long as it is physically feasible). Such a setting can avoid a single point of failure and balance the traffic across nodes, which can significantly reduce the communication complexity at the busiest node without increasing the computational complexity [13].

Our solution is inspired by a recent discovery in [14]: *not all the links are equally important for convergence*. Hence, instead of activating communications over all the links at the same frequency, activating different links with different frequencies can further reduce the communication cost without hurting convergence. While [14] gave a solution, called MATCHA, to minimize the communication time by activating certain sets of links (matchings) with designed probabilities, it left open many important questions, such as: (i) Will this idea be useful in reducing other types of costs (e.g., bandwidth/energy consumption)? (ii) Is there a better design that can further improve the tradeoff between communication cost and convergence? In this work, we answer both questions affirmatively by developing a general framework and the corresponding algorithms to design the communication patterns during decentralized learning that can further improve the cost-convergence tradeoff while supporting a broader set of cost models.

A. Related Work

Decentralized learning. Initially proposed under a parameter server architecture [1], learning from decentralized data was later extended to a fully decentralized architecture [13], where a training algorithm called Decentralized Parallel Stochastic Gradient Descent (D-PSGD) was shown to achieve the same computational complexity but a lower communication complexity than

Chiu, Zhang, and He are with the Pennsylvania State University, University Park, PA 16802 USA. Email: {cuc496, xzz5349, tzh58}@psu.edu. Wang is with IBM. Email: wangshiq@us.ibm.com. Swami is with DEVCOM Army Research Laboratory. Email: a.swami@ieee.org.

training via a central server. Since then a number of improvements have been developed, e.g., [15] improved the robustness to data variance by adding variance reduction, [16] proposed an asynchronous version to reduce the idle time due to synchronization barriers, and [17] provided a lower bound on the iteration complexity and an algorithm that achieves the bound. These works focused on the number of iterations.

Communication cost reduction. Communication cost is an important consideration in decentralized learning. One line of work tried to reduce the amount of data per communication through model compression, e.g., [7], [8], [9]. Another line of works tuned the frequency of communications to balance communication cost and convergence rate, e.g., [10], [11], [12]. Recent works [18], [19] started to combine model compression and infrequent communications to improve the communication efficiency of decentralized learning, which achieved the same convergence rate of $O(1/\sqrt{mK})$ as the vanilla D-PSGD (where m is the number of nodes and K the number of iterations). Instead of either activating all the links for communications or activating none, it has been recognized that better tradeoffs can be achieved by activating subsets of links. To this end, [18], [19] proposed an event-triggered mechanism where a node sends (a compressed version of) its local model to neighbors only if the model has changed sufficiently, and [14], [20] proposed to activate subsets of links that form matchings with predetermined probabilities. Although these solutions all achieved the same asymptotic convergence rate as the vanilla D-PSGD, the number of iterations to achieve a given convergence criterion can vary a lot based on the design of hyperparameters. In this regard, our work focuses on the design of the mixing matrix, which is closest to [14], [20] that proposed a framework called MATCHA to design the probabilities of activating matchings. Our work significantly improves [14], [20] in multiple ways: (i) instead of only considering communication time as the cost measure, we adopt a more general cost model that can represent other important cost measures such as bandwidth/energy consumption; (ii) instead of giving the same weight to all the received models during model aggregation, we consider a larger solution space by allowing different weights for models from different nodes; (iii) instead of using a heuristic design objective (algebraic connectivity), we directly optimize an objective with provable relationship to the convergence rate of decentralized learning.

Mixing matrix design. Decentralized learning is closely related to the classical problem of distributed averaging, where nodes with different initial values need to achieve consensus by taking a weighted average of the values received from their neighbors. The matrix containing these weights is referred to as the *mixing matrix* (a.k.a. consensus/communication/gossip matrix). Multiple solutions were proposed to optimize the mixing matrix for distributed averaging, e.g., [21], [22] designed a mixing matrix with the fastest convergence to ϵ-average via semi-definite programming, and [23], [24] designed a sequence of mixing matrices to achieve exact average in finite time. Decentralized learning differs from distributed averaging in that it interleaves distributed averaging steps with local model updates. In contrast to the extensive studies of decentralized learning under given mixing matrices, few works have addressed the design of mixing matrices in this context. While it was known that denser mixing matrices generally require fewer iterations [25], [14], sparser mixing matrices can reduce the cost per iteration by reducing the time in waiting for stragglers [25] or the time in exchanging model parameters [14]. In this regard, we develop algorithms to design mixing matrices through Laplacian matrix sampling that outperform the state of the art in minimizing the operational cost for decentralized learning.

B. Summary of Contributions

We consider the design of communication patterns for decentralized learning, with the following contributions:

1) We develop a general framework to minimize the total cost for decentralized learning by designing randomized mixing matrices through Laplacian matrix sampling. Based on an existing convergence bound, we formulate the design problem into a bilevel optimization, where the lower level minimizes the number of iterations to achieve a given convergence bound under a given budget per iteration, and the upper level tunes the per-iteration budget to minimize the total cost until convergence.
2) We tackle the computational challenges in solving the formulated optimization by decomposing it into two subproblems: (i) designing the sampling probabilities and (ii) designing the candidate Laplacian matrices. We solve the first subproblem by formulating it as a semi-definite programming (SDP) problem, and develop a suite of algorithms for the second subproblem that includes both heuristics and a graph-sparsification-based algorithm with guaranteed performance in the special case of additive homogeneous communication costs.
3) We evaluate the performance of decentralized learning under the proposed design in comparison to benchmark designs on real datasets and network topologies. Our experiments under a variety of network settings and cost models show that our design can save 24–50% of the cost compared to the state-of-the-art design in [14] while yielding a model of similar or better quality.

Roadmap. Sections II and III provide the background and the description of our design framework. Section IV presents our solution and performance analysis. Section V evaluates our solution in comparison with benchmarks. Section VI concludes the paper. **All the proofs can be found in Appendix.**

II. Background and Problem Formulation

A. Notations

Let $\boldsymbol{a} \in \mathbb{R}^m$ denote a vector and $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ a matrix. We use $\|\boldsymbol{a}\|$ to denote the ℓ-2 norm, $\|\boldsymbol{A}\|$ to denote the spectral norm, and $\|\boldsymbol{A}\|_F$ to denote the Frobenius norm. We use $\mathrm{diag}(\boldsymbol{a})$ to denote a diagonal matrix with the entries in $\boldsymbol{a}$ on the main diagonal, and $\mathrm{diag}(\boldsymbol{A})$ to denote a vector formed by the diagonal entries of $\boldsymbol{A}$. We use $\lambda_i(\boldsymbol{A})$ $(i = 1, \ldots, m)$ to denote the i-th smallest eigenvalue of $\boldsymbol{A}$.

B. Decentralized Learning

Consider a network of m nodes (i.e., learning agents) connected through a connected undirected graph $G = (V, E)$ ($V := \{1, \ldots, m\}$), referred to as the *base topology*. This topology may be determined by the physical connectivity between nodes (when representing a physical network) or the allowed communications (when representing an overlay), in both cases E defines the pairs of nodes that can directly communicate. Each node $i \in V$ has a local, possibly non-convex objective function $F_i(\boldsymbol{x})$ that depends on the parameter vector $\boldsymbol{x}$ and the local dataset $\mathcal{D}_i$. The goal is to find the parameter vector $\boldsymbol{x}$ that minimizes the global objective function $F(\boldsymbol{x})$, defined as

$$F(\boldsymbol{x}) := \frac{1}{m} \sum_{i=1}^{m} F_i(\boldsymbol{x}). \tag{1}$$

For example, the local objective may be to minimize the local loss by defining $F_i(\boldsymbol{x}) := \sum_{p \in \mathcal{D}_i} \ell(\boldsymbol{x}, p)$, where $\ell(\boldsymbol{x}, p)$ is the per-sample loss function, and the corresponding global objective is to minimize the global loss over all the samples since $F(\boldsymbol{x}) = \left(\sum_{p \in \bigcup_{i \in V} \mathcal{D}_i} \ell(\boldsymbol{x}, p) \right) / m$. Our results are independent of the specific definition of $F_i(\boldsymbol{x})$, as long as several commonly-made assumptions hold (see Section III-A).

We consider a canonical decentralized training algorithm called D-PSGD [13], where each node maintains its own parameter vector that is repeatedly updated and averaged with the parameter vectors of its neighbors to converge towards a consensus that minimizes the global objective function. Specifically, let $\boldsymbol{x}_i^{(k)}$ ($k \geq 1$) denote the parameter vector at node i after $k-1$ iterations and $g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)})$ the stochastic gradient computed in iteration k (where $\xi_i^{(k)}$ is the mini-batch sampled by node i). In iteration k, node i updates its parameter vector by

$$\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)} (\boldsymbol{x}_j^{(k)} - \eta g(\boldsymbol{x}_j^{(k)}; \xi_j^{(k)})), \tag{2}$$

where $\boldsymbol{W}^{(k)} = (W_{ij}^{(k)})_{i,j=1}^{m}$ is the $m \times m$ *mixing matrix* in iteration k, and $\eta > 0$ is the learning rate. Each update by (2) includes (i) a *gradient descent step* $\tilde{\boldsymbol{x}}_i^{(k)} := \boldsymbol{x}_i^{(k)} - \eta g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)})$, (ii) a *communication step* to exchange $\tilde{\boldsymbol{x}}_i^{(k)}$ between neighbors, and (iii) a *consensus step* $\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)} \tilde{\boldsymbol{x}}_j^{(k)}$. To be consistent with the base topology, $W_{ij}^{(k)} \neq 0$ only if $(i, j) \in E$. One can swap step (i) and steps (ii–iii), i.e., $\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)} \boldsymbol{x}_j^{(k)} - \eta g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)})$, and the performance analysis will remain the same [13], [14].

We want to improve the communication efficiency of D-PSGD by designing the mixing matrix $\boldsymbol{W}^{(k)}$, which plays an important role in controlling the communication cost, as node j needs to send its parameter vector to node i in iteration k only if $W_{ij}^{(k)} \neq 0$. According to [13], it is desirable to keep the mixing matrix *symmetric with each row/column summing up to one*[1] in order to ensure convergence for D-PSGD. Inspired by this requirement, we design the mixing matrix as

$$\boldsymbol{W}^{(k)} := \boldsymbol{I} - \boldsymbol{L}^{(k)}, \tag{3}$$

where $\boldsymbol{L}^{(k)}$ is the weighted Laplacian matrix [26] of the topology $G^{(k)} = (V, E^{(k)})$ that is *activated* in iteration k, defined as

$$\boldsymbol{L}^{(k)} := \boldsymbol{D}^{(k)} - \boldsymbol{A}^{(k)}. \tag{4}$$

Here, $\boldsymbol{A}^{(k)}$ denotes the weighted adjacency matrix of $G^{(k)}$, where $A_{ij}^{(k)} \geq 0$ for $(i, j) \in E$, $A_{ij}^{(k)} = 0$ for $(i, j) \notin E$, and $A_{ij}^{(k)} = A_{ji}^{(k)}$ for all $i, j \in V$, and $\boldsymbol{D}^{(k)} := \mathrm{diag}(d_1^{(k)}, ..., d_m^{(k)})$ is the weighted degree matrix of $G^{(k)}$, with $d_i^{(k)} := \sum_{j=1}^{m} A_{ij}^{(k)}$. The above construction guarantees that $\boldsymbol{W}^{(k)}$ is symmetric with each row/column summing up to one.

We can write $\boldsymbol{L}^{(k)}$ as a function of the incidence matrix $\boldsymbol{B}$ of the base topology G and a vector of *link weights* $\boldsymbol{\alpha}^{(k)}$. The incidence matrix $\boldsymbol{B}$ is a $|V| \times |E|$ matrix defined as

$$B_{ij} = \begin{cases} +1 & \text{if } s(e_j) = i, \\ -1 & \text{if } t(e_j) = i, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $s(e_j)$ and $t(e_j)$ are the beginning/ending points of link e_j under an arbitrary orientation. The link weight vector $\boldsymbol{\alpha}^{(k)} \in \mathbb{R}^{|E|}$ consists of $\alpha_{(i,j)}^{(k)} := A_{ij}^{(k)}$ for each $(i, j) \in E$. Then the weighted Laplacian matrix $\boldsymbol{L}^{(k)}$ defined in (4) is equal to

$$\boldsymbol{L}^{(k)} = \boldsymbol{B}\, \mathrm{diag}(\boldsymbol{\alpha}^{(k)}) \boldsymbol{B}^T. \tag{6}$$

When $\boldsymbol{\alpha}^{(k)} = \boldsymbol{1}$, we obtain the unweighted Laplacian matrix of the base topology G. Together, (3) and (6) reduce the mixing matrix design problem to a problem of designing the link weights $\boldsymbol{\alpha}^{(k)}$, where *a link* $(i, j) \in E$ *will be activated in iteration* k (i.e., nodes i and j will exchange parameter vectors in iteration k) *if and only if* $\alpha_{(i,j)}^{(k)} \neq 0$.

Remark: In contrast to the previous work [14] that requires all the activated links to have an identical weight, we allow heterogeneous link weights.

