ENERGY-EFFICIENT DECENTRALIZED LEARNING VIA GRAPH SPARSIFICATION

Xusheng Zhang, Cho-Chun Chiu, and Ting He

Pennsylvania State University, University Park, PA 16802, USA

ABSTRACT

This work aims at improving the energy efficiency of decentralized learning by optimizing the mixing matrix, which controls the communication demands during the learning process. Through rigorous analysis based on a state-of-the-art decentralized learning algorithm, the problem is formulated as a bi-level optimization, with the lower level solved by graph sparsification. A solution with guaranteed performance is proposed for the special case of fully-connected base topology and a greedy heuristic is proposed for the general case. Simulations based on real topology and dataset show that the proposed solution can lower the energy consumption at the busiest node by 54%–76% while maintaining the quality of the trained model.

1. INTRODUCTION

Learning from decentralized data [1] is an emerging machine learning paradigm that has found many applications [2]. Communication efficiency has been a major consideration in designing learning algorithms, as the cost in communicating model updates, e.g., communication time, bandwidth consumption, and energy consumption, dominates the total operation cost in many application scenarios [1]. Existing works on reducing this cost can be broadly classified into (i) model compression for reducing the cost per communication [3] and (ii) hyperparameter optimization for reducing the number of communications until convergence [4]. The two approaches are orthogonal and can be applied jointly.

In this work, we focus on hyperparameter optimization in the decentralized learning setting, where nodes communicate with neighbors according to a given base topology [5]. To this end, we adopt a recently proposed optimization framework from [6] that allows for systematic design of a critical hyperparameter in decentralized learning, the *mixing matrix*, to minimize a generally-defined cost measure. The choice of mixing matrix as the design parameter utilizes the observation from [7] that *not all the links are equally important for convergence*. Hence, instead of communicating over all the links at the same frequency as in most of the existing works [1, 4], communicating on different links with different frequencies can further improve the communication efficiency. However, the existing mixing matrix designs [7, 6] fall short at addressing a critical cost measure in wireless networks: *energy consumption at the busiest node*. Although energy consumption is considered in [6], its cost model only captures the total energy consumption over all the nodes. In this work, we address this gap based on a rigorous theoretical foundation.

1.1. Related Work

Decentralized learning algorithms. The standard algorithm for learning under a fully decentralized architecture was an algorithm called Decentralized Parallel Stochastic Gradient Descent (D-PSGD) [5], which was shown to achieve the same computational complexity but a lower communication complexity than training via a central server. Since then a number of improvements have been developed, e.g., [8], but these works only focused on the number of iterations.

Communication cost reduction. One line of works tried to reduce the amount of data per communication through model compression, e.g., [3]. Another line of works reduced the frequency of communications, e.g., [4]. Later works [9, 10] started to combine model compression and infrequent communications. Recently, it was recognized that better tradeoffs can be achieved by activating subsets of links, e.g., via event-based triggers [9, 10] or predetermined mixing matrices [7, 6]. Our work is closest to [7, 6] by also designing the mixing matrix, but we address a different objective of maximum per-node energy consumption.

Mixing matrix design. Mixing matrix design has been considered in the classical problem of distributed averaging, e.g., [11, 12] designed a mixing matrix with the fastest convergence to ϵ-average and [13] designed a sequence of mixing matrices to achieve exact average in finite time. In contrast, fewer works have addressed the design of mixing matrices in decentralized learning, e.g., [7, 6]. We refer to [14] for a more complete overview of related works.

1.2. Summary of Contributions

We study the design of mixing matrix in decentralized learning with the following contributions:

1) Instead of considering the total energy consumption as in [6], our design aims at minimizing the energy consumption at the busiest node, leading to a more balanced load.

2) Instead of using a heuristic objective as in [7] or a partially justified objective as in [6], we use a fully theoretically-justified design objective, which enables a new approach for

mixing matrix design based on graph sparsification.

3) Based on the new approach, we propose an algorithm with guaranteed performance for a special case and a greedy heuristic for the general case. Our solution achieves 54%–76% lower energy consumption at the busiest node while producing a model of the same quality as the best-performing benchmark in simulations based on real topology and dataset.

Roadmap. Section 2 formulates our problem, Section 3 presents the proposed solution, Section 4 evaluates it against benchmarks, and Section 5 concludes the paper. **Proofs and additional evaluation results are provided in [14].**

2. BACKGROUND AND PROBLEM FORMULATION

2.1. Decentralized Learning Algorithm

Consider a network of m nodes connected through a *base topology* $G = (V, E)$ ($|V| = m$), where E defines the pairs of nodes that can directly communicate. Each node $i \in V$ has a local objective function $F_i(\boldsymbol{x})$ that depends on the parameter vector $\boldsymbol{x} \in \mathbb{R}^d$ and its local dataset. The goal is to minimize the global objective function $F(\boldsymbol{x}) := \frac{1}{m}\sum_{i=1}^m F_i(\boldsymbol{x})$.

We consider a state-of-the-art decentralized learning algorithm called D-PSGD [5]. 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. D-PSGD runs the following update in parallel at each node i:

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

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. To be consistent with the base topology, $W_{ij}^{(k)} \neq 0$ only if $(i,j) \in E$.

The convergence of this algorithm is guaranteed under the following assumptions:

(1) Each local objective function $F_i(\boldsymbol{x})$ is l-Lipschitz smooth, i.e.,[1] $\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{x}')\| \leq l\|\boldsymbol{x} - \boldsymbol{x}'\|$, $\forall i \in V$.

(2) There exist constants $M_1, \hat{\sigma}$ such that $\frac{1}{m}\sum_{i\in V}\mathbf{E}[\|g(\boldsymbol{x}_i;\xi_i) - \nabla F_i(\boldsymbol{x}_i)\|^2] \leq \hat{\sigma}^2 + \frac{M_1}{m}\sum_{i\in V}\|\nabla F(\boldsymbol{x}_i)\|^2$, $\forall \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{R}^d$.