C. Cost Model

Based on the update rule (2), the cost incurred at node $i \in V$ during iteration k contains two parts: (i) computation cost for executing the gradient descent and the consensus steps, and (ii) communication cost for communicating the gradient-descended parameter vector (i.e., $\tilde{\boldsymbol{x}}_i^{(k)}$) to the neighbors of node i over activated links (if any). While the computation cost is largely independent of the mixing matrix[2], the communication cost directly depends on it, where the specific form of dependency depends on how the communication cost is measured.

We use $c(\boldsymbol{L}^{(k)})$ to denote the cost in iteration k when the (weighted) Laplacian of the activated topology is $\boldsymbol{L}^{(k)}$, assumed to be additive across iterations. This is a general model that can represent any cost measure as long as (i) the per-iteration cost is determined by the activated topology, and

[1] In [13], the mixing matrix was assumed to be symmetric and *doubly stochastic* with entries constrained to $[0, 1]$, but we find this requirement unnecessary for the convergence bound we use from [14, Theorem 2], which holds as long as the mixing matrix is symmetric with each row/column summing up to one.

[2] As discussed in Section II-E, the type of mixing matrices we adopt requires nodes to always perform local SGD regardless of the activated topology.

(ii) the overall cost over multiple iterations is the sum of the per-iteration costs. Below are some concrete examples:

1) Communication time: Based on the observation that the time to complete the communication step of D-PSGD is typically proportional to the maximal node degree in the activated topology, [14] defined $c(\boldsymbol{L}^{(k)})$ as the number of matchings in the activated topology. This cost measure is proportional to the communication time in an iteration that activates a topology according to $\boldsymbol{L}^{(k)}$, under the assumption that communications on disjoint links can occur in parallel.

2) Energy consumption: In wireless networks, an important cost measure is energy consumption due to communication and computation. This can be modeled by defining $c(\boldsymbol{L}^{(k)}) := \sum_{i=1}^{m} c_i(\boldsymbol{L}^{(k)})$ and defining $c_i(\boldsymbol{L}^{(k)})$ to represent the energy consumption at node i when activating topology $\boldsymbol{L}^{(k)}$. Under unicast communication, a node needs to separately transmit its parameter vector over each activated link incident to it. Let $c_i^a > 0$ denote the energy consumption due to the computation in an iteration at node i and $c_{ij}^b > 0$ denote the energy consumption due to a transmission of the parameter vector from node i to node j. Then

$$c_i(\boldsymbol{L}^{(k)}) := c_i^a + \sum_{j:(i,j)\in E} c_{ij}^b \mathbb{1}(L_{ij}^{(k)} \neq 0) \tag{7}$$

models the per-iteration energy consumption at node i. Under broadcast communication, when a node is incident to multiple activated links (i.e., it needs to share its parameter vector with multiple neighbors), it only needs to broadcast once. Thus, the per-iteration energy consumption at node i becomes

$$c_i(\boldsymbol{L}^{(k)}) := c_i^a + c_i^b \mathbb{1}(\exists (i,j) \in E:\ L_{ij}^{(k)} \neq 0), \tag{8}$$

where $c_i^a > 0$ denotes the computation energy consumption per iteration and c_i^b the communication energy consumption per broadcast, both for node i.

3) Bandwidth consumption: The model of $c(\boldsymbol{L}^{(k)}) := \sum_{i=1}^{m} c_i(\boldsymbol{L}^{(k)})$ with $c_i(\boldsymbol{L}^{(k)})$ defined as in (7) can also model other cost measures of practical significance. For example, consider the case that the nodes participating in learning form an overlay network with topology G, where the existence of an overlay link $(i,j) \in E$ means that nodes i and j can communicate with each other through a connection (e.g., a TCP connection) supported by a path in the underlay network. Then defining c_i^a as 0 and c_{ij}^b as the hop count in the underlay path from node i to node j allows $c(\boldsymbol{L}^{(k)})$ to represent the total bandwidth consumption in the underlay network when activating $\boldsymbol{L}^{(k)}$ (measured by the total number of times the parameter vector is transmitted), as communicating the parameter vector from i to j involves c_{ij}^b transmissions of the parameter vector through the underlay network.

Remark: Note that instead of transmitting the original $\tilde{\boldsymbol{x}}_i^{(k)}$, node i may transmit a compressed version (e.g., [7], [8], [9]), and our solution can be used in combination with such compression schemes to further reduce communication cost, where the effect of the compression can be incorporated into $c(\boldsymbol{L}^{(k)})$.

D. Design Parameters

In the vanilla D-PSGD [13], the entire base topology G is activated in every iteration. While activating all the links is intuitively beneficial for convergence rate as more information is exchanged in each iteration, such a strategy can be inefficient in terms of cost. We seek to design the mixing matrix by designing the Laplacian matrix (that determines the mixing matrix as in (3)) to minimize the total cost of running D-PSGD, through optimizing the following design parameters.

1) Candidate Laplacian Matrices: We want to design a set of candidate Laplacian matrices $\mathcal{L} := \{\boldsymbol{L}_1, \ldots, \boldsymbol{L}_n\}$, which can achieve a diverse range of tradeoffs between the per-iteration cost and the number of iterations required for D-PSGD to converge. Under (6), this boils down to the design of candidate link weights $(\boldsymbol{\alpha}_j)_{j=1}^n$.

2) Sampling Probabilities: As the cost is typically a discontinuous function of the Laplacian matrix (e.g., (7)), the design of Laplacian matrices usually cannot be solved to optimality. To reduce the optimality gap, we will also optimize how to choose from the set of candidate Laplacian matrices by designing their sampling probabilities $\boldsymbol{p} := (p_j)_{j=1}^n$ with $\sum_{i=1}^n p_j = 1$, where p_j denotes the probability of sampling $\boldsymbol{L}_j$. We assume that the sampling is i.i.d. across iterations. If $\boldsymbol{L}_j$ is sampled in iteration k, then $\boldsymbol{W}^{(k)} = \boldsymbol{I} - \boldsymbol{L}_j$. A justification of this randomized approach is given in Section IV-D3.

E. Design Objective

Our goal is to minimize the total cost for running D-PSGD till convergence by designing the candidate Laplacian matrices $\mathcal{L}$ and their sampling probabilities $\boldsymbol{p}$. Although $n := |\mathcal{L}|$ is also a design parameter, our focus in this work is on optimizing $(\mathcal{L}, \boldsymbol{p})$ for a given n. Our experiments have validated that the performance of our proposed design is not sensitive to the value of n (see Section V-C1).

Remark: (i) Although generally the activated topologies may only contain a subset of nodes as in [1], we focus on link activation in this work, as the definition of mixing matrix in (3) implies that nodes will still perform local SGD in iteration k even if $\boldsymbol{L}^{(k)} = \boldsymbol{0}$. Defining the mixing matrix as in (3) allows guaranteed convergence (see Theorem III.1). There has been some work on optimizing node activation in the centralized setting [27]. We leave the design of both link and node activation in the decentralized setting to future work. (ii) The design of $(\mathcal{L}, \boldsymbol{p})$ together with a common random seed can be computed by the central authority when setting up the learning task [2] and shared with the nodes, such that nodes will sample the same Laplacian matrix in each iteration and execute the communication and consensus steps accordingly. (iii) As our design focuses on optimizing the communication patterns, it can be combined with orthogonal techniques such as model compression [7], [8], [9] and runtime gradient tracking [15].

III. Optimization Framework

To rigorously formulate the problem of mixing matrix design, we will quantify the total cost under a given design in order to formulate the problem as an optimization.

A. Convergence Bound

To bound the number of iterations for D-PSGD to converge under a given design of the mixing matrices, we will leverage an existing result from [14].

Define $\boldsymbol{J} := \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ as an $m \times m$ matrix with all entries being $\frac{1}{m}$. Let the mixing matrices for different iterations be i.i.d. with the same distribution as $\boldsymbol{W}$, and ρ denote the *spectral norm* (i.e., the largest singular value) of $\mathbb{E}[\boldsymbol{W}^\top\boldsymbol{W}] - \boldsymbol{J}$, i.e.,

$$\rho := \|\mathbb{E}[\boldsymbol{W}^\top\boldsymbol{W}] - \boldsymbol{J}\|. \tag{9}$$

It has been shown in [14] that to guarantee convergence for D-PSGD, i.e., $\boldsymbol{x}_i^{(k)}$ for all $i \in V$ will converge to the same $\boldsymbol{x}$ as k increases, we need to have $\rho < 1$. A more precise bound is available under the following assumptions:

1) Each local objective function $F_i(\boldsymbol{x})$ is differentiable with an l-Lipschitz gradient, i.e., $\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{x}')\| \leq l\|\boldsymbol{x} - \boldsymbol{x}'\|$, $\forall i \in V$.
2) Stochastic gradients at each node are unbiased estimates of the true local gradient, i.e., $\mathbb{E}[g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)})|\mathcal{F}^{(k)}] = \nabla F_i(\boldsymbol{x}_i^{(k)})$, $\forall i \in V, k \geq 1$, where $\mathcal{F}^{(k)}$ is the σ-algebra capturing all the randomness until iteration k.
3) The variance of the stochastic gradient at each node is uniformly bounded by σ^2, i.e., $\mathbb{E}[\|g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)}) - \nabla F_i(\boldsymbol{x}_i^{(k)})\|^2|\mathcal{F}^{(k)}] \leq \sigma^2$, $\forall i \in V, k \geq 1$.
4) The deviation between the local gradients and the global gradient is bounded by ζ^2, i.e., $\frac{1}{m}\sum_{i\in V}\|\nabla F_i(\boldsymbol{x}) - \nabla F(\boldsymbol{x})\|^2 \leq \zeta^2$, $\forall \boldsymbol{x}$.

We note that these assumptions are commonly made in convergence analysis for decentralized SGD [14], [13], [16]. We also note that these assumptions are only related to the objective functions, the data distributions, the mini-batch sampling mechanism, and the mini-batch size, but not our design parameters of interest (i.e., $\mathcal{L}$, $\boldsymbol{p}$).

As the objective function $F(\boldsymbol{x})$ can generally be non-convex, convergence to the global minimizer cannot be guaranteed. Instead, convergence is considered to be achieved if the time-averaged expected gradient norm $\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\nabla F(\overline{\boldsymbol{x}}^{(k)})\|^2]$ is sufficiently small [14], [13], [16], where $\overline{\boldsymbol{x}}^{(k)} := \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i^{(k)}$. The following theorem gives a sufficient condition for this type of convergence.

Theorem III.1. [14, Theorem 2] Under assumptions (1–4), if the learning rate $\eta := \sqrt{\frac{m}{K}}$ satisfies $\eta l \leq \min\left\{1, \left(\sqrt{\rho^{-1}} - 1\right)/4\right\}$, the mixing matrices $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$ are i.i.d., and each $\boldsymbol{W}^{(k)}$ is symmetric with every row/column summing to one[3], then after K iterations,

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\left\|\nabla F(\overline{\boldsymbol{x}}^{(k)})\right\|^2\right] \leq \frac{8[F(\overline{\boldsymbol{x}}^{(1)}) - F_{\inf}] + 4l\sigma^2}{\sqrt{mK}} + \\ + \frac{8ml^2\rho}{K(1-\sqrt{\rho})}\left(\frac{\sigma^2}{1+\sqrt{\rho}} + \frac{3\zeta^2}{1-\sqrt{\rho}}\right), \tag{10}$$

where $\overline{\boldsymbol{x}}^{(1)}$ is the initial parameter vector, $F_{\inf}$ is a lower bound on $F(\cdot)$, and ρ is defined as in (9).

Theorem III.1 implies the following requirement on the number of iterations to achieve a given level of convergence.

Corollary III.2. Under the assumptions in Theorem III.1, to ensure that $\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\nabla F(\overline{\boldsymbol{x}}^{k})\|^2] \leq \epsilon_0$ for any given $\epsilon_0 > 0$, it suffices for the number of iterations to be

$$K(\rho) := \frac{c_2}{\epsilon_0} + \frac{c_1^2}{2\epsilon_0^2} + \frac{c_1\sqrt{4c_2\epsilon_0 + c_1^2}}{2\epsilon_0^2}, \tag{11}$$

where

$$c_1 = \frac{8(F(\overline{\boldsymbol{x}}^{(1)}) - F_{\inf}) + 4l\sigma^2}{\sqrt{m}}, \tag{12}$$

$$c_2 = \frac{8ml^2\sigma^2}{1-\rho} + \frac{24ml^2\zeta^2}{(1-\sqrt{\rho})^2}. \tag{13}$$

Remark: Theorem III.1 may be applicable only if $\rho < 1$. Under this condition, the design parameters $\mathcal{L}$ and $\boldsymbol{p}$ affect the number of iterations $K(\rho)$ required for ϵ_0-convergence only through ρ, where a smaller ρ means fewer iterations.

The above analysis is based on assumptions (1–4), which impose limitations on the supported objective functions and data distributions. In particular, assumption (3) (uniformly bounded variance of gradient estimate) and assumption (4) (uniformly bounded gradient divergence) were considered strong assumptions that have been relaxed in [28]. Specifically, for possibly non-convex objective functions as considered in our work, [28] provided a new convergence bound under the following relaxations of assumptions (3–4):

3') There exist constants $M_1, \hat{\sigma}$ such that $\forall \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$,

$$\frac{1}{m}\sum_{i\in V}\mathbb{E}[\|g(\boldsymbol{x}_i;\xi_i) - \nabla F_i(\boldsymbol{x}_i)\|^2] \leq \hat{\sigma}^2 + \frac{M_1}{m}\sum_{i\in V}\|\nabla F(\boldsymbol{x}_i)\|^2.$$

4') There exist constants $M_2, \hat{\zeta}$ such that $\forall \boldsymbol{x}$,

$$\frac{1}{m}\sum_{i\in V}\|\nabla F_i(\boldsymbol{x})\|^2 \leq \hat{\zeta}^2 + M_2\|\nabla F(\boldsymbol{x})\|^2.$$

Assumptions (3'–4') are weaker (and hence easier to satisfy) than assumptions (3–4) because they are satisfied whenever assumptions (3–4) are satisfied as shown in [28].

Theorem III.3. [28, Theorem 2] Under assumptions (1), (3'), and (4'), if there exist constants $p \in (0, 1]$ and integer $\tau \geq 1$ such that the mixing matrices $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$, each being symmetric and doubly stochastic[4], satisfy

$$\mathbb{E}[\|\boldsymbol{X}\prod_{k=k'\tau+1}^{(k'+1)\tau}\boldsymbol{W}^{(k)} - \boldsymbol{X}\boldsymbol{J}\|_F^2] \leq (1-p)\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{J}\|_F^2 \tag{14}$$

for all $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$ and integer $k' \geq 0$, then D-PSGD can achieve $\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[\|\nabla F(\overline{\boldsymbol{x}}^{k})\|^2] \leq \epsilon_0$ for any given $\epsilon_0 > 0$ when the number of iterations reaches

$$O\left(\frac{\hat{\sigma}^2}{m\epsilon_0^2} + \frac{\hat{\zeta}\tau\sqrt{M_1+1} + \hat{\sigma}\sqrt{p\tau}}{p\epsilon_0^{3/2}} + \frac{\tau\sqrt{(M_2+1)(M_1+1)}}{p\epsilon_0}\right) \\ \cdot l(F(\overline{\boldsymbol{x}}^{(1)}) - F_{\inf}). \tag{15}$$

Besides relaxing assumptions (3–4), Theorem III.3 also generalizes Theorem III.1 in that it allows the mixing matrices to be non-i.i.d., as long as (14) is satisfied. For any given

[3]Although [14] assumed $\boldsymbol{W}^{(k)}$ to be symmetric and doubly stochastic, the proof of [14, Theorem 2] only required it to be symmetric with rows/columns summing to one. We thus use this relaxed assumption in mixing matrix design.

[4]In addition to having rows/columns summing to one, being doubly stochastic also requires the entries $\in [0, 1]$. This constraint is only needed to prove [28, Theorem 2] when $\tau > 1$, but not needed under our design.

$\tau \geq 1$, the number of iterations according to (15) depends on the mixing matrices only through parameter p, where the larger p is, the smaller the required number of iterations. In the special case of $\tau = 1$, this implies a design objective of designing a randomized matrix $\boldsymbol{W}$, used to generate i.i.d. mixing matrices across iterations, by maximizing

$$p := \min_{\boldsymbol{X} \neq \boldsymbol{0}} \left(1 - \frac{\mathbb{E}[\|\boldsymbol{X}(\boldsymbol{W} - \boldsymbol{J})\|_F^2]}{\|\boldsymbol{X}(\boldsymbol{I} - \boldsymbol{J})\|_F^2}\right). \tag{16}$$

However, (16) is difficult to optimize as it is not a closed-form function of $\boldsymbol{W}$. Nevertheless, as it is easy to obtain upper bounds on (16), we can try to maximize a good upper bound. To this end, we have found an upper bound that reduces the design objective under the relaxed assumptions (3'–4') to the original design objective.

Lemma III.4. For any randomized mixing matrix $\boldsymbol{W}$ that is symmetric with every row/column summing to one, p as defined in (16) satisfies $p \leq 1 - \rho$ for $\rho := \|\mathbb{E}[\boldsymbol{W}^\top \boldsymbol{W}] - \boldsymbol{J}\|$.

We have empirically verified that $1 - \rho$ is a tight upper bound on p (e.g., $(1 - \rho) - p \leq 10^{-4}$ for $\boldsymbol{W}$ designed by our algorithm). Thus, approximating p by $1 - \rho$ reduces the design objective to minimizing ρ, which coincides with the original design objective derived under assumptions (1–4). This observation together with the efficacy of mixing matrices designed via ρ minimization (see Section V) suggests the value of ρ as an objective of mixing matrix design.

Remark: As noted earlier, Theorem III.3 allows the mixing matrices to be non-i.i.d. as long as (14) is satisfied for some $p \in (0, 1]$ and $\tau \geq 1$. This extends the solution space for mixing matrix design to the joint design of a period τ and a sequence of randomized mixing matrices $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_\tau$, such that the number of iterations (15) for a variation of D-PSGD that iterates through these mixing matrices repeatedly is minimized. In this work, we focus on the special case of $\tau = 1$, which is already challenging as explained in Section IV, and leave the general case to future work.

B. Bilevel Parameter Optimization

Based on the analysis in Section III-A, we should design the candidate Laplacian matrices $\mathcal{L}$ and their sampling probabilities $\boldsymbol{p}$ to minimize the spectral norm ρ in order to minimize the required number of iterations $K(\rho)$ according to (11). However, minimizing ρ alone may not minimize the total cost, as it may require activating more links and hence incurring more cost per iteration as discussed in Section II-C. To find the optimal tradeoff, we introduce an auxiliary design parameter C, representing the *budget on the expected cost per iteration*. This allows us to formulate our problem as a bilevel optimization:

Lower-level optimization: design $\boldsymbol{p}$ and $\mathcal{L}$ to minimize ρ under a given budget C, which results in a spectral norm of $\rho(C)$ and a required number of iterations of $K(\rho(C))$.
Upper-level optimization: design C to minimize the total cost $C \cdot K(\rho(C))$.

The above bilevel formulation preserves optimality in the following sense.

Lemma III.5. Suppose that the number of iterations of D-PSGD is determined by (11). Then the design $(\mathcal{L}^o, \boldsymbol{p}^o)$ obtained by solving the bilevel optimization to optimality will minimize the expected total cost.

IV. Communication-efficient Design for D-PSGD

As the upper-level optimization only has one scalar decision variable C, our focus will be on the lower-level optimization, which is tackled below.

A. Formulation of Lower-level Optimization

We start by writing the spectral norm ρ as an explicit function of the design parameters $\boldsymbol{p}$ and $\mathcal{L}$. By construction, $\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)} - \boldsymbol{J}$ is symmetric. Moreover, by definition (3), we have that $\boldsymbol{W}^{(k)}\boldsymbol{J} = \boldsymbol{J}\boldsymbol{W}^{(k)} = \boldsymbol{J}$. This combined with the fact of $\boldsymbol{J}^2 = \boldsymbol{J}$ implies that $\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)} - \boldsymbol{J} = \left(\boldsymbol{W}^{(k)} - \boldsymbol{J}\right)^\top \left(\boldsymbol{W}^{(k)} - \boldsymbol{J}\right)$, and hence $\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)} - \boldsymbol{J}$ is positive semi-definite. Thus, its expectation $\mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \boldsymbol{J}$ is also symmetric and positive semi-definite. This implies that ρ defined in (9) is equal to the largest eigenvalue of $\mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \boldsymbol{J}$. Minimizing (9) is thus equivalent to

$$\min \rho \quad \text{s.t. } \mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \boldsymbol{J} \preceq \rho \boldsymbol{I}, \tag{17}$$

where $\boldsymbol{A} \preceq \boldsymbol{B}$ means that matrix $\boldsymbol{B} - \boldsymbol{A}$ is positive semi-definite. This is because by the eigendecomposition $\mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \boldsymbol{J} = \boldsymbol{Q}\,\mathrm{diag}(\lambda_1, \ldots, \lambda_m)\boldsymbol{Q}^\top$, (17) is equivalent to minimizing ρ subject to $\rho \geq \max_{i=1,\ldots,m} \lambda_i$, for which the optimal solution must satisfy $\rho = \max_{i=1,\ldots,m} \lambda_i$.

Based on the above analysis, we can formulate the lower-level optimization as

$$\min_{\boldsymbol{p}, (\boldsymbol{\alpha}_j)_{j=1}^n} \quad \rho \tag{18a}$$

$$\text{s.t. } \boldsymbol{I} - 2\sum_{j=1}^n p_j \boldsymbol{L}_j + \sum_{j=1}^n p_j \boldsymbol{L}_j^\top \boldsymbol{L}_j - \boldsymbol{J} \preceq \rho \boldsymbol{I}, \tag{18b}$$

$$\sum_{j=1}^n p_j c(\boldsymbol{L}_j) \leq C, \tag{18c}$$

$$\sum_{j=1}^n p_j = 1, \tag{18d}$$

$$p_j \geq 0, \quad \forall j = 1, \ldots, n, \tag{18e}$$

$$\boldsymbol{L}_j = \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}_j)\boldsymbol{B}^\top, \quad \forall j = 1, \ldots, n, \tag{18f}$$

$$\boldsymbol{\alpha}_j \geq \boldsymbol{0}, \quad \forall j = 1, \ldots, n, \tag{18g}$$

where the main decision variables are $(\boldsymbol{\alpha}_j)_{j=1}^n$ (that specifies the set of candidate Laplacian matrices $\mathcal{L} := \{\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}_j)\boldsymbol{B}^\top\}_{j=1}^n$) and $\boldsymbol{p} = (p_j)_{j=1}^n$. Constraint (18b) ensures $\rho = \|\mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \boldsymbol{J}\|$ under the optimal solution, where the *left-hand side (LHS)* of (18b) is an expansion of $\mathbb{E}[\boldsymbol{W}^{(k)\top}\boldsymbol{W}^{(k)}] - \boldsymbol{J}$; (18c) requires the expected cost per iteration to be bounded by C; (18d)–(18e) ensure that $\boldsymbol{p}$ is a valid probability distribution; (18f) relates the weighted Laplacian matrices to the designed link weights.

However, (18) is hard to solve due to the non-linear matrix inequality constraint (18b), which generally makes the optimization NP-hard [29]. Below we will address this challenge by optimizing $\boldsymbol{p}$ and $\mathcal{L}$ separately.

B. Design of Sampling Probabilities

Given a set $\mathcal{L}$ of candidate Laplacian matrices, (18) is reduced to the optimization of a linear objective function (19a) under a linear matrix inequality (19b) and linear constraints (19c)–(19e):

$$\min_{\boldsymbol{p}} \ \rho \tag{19a}$$

$$\text{s.t. } \boldsymbol{I} - 2\sum_{j=1}^{n} p_j \boldsymbol{L}_j + \sum_{j=1}^{n} p_j \boldsymbol{L}_j^\top \boldsymbol{L}_j - \boldsymbol{J} \preceq \rho \boldsymbol{I}, \tag{19b}$$

$$\sum_{j=1}^{n} p_j c(\boldsymbol{L}_j) \le C, \tag{19c}$$

$$\sum_{j=1}^{n} p_j = 1, \tag{19d}$$

$$p_j \ge 0, \quad \forall j = 1, \dots, n. \tag{19e}$$

We note that (19) is a (linear) semi-definite programming (SDP) problem that is convex [30] and can thus be solved in polynomial time, e.g., using the interior point method [30].

C. Design of Candidate Laplacian Matrices

The design of candidate Laplacian matrices is much harder because it involves the design of candidate topologies to activate, which has a discrete solution space. We formulate this problem as a special case of (18) with $n = 1$. In this case, (18) reduces to the design of a weighted Laplacian matrix $\boldsymbol{L}$ such that $\rho := \|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{J}\|$ for $\boldsymbol{W} = \boldsymbol{I} - \boldsymbol{L}$ is minimized under $c(\boldsymbol{L}) \le C$, which helps maximize the convergence rate subject to the given budget C. As $\|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{J}\| = \|\boldsymbol{W} - \boldsymbol{J}\|^2$, minimizing ρ is equivalent to minimizing $\tilde{\rho} := \|\boldsymbol{W} - \boldsymbol{J}\|$.

The specific formulation of this optimization depends on the cost model $c(\boldsymbol{L})$. Below we will detail the optimization under the cost model in (7) as a concrete example, but analogous optimizations can be formulated under the other cost models discussed in Section II-C. Under the cost model in (7), the design of $\boldsymbol{L} := \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$ can be formulated as follows:

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \ \tilde{\rho} \tag{20a}$$

$$\text{s.t. } -\tilde{\rho}\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \tilde{\rho}\boldsymbol{I}, \tag{20b}$$

$$\alpha_{(i,j)} \le m\beta_{(i,j)}, \quad \forall (i,j) \in E, \tag{20c}$$

$$\sum_{i=1}^{m} (c_i^a + \sum_{j:\,(i,j)\in E} c_{ij}^b \beta_{(i,j)}) \le C, \tag{20d}$$

$$\alpha_{(i,j)} \ge 0,\ \beta_{(i,j)} \in \{0,1\} \quad \forall (i,j) \in E, \tag{20e}$$

where $\boldsymbol{\alpha}$ is the primary decision variable and $\boldsymbol{\beta}$ is a dependent variable. We claim that:

1) (20a) and (20b) set the objective to $\min \|\boldsymbol{W} - \boldsymbol{J}\|$, and
2) (20c) – (20e) enforce the constraint of $c(\boldsymbol{L}) \le C$.

To see the first point, let $\tilde{\lambda}_1, \dots, \tilde{\lambda}_m$ denote the eigenvalues of $\boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J}$. Constraint (20b) ensures that $\tilde{\rho} \ge \max(\tilde{\lambda}_i, -\tilde{\lambda}_i)$ for all $i = 1, \dots, m$. This together with the objective (20a) implies that $\tilde{\rho} = \max_{i=1,\dots,m} |\tilde{\lambda}_i|$ under the optimal solution, which in turn equals $\|\boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J}\|$ as $\boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J}$ is a symmetric matrix.

To see the second point, note that (20c) forces $\beta_{(i,j)}$ to be 1 whenever $\alpha_{(i,j)} > 0$, indicating that link (i,j) is activated. Based on this, (20d) enforces $c(\boldsymbol{L}) \le C$ according to the definition in (7). We can express the relationship between $\alpha_{(i,j)}$ and $\beta_{(i,j)}$ by the linear inequality (20c) because it suffices to consider $\alpha_{(i,j)}$'s that are upper-bounded by the number of nodes m, as shown below.

Lemma IV.1. To achieve $\rho < 1$ for $\rho := \|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{J}\|$ and $\boldsymbol{W} := \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$, we must have $\alpha_{(i,j)} < m$ for all $(i,j) \in E$.

As discussed in Section III-A, we need $\rho < 1$ to guarantee convergence for D-PSGD, and thus it suffices to consider the designs that satisfy $\alpha_{(i,j)} < m$ for all $(i,j) \in E$.

Unfortunately, (20) is hard to solve. Specifically, as $\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top = \sum_{(i,j)\in E} \alpha_{(i,j)} \boldsymbol{b}_{(i,j)} \boldsymbol{b}_{(i,j)}^\top$ ($\boldsymbol{b}_{(i,j)}$: column in $\boldsymbol{B}$ corresponding to link (i,j)), (20b) is a linear matrix inequality in $\boldsymbol{\alpha}$. The rest of the constraints are linear, except for the integer constraints on $\boldsymbol{\beta}$. This makes (20) a mixed-integer SDP problem, which is generally NP-hard [31]. To address this challenge, we propose the following solutions, which will be combined to construct a set of candidate Laplacian matrices with diverse cost-convergence tradeoffs (see Section IV-D).

1) Optimal Design without Budget Constraint: We note that the hardness of (20) is due to the budget constraint (20d), which triggers the need for the integer variable $\boldsymbol{\beta}$. Without this constraint, (20) is reduced to

$$\min_{\boldsymbol{\alpha}} \ \tilde{\rho} \tag{21a}$$

$$\text{s.t. } -\tilde{\rho}\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \tilde{\rho}\boldsymbol{I}, \tag{21b}$$

$$\boldsymbol{\alpha} \ge \boldsymbol{0}, \tag{21c}$$

which is an SDP that can be solved in polynomial time by existing algorithms such as [30].

2) Greedy Heuristic: Intuitively, the link (i,j) with the smallest weight $\alpha_{(i,j)}$ transmits the least important information. Thus, we can remove this link from the activated topology (by setting $\alpha_{(i,j)}$ to 0) and recalculate the optimal Laplacian matrix by (21) in an iterative manner, until the designed Laplacian matrix $\boldsymbol{L}$ satisfies the budget constraint $c(\boldsymbol{L}) \le C$. The pseudocode of this algorithm is shown in Algorithm 1. The algorithm is guaranteed to succeed for any feasible budget $C \ge c(\boldsymbol{0})$.

The complexity of Algorithm 1 is dominated by line 6, which is executed at most $O(|E|)$ times as each **while** loop will deactivate one link. Each execution of line 6 takes polynomial time in $|E|$ and m for solving an SDP with $|E|$ variables and $m + |E|$ constraints (where the exact order of polynomial depends on the algorithm used to solve the SDP [30]). The overall complexity is thus polynomial in $|E|$ and m.

3) Sparsifiers: Consider the special case of (7) with homogeneous communication costs $c_{ij}^b \equiv c^b$ for all $(i,j) \in E$. This reduces the budget constraint (20d) to a sparsity constraint:

$$\sum_{(i,j)\in E} \beta_{(i,j)} \le C_E := \lfloor \frac{1}{2c^b}(C - \sum_{i=1}^{m} c_i^a) \rfloor, \tag{22}$$

where C_E is the maximum number of activated links. In this case, we can solve (20) with performance guarantee.

Algorithm 1: Greedy Laplacian Matrix Design

input : Incidence matrix $\boldsymbol{B}$, budget C, cost model $c(\cdot)$
output: Weighted Laplacian matrix $\boldsymbol{L}$
$E_r \leftarrow \emptyset$; // set of deactivated links
Solve $\boldsymbol{\alpha}$ from (21);
while $c(\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top) > C$ **do**
  Find the link $(i,j) \in E \setminus E_r$ with the smallest $\alpha_{(i,j)}$;
  $E_r \leftarrow E_r \cup \{(i,j)\}$;
  Solve $\boldsymbol{\alpha}$ from (21) with the additional constraint $\alpha_{(i,j)} = 0,\ \forall (i,j) \in E_r$;
Return $\boldsymbol{L} \leftarrow \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$;

Algorithm 2: Sparsification-based Laplacian Matrix Design

input : Incidence matrix $\boldsymbol{B}$, sample size q
output: Weighted Laplacian matrix $\boldsymbol{L}_H$
Compute the optimal Laplacian matrix before sparsification $\boldsymbol{L} \leftarrow \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$, where $\boldsymbol{\alpha}$ is the solution to (21);
Compute effective resistance $\boldsymbol{R} \leftarrow \mathrm{diag}(\boldsymbol{B}^\top \boldsymbol{L}^+ \boldsymbol{B})$, where $\boldsymbol{L}^+$ is Moore-Penrose pseudoinverse of $\boldsymbol{L}$;
Compute sampling probabilities $\boldsymbol{P} \in \mathbb{R}^{|E|}$ by $P_i \propto \alpha_i R_i$ and $\sum_i P_i = 1$;
Initialize link weights of the sparsifier $\boldsymbol{s} \leftarrow \{0\}^{|E|}$;
repeat q **times**
  Randomly sample a link $e_i \in E$ with probability P_i;
  $s_i \leftarrow s_i + \frac{\alpha_i}{qP_i}$;
return $\boldsymbol{L}_H \leftarrow \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^\top$;

The solution is inspired by the following relationship between the objective of (20) and the eigenvalues of the designed Laplacian matrix (recall that $\lambda_i(\cdot)$ is the i-th smallest eigenvalue).

Lemma IV.2. For any (weighted) Laplacian matrix $\boldsymbol{L}$,

$$\tilde{\rho} := \|\boldsymbol{I} - \boldsymbol{L} - \boldsymbol{J}\| = \max\{1 - \lambda_2(\boldsymbol{L}),\ \lambda_m(\boldsymbol{L}) - 1\}. \quad (23)$$

Lemma IV.2 allows us to connect (20) to the problem of graph (spectral) sparsification [32]. The idea is: as the Laplacian matrix $\boldsymbol{L}$ that minimizes a function (23) of its eigenvalues without constraint (22) is easily computable by (21), an algorithm that sparsifies $\boldsymbol{L}$ while approximating its eigenvalues should provide an approximate solution to the minimization of $\tilde{\rho}$ under (22).

Algorithm: There is an efficient randomized graph sparsification algorithm [32], which can generate sparsifiers with $O(m \log m/\epsilon^2)$ links that approximate the eigenvalues of the original Laplacian matrix within a factor of $1 \pm \epsilon$. Based on this algorithm, we propose Algorithm 2. First, we solve (21) in line 1 for the Laplacian matrix $\boldsymbol{L}$ that minimizes $\tilde{\rho}$ while potentially activating all the links in G. Then we compute sampling probabilities in lines 2–3, where the probability P_i for sampling link e_i is proportional to the product of its original weight α_i and its effective resistance[5] R_i. According to [32], the effective resistances $\boldsymbol{R} \in \mathbb{R}^{|E|}$ are given by the diagonal entries of $\boldsymbol{B}^\top \boldsymbol{L}^+ \boldsymbol{B}$, where $\boldsymbol{B}$ is incidence matrix of G and $\boldsymbol{L}^+$ is Moore-Penrose pseudoinverse of $\boldsymbol{L}$, defined as

$$\boldsymbol{L}^+ = \sum_{i:\lambda_i > 0} \frac{1}{\lambda_i} \boldsymbol{u_i} \boldsymbol{u_i}^\top, \quad (24)$$

where λ_i and $\boldsymbol{u}_i$ are the eigenvalue and the corresponding orthonormal eigenvector of $\boldsymbol{L}$. Next, we construct the sparsifier by taking q samples from the links in G independently with replacement according to the probability distribution $\boldsymbol{P}$ as in lines 4–7. Each time a link e_i is sampled, its weight s_i in the sparsifier will be increased by $\alpha_i/(qP_i)$. As a result, we obtain a sparsifer with $O(q)$ links and a Laplacian matrix $\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^\top$.

Analysis: We first recall the following result from [32].

Lemma IV.3. ([32, Theorem 1]) Given a weighted Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{m \times m}$ and a constant $\epsilon \in (\frac{1}{\sqrt{m}}, 1]$, there exists a constant[6] c_r such that if $q = \frac{c_r m \log m}{\epsilon^2}$ and m is sufficiently large, then with probability at least $1/2$, $\boldsymbol{L}_H := \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^\top$ for $\boldsymbol{s}$ generated by lines 4–7 of Algorithm 2 satisfies

$$(1-\epsilon)\boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y} \le \boldsymbol{y}^\top \boldsymbol{L}_H \boldsymbol{y} \le (1+\epsilon)\boldsymbol{y}^\top \boldsymbol{L}\boldsymbol{y}, \quad \forall \boldsymbol{y} \in \mathbb{R}^m. \quad (25)$$

In particular, the eigenvalue $\lambda_i(\cdot)$ $(\forall i = 1, \ldots, m)$ satisfies

$$(1-\epsilon)\lambda_i(\boldsymbol{L}) \le \lambda_i(\boldsymbol{L}_H) \le (1+\epsilon)\lambda_i(\boldsymbol{L}). \quad (26)$$

Based on Lemma IV.3, we show that under certain conditions, Algorithm 2 achieves a guaranteed approximation with a guaranteed probability.

Theorem IV.4. Consider a special case of (20) when (20d) is replaced by (22). Let $\tilde{\rho}^*$ be the optimal objective value, and $\epsilon := \frac{1}{\delta}\sqrt{c_r m \log m / C_E}$ for

$$\delta := \min\left\{ \left| \frac{1 - \lambda_2(\boldsymbol{L})}{\lambda_2(\boldsymbol{L})} \right|, \left| \frac{\lambda_m(\boldsymbol{L}) - 1}{\lambda_m(\boldsymbol{L})} \right| \right\}, \quad (27)$$

where $\boldsymbol{L}$ is computed in line 1 of Algorithm 2, and c_r is the constant in Lemma IV.3. If $C_E \ge c_r m \log m$, then the output $\boldsymbol{L}_H$ of Algorithm 2 for $q = C_E$ satisfies the following with probability at least $1/2$:

$$\tilde{\rho}_H := \|\boldsymbol{I} - \boldsymbol{L}_H - \boldsymbol{J}\| \le (1+\epsilon)\tilde{\rho}^*. \quad (28)$$

Two remarks are in order: *First*, although a randomly-generated sparsifier only satisfies (28) with probability at least $1/2$, one can generate r independent sparsifiers by repeating lines 4–7 and return the sparsifier with the minimum $\tilde{\rho}_H$, which will satisfy (28) with probability at least $1 - 2^{-r}$. However, this is not necessary in our case as each sparsifier only gives a candidate Laplacian matrix, and we will further optimize how often to use this candidate (see Section IV-B). It hence does not hurt to include all the generated sparsifiers as candidates (see Algorithm 3). *Moreover*, although Algorithm 2 is designed for a specific cost model (7) with $c_{ij}^b \equiv c^b$ $(\forall (i,j) \in E)$, it can be applied under a general cost model $c(\cdot)$ by replacing the fixed loop in lines 5–7 by an indefinite loop that keeps running as long as $c(\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^\top) \le C$ (although the performance guarantee in Theorem IV.4 no longer applies).

Complexity: In Algorithm 2, solving the SDP (21) in line 1 takes polynomial time in $|E|$ and m. Line 2 takes $O(m^3)$

[5]If viewing G as an electrical network and each weight α_i as the conductance of link e_i, then R_i is the potential difference induced across e_i when a unit current is injected at one end of the link and extracted at the other end.

[6]Specifically, $c_r = 144 c_0^2$, where c_0 is the constant in [33, Lemma 3.5].

Algorithm 3: Laplacian Matrix Sampling (LMS)

input : Incidence matrix $\boldsymbol{B}$, requirement on #iterations $K(\cdot)$ as given in (11), cost model $c(\cdot)$, budgets $\mathcal{C}_g$ for Algorithm 1, sample sizes $\mathcal{Q}$ for Algorithm 2, candidate budgets for upper-level optimization $\mathcal{C}$

output: Laplacian matrices $\mathcal{L}$, sampling probabilities $\boldsymbol{p}$

1 $\mathcal{L} \leftarrow \{\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top\}$ for the solution $\boldsymbol{\alpha}$ to (21);
2 **foreach** $C \in \mathcal{C}_g$ **do**
3 $\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{\boldsymbol{L}\}$, where $\boldsymbol{L}$ is the output of Algorithm 1 for budget C;
4 **foreach** $q \in \mathcal{Q}$ **do**
5 $\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{\boldsymbol{L}_H\}$, where $\boldsymbol{L}_H$ is the output of Algorithm 2 for sample size q;
6 $\boldsymbol{p} \leftarrow$ solution to (19) without constraint (19c), with objective value ρ;
7 **for** $C \in \mathcal{C}$ **do**
8 $\quad \boldsymbol{p}' \leftarrow$ solution to (19) under budget C, with objective value ρ';
9 $\quad$ **if** $K(\rho')\sum_{j:\boldsymbol{L}_j\in\mathcal{L}} p'_j c(\boldsymbol{L}_j) < K(\rho)\sum_{j:\boldsymbol{L}_j\in\mathcal{L}} p_j c(\boldsymbol{L}_j)$ **then**
10 $\quad\quad \boldsymbol{p} \leftarrow \boldsymbol{p}'$, $\rho \leftarrow \rho'$;
11 return $\mathcal{L}$, $\boldsymbol{p}$;

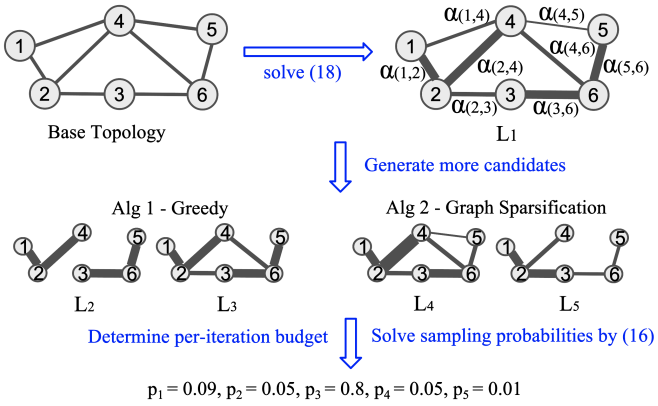


Fig. 1. Illustration of the workflow of LMS

time to compute the pseudoinverse $\boldsymbol{L}^+$ and $O(m^3)$ time to compute the effective resistance $\boldsymbol{R}$. Then line 3 computes the link sampling probabilities in $O(|E|)$ time. Afterwards, it takes $O(q)$ time to sample links as in lines 4–7, and then $O(|E|)$ time to construct the Laplacian matrix $\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{s})\boldsymbol{B}^\top$. The overall complexity is thus polynomial in $|E|$ and m, and linear in q.

D. Overall Solution

1) Algorithm: Based on Sections IV-B–IV-C, we propose an overall algorithm that designs $\mathcal{L}$ and $\boldsymbol{p}$ for D-PSGD, called *Laplacian Matrix Sampling (LMS)*, which is presented in Algorithm 3 and illustrated in Fig. 1.

LMS first constructs a set $\mathcal{L}$ of candidate Laplacian matrices with various tradeoffs between the convergence rate and the cost per iteration. This includes the Laplacian matrix designed to minimize $\tilde{\rho}$ without constraining the per-iteration cost (line 1), the outputs of the greedy heuristic under a given set $\mathcal{C}_g$ of budgets (lines 2–3), and the outputs of the graph sparsification algorithm under a given set $\mathcal{Q}$ of sample sizes (lines 4–5). It then optimizes the probability distribution for sampling from $\mathcal{L}$ under a given set of candidate budgets, and picks the distribution $\boldsymbol{p}$ that minimizes the expected total cost for running D-PSGD until convergence, given by $K(\rho)\sum_{j:\boldsymbol{L}_j\in\mathcal{L}} p_j c(\boldsymbol{L}_j)$ (lines 7–10), where $K(\rho)$ is the required number of iterations as in (11). Given the designed parameters $(\mathcal{L}, \boldsymbol{p})$, D-PSGD will be executed based on randomized mixing matrices $\boldsymbol{W}^{(k)} := \boldsymbol{I} - \boldsymbol{L}^{(k)}$ $(k = 1, 2, \ldots)$ for Laplacian matrices $\boldsymbol{L}^{(1)}, \boldsymbol{L}^{(2)}, \ldots$ sampled i.i.d. from $\mathcal{L}$ according to the distribution $\boldsymbol{p}$.

LMS can be run by a task dispatcher *ahead of time* before launching learning tasks, which then distributes the designed parameters $(\mathcal{L}, \boldsymbol{p})$ to the nodes together with a pseudo random number generator seed. When receiving a learning task, the nodes can use their seeded pseudo random number generators to sample Laplacian matrices from $\mathcal{L}$ and communicate accordingly, where the shared seed ensures that all the nodes will sample the same Laplacian matrix in each iteration.

Due to the polynomial complexity of each of the invoked subroutines, the overall complexity of LMS is polynomial in $|E|$, m, n, $|\mathcal{C}_g|$, $|\mathcal{Q}|$, and $|\mathcal{C}|$. While some of these factors are design parameters, $|E|$ and m are determined by the base topology, which limits the scalability of our solution. In this regard, we envision LMS to be easily applicable for cross-silo learning (with < 100 nodes) or cross-device learning over a limited number of devices [2], and leave the development of more scalable (but less optimal) solutions that can support massive cross-device learning to future work.

Remark: Although the sparsifiers (generated by Algorithm 2) only provide guaranteed approximation in a special case (as stated in Theorem IV.4), we can always include them in the candidate set $\mathcal{L}$ under any cost model without degrading the performance, thanks to the fact that the sampling distribution $\boldsymbol{p}$ can be solved to optimality in polynomial time.

2) Analysis: LMS can achieve a guaranteed performance in minimizing the expected total cost till convergence as follows.

Theorem IV.5. Under the cost model in (7) with $c^b_{ij} \equiv c^b$ for all $(i,j) \in E$, let C^* be the expected per-iteration cost under the optimal design, $C^*_E := \lfloor \frac{1}{2c^b}(C^* - \sum_{i=1}^m c^a_i) \rfloor$, and ρ^* the corresponding spectral norm as defined in (9). If $C^*_E \geq c_r m \log m$, $C^*_E \in \mathcal{Q}$, and $C^* \in \mathcal{C}$, then the design $(\mathcal{L}, \boldsymbol{p})$ computed by LMS incurs an expected total cost of

$$K(\rho) \sum_{j:\boldsymbol{L}_j\in\mathcal{L}} p_j c(\boldsymbol{L}_j) \leq K\left((1+\epsilon)^2\rho^*\right) C^* \qquad (29)$$

with probability at least $1/2$, where $\rho := \|\sum_{j:\boldsymbol{L}_j\in\mathcal{L}} p_j(\boldsymbol{I} - \boldsymbol{L}_j)^\top(\boldsymbol{I} - \boldsymbol{L}_j) - \boldsymbol{J}\|$, and $\epsilon := \frac{1}{\delta}\sqrt{c_r m \log m / C^*_E}$ for δ defined in (27).

Remark: When the expected number of links activated by the optimal design is much larger than $c_r m \log m$, ϵ will be small, and hence the bound in (29) will be close to the minimum expected total cost $K(\rho^*)C^*$. Intuitively, this can occur in large dense networks where the total number of links can be $O(m^2)$. Moreover, by generating r independent sparsifiers for each $q \in \mathcal{Q}$ and including all of them into $\mathcal{L}$, we can improve the probability of satisfying (29) to at least $1 - 2^{-r}$.

3) Justification of Randomized Design: To justify our approach of designing the Laplacian matrix (and hence the mixing matrix) in a randomized manner, we note that first, as

shown in (20), designing a deterministic Laplacian matrix to minimize the spectral norm under a budget constraint will lead to a mixed-integer optimization that is hard to solve, for which we can only obtain suboptimal solutions. Moreover, optimally sampling from candidate solutions can improve performance. Specifically, if $\boldsymbol{W}_j := \boldsymbol{I} - \boldsymbol{L}_j$ for $\boldsymbol{L}_j \in \mathcal{L}$, then

$$\min_{\boldsymbol{p}} \| \sum_j p_j (\boldsymbol{W}_j^\top \boldsymbol{W}_j - \boldsymbol{J}) \| \leq \| \boldsymbol{W}_j^\top \boldsymbol{W}_j - \boldsymbol{J} \|, \quad \forall j, \quad (30)$$

i.e., suitable randomization among given candidates is no worse than deterministically using any of the candidates. Our design is guaranteed to satisfy (30) as we can compute the optimal $\boldsymbol{p}$ in polynomial time as explained in Section IV-B.

V. Performance Evaluation

We have conducted extensive data-driven simulations to evaluate the proposed solution in comparison with the state of the art under a variety of cost models of practical importance.

A. Experiment Setup

1) Dataset and ML model: We consider training for image classification based on CIFAR-10, which consists of 60,000 color images in 10 classes. We train the ResNet-50 model over its training dataset with 50,000 images, and then test the trained model over the testing dataset with 10,000 images.

2) Base Topology: We simulate the base topology based on real wireless/wired networks. In the wireless setting, we use the topology of the Roofnet [34] mesh network at data rate 1Mbps, which contains 33 nodes and 187 links. In the wired setting, we simulate the scenario in which nearby desktop applications communicate with each other for model training. To this end, we generate an overlay network based on the Cogent network from the Internet Topology Zoo [35], treating the degree-1 nodes as (gateways of) learning agents, and connecting each pair of degree-1 nodes within 14 hops of each other by an overlay link (i.e., allowing them to communicate during learning). The generated overlay network, which is the base topology in the wired setting, contains 21 nodes and 136 links.

3) Cost Model: In the wireless setting, we evaluate two cost models: communication time and energy consumption. The former is measured by the number of matchings in the activated topology according to [14], and the latter is measured by (7) under the assumption of unicast communication (see Section V-C2 for the case of broadcast communication), where we set the computation energy as $c_i^a = 0.0003342$ (Wh) and the communication energy as $c_{ij}^b = 0.0009129$ (Wh) based on our model size and the parameters from [36], [37]. In the wired setting, we also evaluate two cost models: communication time and bandwidth consumption. The former is again measured by the number of activated matchings as in [14]. The latter is measured by the total number of times the parameter vector is transmitted in the underlay network, calculated by the cost model in Section II-C3. We note that although using the number of activated matchings as the cost measure has limitations (e.g., not capturing the computation time or variations in the communication time), we still include this cost model to enable a fair comparison with MATCHA [14] under its cost measure.

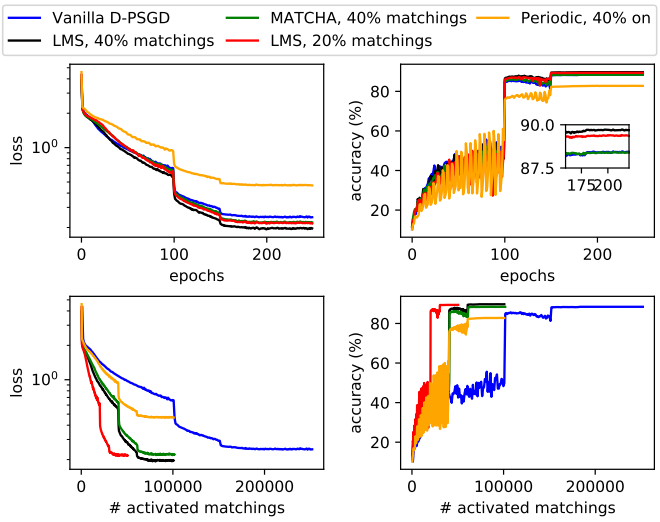


Fig. 2. Communication time in wireless network

4) Parameters: We set the learning rate as 0.8 at the beginning and reduce it by 10X after 100, 150, 180, 200 epochs, which is consistent with [14] (to facilitate comparison) while ensuring convergence. We set the mini-batch size to 32. For LMS, we set $n = 3091$ by default, by setting $\mathcal{C}_g = \{0.10, 0.11, ..., 0.99\}$ and constructing $\mathcal{Q}$ by generating 3000 instances of $q = \frac{0.09m \log m}{\epsilon^2}$ with ϵ sampled uniformly at random from $[0.01, 0.8]$. These parameters allow LMS to generate 3091 candidate Laplacian matrices (including the solution to (21)). We will test the sensitivity of LMS to these configuration parameters in Section V-C1.

5) Benchmarks: We compare LMS with three benchmarks: Vanilla D-PSGD (where all the neighbors communicate in all the iterations), Periodic (periodically, there are either communications over all the links or no communication at all), and MATCHA [14] (state of the art). We use the open-source code [38] provided by the authors of [14] for MATCHA. We use the approach in [14] to design the link weights for Vanilla and Periodic, which is the state of the art before our work. We first fine-tune the parameter of MATCHA to achieve the lowest loss at convergence, and then tune the communication frequency/budget for Periodic and the proposed algorithm (LMS) to align the average cost per iteration. We also test LMS when it is tuned to achieve the same loss as MATCHA at convergence, to evaluate its potential at saving cost while achieving the same quality of training. Our code is available at [39].

B. Performance Comparison

We have compared the learning performance under each of the designs via training loss and testing accuracy, in a variety of settings. The results, given in Fig. 2–5, show that: (i) all the algorithms with budgeted communications can achieve convergence at a much lower cost than Vanilla D-PSGD; (ii) compared to Vanilla D-PSGD, Periodic converges to a worse model (with a higher loss and a lower accuracy), but MATCHA and both versions of LMS all converge to models that are similar or better, signifying the value of fine-grained

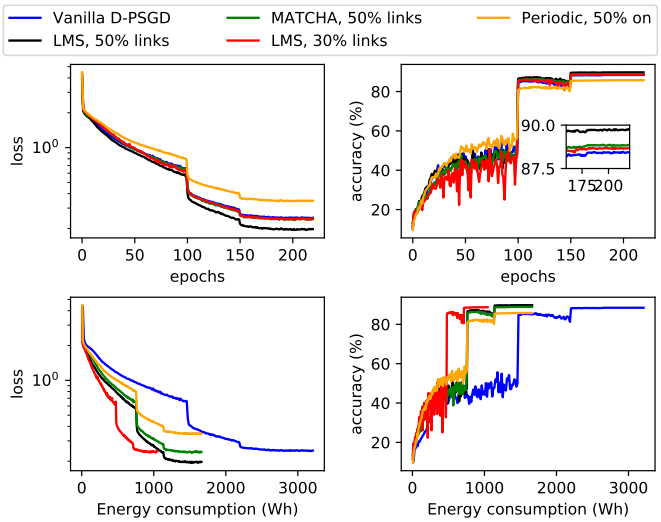


Fig. 3. Energy consumption in wireless network

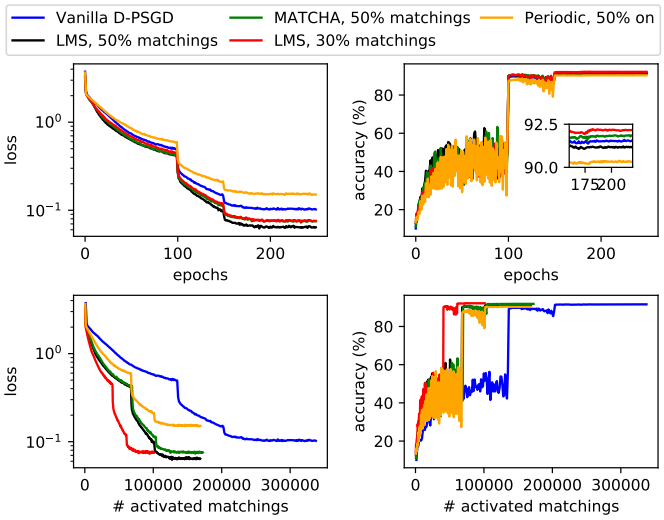


Fig. 4. Communication time in wired network

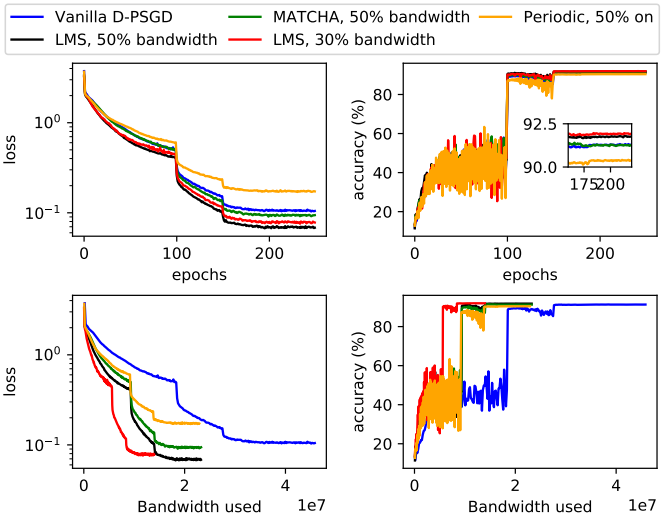


Fig. 5. Bandwidth consumption in wired network

communication design; (iii) LMS outperforms MATCHA by either achieving a lower loss and a higher accuracy at the same cost[7] or achieving the same loss/accuracy at a lower cost. While the difference in loss/accuracy at convergence seems small, the difference in the cost to achieve convergence is significant: LMS saves 40–50% of the cost compared to MATCHA and 70–80% of the cost compared to Vanilla D-PSGD, while yielding a model of similar or better quality.

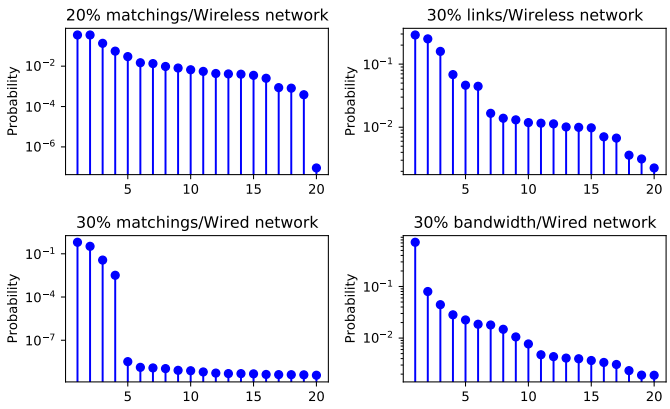


Fig. 6. Probabilities of sampling the top 20 candidate Laplacian matrices under LMS ($n = 3091$)

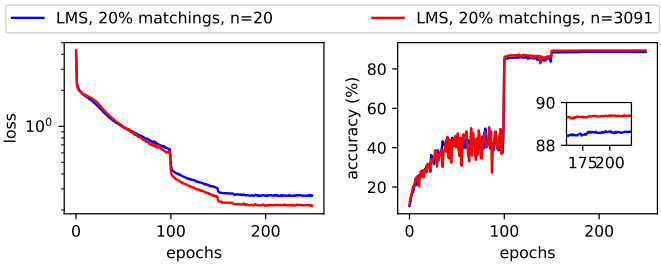


Fig. 7. Varying n: Communication time in wireless network

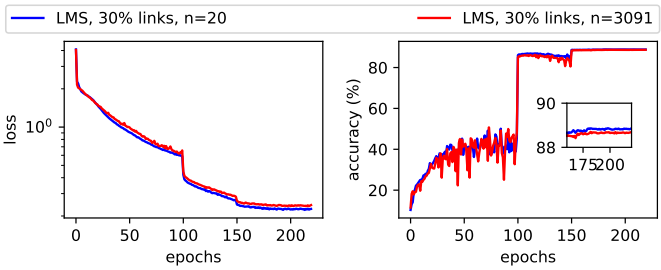


Fig. 8. Varying n: Energy consumption in wireless network

C. Additional Experiments

We have conducted additional experiments to explore other configurations and settings.

1) Sensitivity to n: In order to understand the impact of the number n of candidate Laplacian matrices on the performance of LMS, we test LMS under another configuration that leads to $n = 20$ candidate Laplacian matrices. We choose $n = 20$ because out of the original 3091 candidate matrices, the top 20 are sampled with a total probability of at least 0.99 across all the evaluated scenarios, as shown

[7]This is because LMS allows different links to have heterogeneous weights, as opposed to the identical weight used in MATCHA.

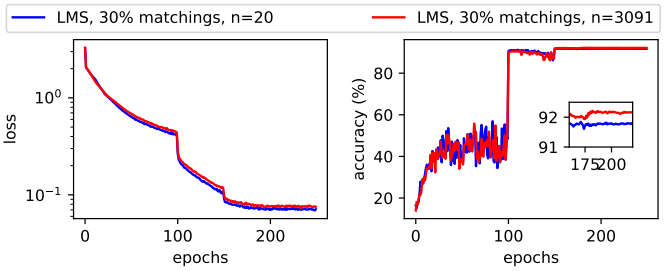


Fig. 9. Varying n: Communication time in wired network

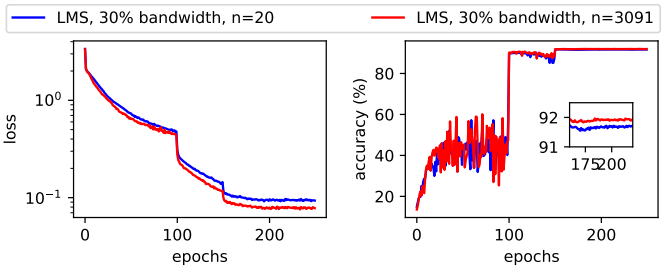


Fig. 10. Varying n: Bandwidth consumption in wired network

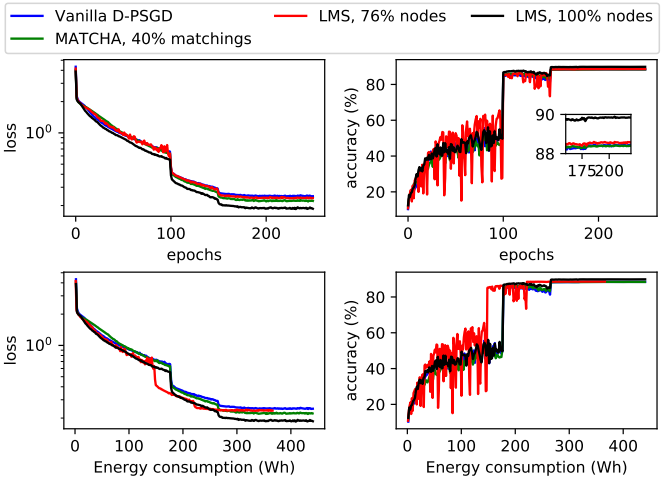


Fig. 11. Energy consumption under broadcast communication

in Fig. 6, suggesting that $n = 20$ will almost suffice. To validate this intuition, we configure LMS to construct (in addition to the solution to (21)) 9 candidates by Algorithm 1 according to $\mathcal{C}_g = \{0.1, 0.2, \ldots, 0.9\}$, and 10 candidates by Algorithm 2 according to 10 random instances of q generated as in Section V-A4.

We repeat the experiments in Fig. 2–5 under the new configuration, with results in Fig. 7–10. The plots of loss/accuracy vs. cost (#activated matchings, energy consumption, or bandwidth consumption) look similar to Fig. 7–10 and are thus omitted for conciseness. The results show that while sampling from a larger set of candidate Laplacian matrices ($n = 3091$) can lead to a slightly better model, a similar performance (with $< 1\%$ degradation in accuracy) can be achieved by sampling from a much smaller set of candidates ($n = 20$). These experiments demonstrate the robustness of LMS to the value of n.

2) Broadcast communication: Previously we have assumed unicast communication in the wireless setting, indicated by measuring the energy consumption by (7). Now we evaluate the case of broadcast communication. Recall that under broadcast communication, the cost model becomes $c(\boldsymbol{L}^{(k)}) := \sum_{i=1}^{m} c_i(\boldsymbol{L}^{(k)})$ with $c_i(\boldsymbol{L}^{(k)})$ defined as in (8). We set $c_i^a = 0.0003342$ (Wh) and $c_i^b = 0.0009129$ (Wh) as in Section V-A3. Under this model, the cost incurred in an iteration is determined by the number of *activated nodes* that share their parameter vectors with neighbors. We note that MATCHA does not support constraining the expected number of activated nodes (as it can only constrain the expected number of activated matchings). Nevertheless, we have tuned MATCHA's parameter to achieve the best performance at convergence and then configured the other algorithms accordingly to facilitate comparison, as explained in Section V-A5. In this case, MATCHA almost always activates all the nodes, and hence the corresponding version of Periodic with the same per-iteration cost coincides with Vanilla D-PSGD.

The results, given in Fig. 11, suggest a similar conclusion as before: LMS can not only achieve a lower loss and a higher accuracy at the same energy consumption, but also reduce the energy consumption by 24% compared to the best-performing benchmark while yielding a model of similar quality.

VI. Conclusion

We aimed to improve the communication efficiency of decentralized learning by carefully designing the communication patterns between nodes. Based on an existing convergence analysis, we formulated the design problem as a bilevel optimization that strives to achieve the maximum convergence rate and the optimal tradeoff with the cost per iteration. Based on this formulation, we developed a suite of efficient algorithms that jointly design both how nodes should communicate with each other and how much weights the communicated model parameters should carry in parameter aggregation. Our design achieves a guaranteed performance in minimizing the total cost till convergence in the special case of additive homogeneous communication costs, and at least 24–50% cost saving compared to existing designs in experiments based on a variety of network settings and cost models, without compromising the quality of the trained model. Our solution has broad applicability by adopting a general cost model that can be customized for various application scenarios.

References

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

[2] P. Kairouz *et al.*, *Advances and Open Problems in Federated Learning*. Now Foundations and Trends, 2021.

[3] "Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[4] "Google Assistant using federated learning on Android to improve 'Hey Google' accuracy," https://9to5google.com/2021/03/26/google-assistant-hotword-federated-learning/.

[5] "Federated Learning of Cohorts (FLoC)," https://github.com/WICG/floc.

[6] "Using Federated Learning to Improve Brave's On-Device Recommendations While Protecting Your Privacy," https://brave.com/federated-learning/.

[7] A. Koloskova, T. Lin, S. U. Stich, and M. Jagg, "Decentralized deep learning with arbitrary communication compression," in *The International Conference on Learning Representations (ICLR)*, 2020.

[8] Y. Lu and C. D. Sa, "Moniqua: Modulo quantized communication in decentralized SGD," in *International Conference on Machine Learning (ICML)*, 2020.

[9] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[10] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Systems for ML*, 2019.

[11] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, 2019.

[12] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[13] 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 5336–5346.

[14] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," in *NeurIPS Workshop on Federated Learning*, 2019. [Online]. Available: https://arxiv.org/abs/1905.09435

[15] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "d^2: Decentralized training over decentralized data," in *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

[16] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proceedings of the 35th International Conference on Machine Learning, ICML*, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 3049–3058.

[17] Y. Lu and C. D. Sa, "Optimal complexity in decentralized training," in *International Conference on Machine Learning (ICML)*, 2021.

[18] N. Singh, D. Data, J. George, and S. Diggavi, "SPARQ-SGD: Event-triggered and compressed communication in decentralized optimization," in *IEEE CDC*, 2020.

[19] ——, "SQuARM-SGD: Communication-efficient momentum SGD for decentralized optimization," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 954–969, 2021.

[20] J. Wang, A. K. Sahu, G. Joshi, and S. Kar, "Exploring the error-runtime trade-off in decentralized optimization," in *Asilomar Conference on Signals, Systems & Computers*, 2020.

[21] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, pp. 65–78, September 2004.

[22] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," in *IEEE Transactions on Information Theory*, vol. 52, 2006.

[23] J. M. Hendrickx, R. M. Jungers, A. Olshevsky, and G. Vankeerberghen, "Graph diameter, eigenvalues, and minimum-time consensus," *Automatica*, pp. 635–640, 2014.

[24] C.-K. Ko, "On matrix factorization and scheduling for finite-time average-consensus," Ph.D. dissertation, California Institute of Technology, 2010.

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

[26] B. Bollobás, *Modern Graph Theory*, ser. Graduate texts in mathematics. Springer, 2013.

[27] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, "Communication-efficient device scheduling for federated learning using stochastic optimization," in *IEEE INFOCOM*, 2022, pp. 1449–1458.

[28] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *ICML*, 2020.

[29] O. Toker and H. Ozbay, "On the NP-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback," in *Proceedings of 1995 American Control Conference*, vol. 4, 1995, pp. 2525–2526 vol.4.

[30] H. Jiang, T. Kathuria, Y. T. Lee, S. Padmanabhan, and Z. Song, "A faster interior point method for semidefinite programming," in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, 2020, pp. 910–918.

[31] R. M. Karp, *Reducibility among Combinatorial Problems*, ser. Complexity of Computer Computations. Springer, 1972.

[32] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *ACM STOC*, 2008.

[33] M. Rudelson and R. Vershynin, "Sampling from large matrices: An approach through geometric functional analysis," *Journal of the ACM*, vol. 54, no. 4, p. 21–es, July 2007.

[34] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *SIGCOMM*, 2004.

[35] "The Internet Topology Zoo," http://www.topology-zoo.org/dataset.html.

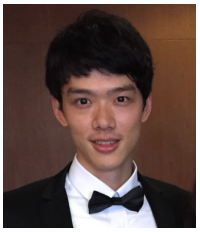
[36] X. Qiu, T. Parcollet, J. Fernandez-Marques, P. P. B. Gusmao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A first look into the carbon footprint of federated learning," 2021. [Online]. Available: https://arxiv.org/abs/2102.07627

[37] "SpeedTest," https://www.speedtest.net/.

[38] "Source code for MATCHA," https://github.com/JYWa/MATCHA.

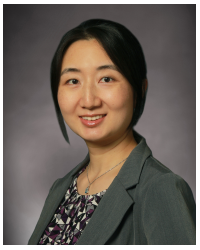
[39] "Source code for LMS," https://github.com/cuc496/LMS.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

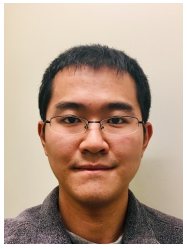


Cho-Chun Chiu (S'20) received the B.S. degree in Space Science and Engineering from National Central University and M.S. in Mechanical Engineering from National Taiwan University. He is a Ph.D. student in Computer Science and Engineering at the Pennsylvania State University, advised by Prof. Ting He. His research interest includes computer networking, network security, differential privacy, and federated learning.

Xusheng Zhang (S'22) received the B.S. degree in computer science and the B.S. degree in mathematics from University of Illinois at Urbana-Champaign in 2017. He is a Ph.D. student in Computer Science and Engineering at the Pennsylvania State University, advised by Prof. Antonio Blanca. His research involves approximation algorithm, Markov chains, computer networking, and computational geometry.



Ting He (SM'13) received the Ph.D. degree in electrical and computer engineering from Cornell University. Dr. He is an Associate Professor in the School of Electrical Engineering and Computer Science at the Pennsylvania State University, University Park, PA. Her interests reside at the intersection of computer networking, performance evaluation, and machine learning. Dr. He has served as Associate Editor for IEEE Transactions on Communications and IEEE/ACM Transactions on Networking, TPC Co-Chair of IEEE ICCCN, Area TPC Chair of IEEE INFOCOM, and TPC member of many international conferences. She received multiple awards from IBM and ITA, and paper awards from IEEE Communications Society, ICDCS, SIGMETRICS, and ICASSP.



Shiqiang Wang (S'13–M'15) is a Research Staff Member at IBM T. J. Watson Research Center, NY, USA. He received his Ph.D. from Imperial College London, United Kingdom, in 2015. His current research focuses on the intersection of distributed computing, machine learning, networking, and optimization, with a broad range of applications including data analytics, edge-based artificial intelligence (Edge AI), Internet of Things (IoT), and future wireless systems. He has made foundational contributions to edge computing and federated learning that generated both academic and industrial impact. Dr. Wang serves as an associate editor of the IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems, and IEEE Transactions on Computational Social Systems. He received the IEEE Communications Society (ComSoc) Leonard G. Abraham Prize in 2021, IEEE ComSoc Best Young Professional Award in Industry in 2021, IBM Outstanding Technical Achievement Awards (OTAA) in 2019, 2021, and 2022, multiple Invention Achievement Awards from IBM since 2016, Best Paper Finalist of the IEEE International Conference on Image Processing (ICIP) 2019, and Best Student Paper Award of the Network and Information Sciences International Technology Alliance (NIS-ITA) in 2015.

Ananthram Swami is with the US Army's DEVCOM Army Research Laboratory as the Army's Senior Research Scientist (ST) for Network Science. He received the B.Tech. degree from IIT-Bombay; the M.S. degree from Rice University, and the Ph.D. degree from the University of Southern California (USC), all in Electrical Engineering. Prior to joining ARL, he held positions with Unocal Corporation, USC, CS-3 and Malgudi Systems. He was a Statistical Consultant to the California Lottery, developed a MATLAB-based toolbox for non-Gaussian signal processing, has held visiting faculty positions at INP, Toulouse, and at Imperial College, London. Swami's work is in the broad area of network science, including communication and information networks and cyber security. Recent awards include a 2018 IEEE ComSoc MILCOM Technical Achievement Award and a 2017 Presidential Rank Award (Meritorious). He is an ARL Fellow and a Fellow of the IEEE.

APPENDIX

Proof of Lemma III.4. By its definition in (16), p will be upper-bounded by

$$1-\frac{\mathbb{E}[\|\boldsymbol{X}(\boldsymbol{W}-\boldsymbol{J})\|_F^2]}{\|\boldsymbol{X}(\boldsymbol{I}-\boldsymbol{J})\|_F^2} \leq 1-\frac{\|\boldsymbol{X}\cdot\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]\|_F^2}{\|\boldsymbol{X}(\boldsymbol{I}-\boldsymbol{J})\|_F^2} \tag{31}$$

for any $\boldsymbol{X}\neq\boldsymbol{0}$, where (31) is due to the convexity of $\|\cdot\|_F^2$. In particular, consider

$$\boldsymbol{X} := \begin{bmatrix} \text{sign}(\lambda^*)\boldsymbol{q}^{*\top} \\ \boldsymbol{0}_{(d-1)\times m} \end{bmatrix}, \tag{32}$$

where λ^* is th eigenvalue of $\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]$ with the largest absolute value, $\boldsymbol{q}^*$ the corresponding eigenvector, and $\boldsymbol{0}_{(d-1)\times m}$ the $(d-1)\times m$ matrix of zeros (d denotes the dimension of each parameter vector $\boldsymbol{x}_i$). For this $\boldsymbol{X}$, we have that

$$\begin{aligned}\|\boldsymbol{X}\cdot\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]\|_F^2 &= \|\begin{bmatrix} |\lambda^*|\boldsymbol{q}^{*\top} \\ \boldsymbol{0}_{(d-1)\times m} \end{bmatrix}\|_F^2 = \lambda^{*2}\|\boldsymbol{q}^{*\top}\|^2 \\ &= \|\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]\|^2, \end{aligned}\tag{33}$$

where the last step is because $\boldsymbol{q}^*$ is a unit vector and $|\lambda^*| = \|\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]\|$ by definition. Moreover,

$$\|\boldsymbol{X}(\boldsymbol{I}-\boldsymbol{J})\|_F^2 = \sum_{i=1}^m (q_i^*-\overline{q^*})^2 = 1-m\overline{q^*}^2, \tag{34}$$

where $\overline{q^*} := \frac{1}{m}\sum_{i=1}^m q_i^*$. Plugging (33)–(34) into (31) yields

$$p \leq 1-\frac{\|\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]\|^2}{1-m\overline{q^*}^2} \leq 1-\rho, \tag{35}$$

where the last inequality is because $1-m\overline{q^*}^2 \leq 1$ and $\|\mathbb{E}[\boldsymbol{W}-\boldsymbol{J}]\|^2 = \|(\mathbb{E}[\boldsymbol{W}^\top]-\boldsymbol{J})(\mathbb{E}[\boldsymbol{W}]-\boldsymbol{J})\| = \|\mathbb{E}[\boldsymbol{W}^\top\boldsymbol{W}]-\boldsymbol{J}\mathbb{E}[\boldsymbol{W}]-\mathbb{E}[\boldsymbol{W}^\top]\boldsymbol{J}+\boldsymbol{J}^2\| = \|\mathbb{E}[\boldsymbol{W}^\top\boldsymbol{W}]-\boldsymbol{J}\| = \rho$, due to the fact that $\boldsymbol{J}\mathbb{E}[\boldsymbol{W}] = \mathbb{E}[\boldsymbol{W}^\top]\boldsymbol{J} = \boldsymbol{J}^2 = \boldsymbol{J}$. □

Proof of Lemma III.5. Let $(\mathcal{L}^*, \boldsymbol{p}^*)$ denote the optimal design that achieves the minimum expected total cost E^*, and ρ^* be the corresponding spectral norm as defined in (9). If the budget is set to be $C = \frac{E^*}{K(\rho^*)}$, then $(\mathcal{L}^*, \boldsymbol{p}^*)$ will be a feasible solution to the lower-level optimization, and hence the optimal solution $(\mathcal{L}(C), \boldsymbol{p}(C))$ to the lower-level optimization would achieve a spectral norm of $\rho(C) \leq \rho^*$, and thus require a number of iterations of $K(\rho(C)) \leq K(\rho^*)$. Therefore, under the design $(\mathcal{L}(C), \boldsymbol{p}(C))$, the expected total cost $CK(\rho(C))$ will be upper-bounded by E^*. The design $(\mathcal{L}^o, \boldsymbol{p}^o)$ obtained by further minimizing $CK(\rho(C))$ over C will thus achieve an expected total cost that is no greater than E^*. □

Proof of Lemma IV.1. If $\boldsymbol{L}_{(i,j)}$ denotes the unweighted Laplacian matrix of the single-link graph $G(V, \{(i,j)\})$, then by definition

$$\boldsymbol{L} := \boldsymbol{B}\,\text{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top = \sum_{(i,j)\in E} \alpha_{(i,j)}\boldsymbol{L}_{(i,j)}. \tag{36}$$

Computing the trace yields

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{L}) = \mathrm{Tr}(\sum_{(i,j)\in E}\alpha_{(i,j)}\boldsymbol{L}_{(i,j)}) &= \sum_{(i,j)\in E}\alpha_{(i,j)}\mathrm{Tr}(\boldsymbol{L}_{(i,j)}) \\ &= 2\sum_{(i,j)\in E}\alpha_{(i,j)}, \end{aligned} \tag{37}$$

where the last equality is because $\mathrm{Tr}(\boldsymbol{L}_{(i,j)}) = 2$. Moreover, since $\mathrm{Tr}(\boldsymbol{L}) = \sum_{i=1}^{m}\lambda_i(\boldsymbol{L})$, we have

$$2\sum_{(i,j)\in E}\alpha_{(i,j)} = \sum_{i=1}^{m}\lambda_i(\boldsymbol{L}). \tag{38}$$

By Lemma IV.2, $\rho = \tilde{\rho}^2 < 1$ implies that $\lambda_m(\boldsymbol{L}) < 2$. Hence,

$$2\sum_{(i,j)\in E}\alpha_{(i,j)} \le m\lambda_m(\boldsymbol{L}) < 2m, \tag{39}$$

which implies $\alpha_{(i,j)} < m$ for all $(i,j) \in E$ as $\alpha_{(i,j)} \ge 0$. □

Proof of Lemma IV.2. Let $\boldsymbol{W} := \boldsymbol{I} - \boldsymbol{L}$ and $\boldsymbol{u}_i$ be the eigenvector of $\boldsymbol{L}$ corresponding to $\lambda_i(\boldsymbol{L})$. We have $\boldsymbol{W}\boldsymbol{u}_i = (\boldsymbol{I}-\boldsymbol{L})\boldsymbol{u}_i = (1-\lambda_i(\boldsymbol{L}))\boldsymbol{u}_i$. Thus, the i-th smallest eigenvalue of $\boldsymbol{W}$ satisfies $\lambda_i(\boldsymbol{W}) = 1 - \lambda_{m-i+1}(\boldsymbol{L})$. Therefore, based on the eigendecomposition $\boldsymbol{L} = \boldsymbol{Q}\,\mathrm{diag}(\lambda_1(\boldsymbol{L}),\ldots,\lambda_m(\boldsymbol{L}))\boldsymbol{Q}^\top$, we have

$$\begin{aligned}\tilde{\rho} = \|\boldsymbol{Q}&\begin{bmatrix}1-\lambda_1(\boldsymbol{L}) & & \\ & 1-\lambda_2(\boldsymbol{L}) & \\ & & \ddots\end{bmatrix}\boldsymbol{Q}^\top \\ &- \boldsymbol{Q}\begin{bmatrix}1 & & \\ & 0 & \\ & & \ddots\end{bmatrix}\boldsymbol{Q}^\top\|, \end{aligned} \tag{40}$$

This is because by definition, $\lambda_1(\boldsymbol{L}) = 0$, corresponding to the eigenvector $\boldsymbol{u}_1 = \left[\frac{1}{\sqrt{m}},...,\frac{1}{\sqrt{m}}\right]^\top$, and $\boldsymbol{J} = \boldsymbol{u}_1\boldsymbol{u}_1^\top = \boldsymbol{Q}\,\mathrm{diag}(1,0,...,0)\boldsymbol{Q}^\top$. Thus, we can express $\tilde{\rho}$ as

$$\begin{aligned}\tilde{\rho} =& \|\boldsymbol{Q}\begin{bmatrix}0 & & & \\ & 1-\lambda_2(\boldsymbol{L}) & & \\ & & \ddots & \\ & & & 1-\lambda_m(\boldsymbol{L})\end{bmatrix}\boldsymbol{Q}^\top\| \\ =& \max\{|1-\lambda_2(\boldsymbol{L})|, |1-\lambda_m(\boldsymbol{L})|\} \\ =& \max\{1-\lambda_2(\boldsymbol{L}), \lambda_m(\boldsymbol{L})-1\}, \end{aligned} \tag{41}$$

where (41) is implied by $1-\lambda_2(\boldsymbol{L}) \ge 1-\lambda_m(\boldsymbol{L})$. □

Proof of Theorem IV.4. Let $\tilde{\rho}_L := \|\boldsymbol{I} - \boldsymbol{L} - \boldsymbol{J}\|$ for $\boldsymbol{L}$ computed in line 1 of Algorithm 2. By construction, $C_E = \frac{c_r m\log m}{(\epsilon\delta)^2}$, and $\epsilon\delta \in (\frac{1}{\sqrt{m}}, 1]$. By Lemma IV.3, $\boldsymbol{L}_H$ satisfies $\lambda_2(\boldsymbol{L}_H) \ge (1-\delta\epsilon)\lambda_2(\boldsymbol{L})$ and $\lambda_m(\boldsymbol{L}_H) \le (1+\delta\epsilon)\lambda_m(\boldsymbol{L})$ with probability at least $1/2$.

Case 1: If $\lambda_2(\boldsymbol{L}) \le 1 \le \lambda_m(\boldsymbol{L})$, then

$$\delta = \min\left\{\frac{1-\lambda_2(\boldsymbol{L})}{\lambda_2(\boldsymbol{L})}, \frac{\lambda_m(\boldsymbol{L})-1}{\lambda_m(\boldsymbol{L})}\right\}. \tag{42}$$

Since $\lambda_2(\boldsymbol{L}_H) \ge (1-\delta\epsilon)\lambda_2(\boldsymbol{L})$ and $\delta \le (1-\lambda_2(\boldsymbol{L}))/\lambda_2(\boldsymbol{L})$,

$$\begin{aligned}1-\lambda_2(\boldsymbol{L}_H) &\le 1 - \left(1 - \frac{(1-\lambda_2(\boldsymbol{L}))}{\lambda_2(\boldsymbol{L})}\cdot\epsilon\right)\lambda_2(\boldsymbol{L}) \\ &= (1+\epsilon)(1-\lambda_2(\boldsymbol{L})). \end{aligned} \tag{43}$$

Similarly, since $\lambda_m(\boldsymbol{L}_H) \le (1+\delta\epsilon)\lambda_m(\boldsymbol{L})$ and $\delta \le (\lambda_m(\boldsymbol{L})-1)/\lambda_m(\boldsymbol{L})$,

$$\begin{aligned}\lambda_m(\boldsymbol{L}_H) - 1 &\le \left(1 + \frac{(\lambda_m(\boldsymbol{L})-1)}{\lambda_m(\boldsymbol{L})}\cdot\epsilon\right)\lambda_m(\boldsymbol{L}) - 1 \\ &= (1+\epsilon)(\lambda_m(\boldsymbol{L})-1). \end{aligned} \tag{44}$$

Combining (43)–(44) and Lemma IV.2 implies that $\tilde{\rho}_H \le (1+\epsilon)\tilde{\rho}_L$.

Case 2: If $0 < \lambda_2(\boldsymbol{L}) \le \lambda_m(\boldsymbol{L}) < 1$, then

$$\delta = \frac{1-\lambda_m(\boldsymbol{L})}{\lambda_m(\boldsymbol{L})} \le \frac{1-\lambda_2(\boldsymbol{L})}{\lambda_2(\boldsymbol{L})}. \tag{45}$$

Thus, (43) holds, and $\lambda_m(\boldsymbol{L}_H) \le (1+\delta\epsilon)\lambda_m(\boldsymbol{L})$ implies that

$$\begin{aligned}\lambda_m(\boldsymbol{L}_H) - 1 &\le (1 + \frac{(1-\lambda_m(\boldsymbol{L}))}{\lambda_m(\boldsymbol{L})}\cdot\epsilon)\lambda_m(\boldsymbol{L}) - 1 \\ &= (1-\epsilon)(\lambda_m(\boldsymbol{L})-1) \le (1+\epsilon)(1-\lambda_2(\boldsymbol{L})). \end{aligned} \tag{46}$$

Combining (43), (46), and Lemma IV.2 implies that $\tilde{\rho}_H \le (1+\epsilon)\tilde{\rho}_L$.

Case 3: If $1 < \lambda_2(\boldsymbol{L}) \le \lambda_m(\boldsymbol{L})$, then

$$\delta = \frac{\lambda_2(\boldsymbol{L})-1}{\lambda_2(\boldsymbol{L})} \le \frac{\lambda_m(\boldsymbol{L})-1}{\lambda_m(\boldsymbol{L})}. \tag{47}$$

Thus, (44) holds, and $\lambda_2(\boldsymbol{L}_H) \ge (1-\delta\epsilon)\lambda_2(\boldsymbol{L})$ implies that

$$\begin{aligned}1-\lambda_2(\boldsymbol{L}_H) &\le 1 - (1 - \frac{(\lambda_2(\boldsymbol{L})-1)}{\lambda_2(\boldsymbol{L})}\cdot\epsilon)\lambda_2(\boldsymbol{L}) \\ &= (1-\epsilon)(1-\lambda_2(\boldsymbol{L})) \le (1+\epsilon)(\lambda_m(\boldsymbol{L})-1). \end{aligned} \tag{48}$$

Combining (44), (48), and Lemma IV.2 implies that $\tilde{\rho}_H \le (1+\epsilon)\tilde{\rho}_L$.

Thus, with probability at least $1/2$, $\tilde{\rho}_H \le (1+\epsilon)\tilde{\rho}_L$. Moreover, as $\tilde{\rho}_L$ is the optimal objective value of (20) without the constraint $\sum_{(i,j)\in E}\beta_{(i,j)} \le C_E$, it must satisfy $\tilde{\rho}_L \le \tilde{\rho}^*$, which completes the proof. □

Proof of Theorem IV.5. Let $\boldsymbol{L}_H$ be the output of Algorithm 2 for $q = C_E^*$. The proof of Theorem IV.4 implies that with probability $\ge 1/2$,

$$\begin{aligned}\rho_H := \|\boldsymbol{I} - \boldsymbol{L}_H - \boldsymbol{J}\|^2 &\le (1+\epsilon)^2\tilde{\rho}_L^2 \qquad (49)\\ &\le (1+\epsilon)^2\rho^*, \qquad (50)\end{aligned}$$

where $\tilde{\rho}_L$ is the optimal objective value of (21) and (50) is because $\tilde{\rho}_L^2 \le \rho^*$ as shown later. Since $C_E^* \in \mathcal{Q}$, we have $\boldsymbol{L}_H \in \mathcal{L}$. Under budget $C^* \in \mathcal{C}$, a design that samples $\boldsymbol{L}_H$ with probability one will achieve an expected total cost of

$$\begin{aligned}K(\rho_H)c(\boldsymbol{L}_H) &\le K(\rho_H)C^* \\ &\le K\left((1+\epsilon)^2\rho^*\right)C^*, \end{aligned} \tag{51}$$

where (51) is because $K(\cdot)$ in (11) is an increasing function. The design optimized over all the budgets in $\mathcal{C}$ and all the sampling distributions will thus achieve a cost that is no worse.

We now show $\tilde{\rho}_L^2 \le \rho^*$. Let $\boldsymbol{L}^*$ denote the random Laplacian matrix distributed according to the optimal design. By (9),

$$\rho^* = \|\mathbb{E}[(\boldsymbol{I}-\boldsymbol{L}^*)^\top(\boldsymbol{I}-\boldsymbol{L}^*)] - \boldsymbol{J}\|$$

$$= \|\mathbb{E}[(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})^\top(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})]\|$$
$$= \lambda_m\left(\mathbb{E}[(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})^\top(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})]\right), \tag{52}$$

where (52) is because $\mathbb{E}[(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})^\top(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})]$ is symmetric and positive semi-definite. For any random vector $\boldsymbol{v}\in\mathbb{R}^m$, since $\|\cdot\|^2$ is convex [40], Jensen's inequality implies that $\mathbb{E}[\boldsymbol{v}^\top\boldsymbol{v}]\geq\mathbb{E}[\boldsymbol{v}]^\top\mathbb{E}[\boldsymbol{v}]$. For any constant vector $\boldsymbol{u}\in\mathbb{R}^m$, plugging in $\boldsymbol{v}:=(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})\boldsymbol{u}$ yields

$$\boldsymbol{u}^\top\mathbb{E}\left[(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})^\top(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})\right]\boldsymbol{u}$$
$$\geq \boldsymbol{u}^\top(\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})^\top(\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})\boldsymbol{u}. \tag{53}$$

By the Courant-Fischer theorem [32], (53) implies

$$\lambda_m\left(\mathbb{E}[(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})^\top(\boldsymbol{I}-\boldsymbol{L}^*-\boldsymbol{J})]\right)$$
$$\geq \lambda_m\left((\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})^\top(\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})\right). \tag{54}$$

This together with (52) implies that

$$\rho^* \geq \lambda_m\left((\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})^\top(\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})\right)$$
$$= \|(\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})^\top(\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J})\|$$
$$= \|\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J}\|^2 \geq \tilde{\rho}_L^2, \tag{55}$$

where (55) is because $\mathbb{E}[\boldsymbol{L}^*]$ is a feasible solution to (21) and hence $\|\boldsymbol{I}-\mathbb{E}[\boldsymbol{L}^*]-\boldsymbol{J}\|\geq\tilde{\rho}_L$. □