(3) There exist constants $M_2, \hat{\zeta}$ such that $\frac{1}{m}\sum_{i\in V}\|\nabla F_i(\boldsymbol{x})\|^2 \leq \hat{\zeta}^2 + M_2\|\nabla F(\boldsymbol{x})\|^2, \forall \boldsymbol{x} \in \mathbb{R}^d$.

Theorem 2.1. [15, Theorem 2] Let $\boldsymbol{J} := \frac{1}{m}\mathbf{1}\mathbf{1}^\top$. Under assumptions (1)–(3), if there exist a constant $p \in (0, 1]$ such that the mixing matrices $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$, each being symmetric with each row/column summing to one[2], satisfy

$$\mathbf{E}[\|\boldsymbol{X}\boldsymbol{W}^{(k)} - \boldsymbol{X}\boldsymbol{J}\|_F^2] \leq (1-p)\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{J}\|_F^2, \tag{2}$$

for all $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$ and integer $k \geq 1$, 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 $\epsilon_0 > 0$ ($\overline{\boldsymbol{x}}^{(k)} := \frac{1}{m}\sum_{i=1}^m \boldsymbol{x}_i^{(k)}$) when the number of iterations reaches

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

Remark: The required number of iterations $K(p)$ depends on the mixing matrix only through the parameter p: the larger p, the smaller $K(p)$. Originally, [15, Theorem 2] only mandates (2) for the product of τ mixing matrices, but we consider the case of $\tau = 1$ for the tractability of mixing matrix design.

2.2. Mixing Matrix

As node i needs to send its parameter vector to node j in iteration k only if $W_{ij}^{(k)} \neq 0$, we can control the communications by designing the mixing matrix $\boldsymbol{W}^{(k)}$. To this end, we use $\boldsymbol{W}^{(k)} := \boldsymbol{I} - \boldsymbol{L}^{(k)}$, where $\boldsymbol{L}^{(k)}$ is the weighted Laplacian matrix [16] of the topology $G^{(k)} = (V, E^{(k)})$ *activated* in iteration k. Given the incidence matrix[3] $\boldsymbol{B}$ of the base topology G and a vector of *link weights* $\boldsymbol{\alpha}^{(k)}$, the Laplacian matrix $\boldsymbol{L}^{(k)}$ is given by $\boldsymbol{L}^{(k)} = \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^{(k)})\boldsymbol{B}^T$. The above reduces 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 *if and only if* $\alpha_{(i,j)}^{(k)} \neq 0$. This construction guarantees that $\boldsymbol{W}^{(k)}$ is symmetric with each row/column summing up to one.

2.3. Cost Model

We use $c(\boldsymbol{\alpha}^{(k)}) := (c_i(\boldsymbol{\alpha}^{(k)}))_{i=1}^m$ to denote the cost vector in an iteration when the link weight vector is $\boldsymbol{\alpha}^{(k)}$. We focus on the energy consumption at each node i, which contains two parts: (i) computation energy c_i^a for computing the local stochastic gradient and the local aggregation, and (ii) communication energy c_{ij}^b for sending the updated local parameter vector from node i to node j. Then the energy consumption at node i in iteration k is modeled as

$$c_i(\boldsymbol{\alpha}^{(k)}) := c_i^a + \sum_{j:(i,j)\in E} c_{ij}^b \mathbb{1}(\alpha_{(i,j)}^{(k)} \neq 0), \tag{3}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. This cost function models the basic scenario where all communications are point-to-point and independent. Other scenarios are left to future work.

2.4. Optimization Framework

To trade off between the cost per iteration and the convergence rate, we adopt a bi-level optimization framework:

[1] For a vector $\boldsymbol{a}$, $\|\boldsymbol{a}\|$ denotes the ℓ-2 norm. For a matrix $\boldsymbol{A}$, $\|\boldsymbol{A}\|$ denotes the spectral norm, and $\|\boldsymbol{A}\|_F$ denotes the Frobenius norm.

[2] Originally, [15, Theorem 2] had a stronger assumption that each mixing matrix is doubly stochastic, but we have verified that it suffices to have each row/column summing to one.

[3] Matrix $\boldsymbol{B}$ is a $|V| \times |E|$ matrix, defined as $B_{ij} = 1$ if link e_j starts at node i (under arbitrary link orientation), -1 if e_j ends at i, and 0 otherwise.

Lower-level optimization: design link weights $\boldsymbol{\alpha}$ to maximize the convergence rate (by maximizing p) under a given budget Δ on the maximum cost per node in each iteration, which results in a required number of iterations of $K(p_\Delta)$.
Upper-level optimization: design Δ to minimize the total maximum cost per node $\Delta \cdot K(p_\Delta)$.

3. MIXING MATRIX DESIGN VIA GRAPH SPARSIFICATION

As the upper-level optimization only involves one scalar decision variable, we will focus on the lower-level optimization.

3.1. Simplified Objective

The parameter p that minimizes the required number of iterations $K(p)$ is given by

$$p := \min_{\boldsymbol{X}\neq\boldsymbol{0}} \left(1 - \frac{\mathbf{E}[\|\boldsymbol{X}(\boldsymbol{W}-\boldsymbol{J})\|_F^2]}{\|\boldsymbol{X}(\boldsymbol{I}-\boldsymbol{J})\|_F^2}\right). \tag{4}$$

As (4) is not an explicit function of the mixing matrix, we first relate it to an equivalent quantity that is easier to handle. We can relate p to an explicit function of $\boldsymbol{W}$ as follows.

Lemma 3.1. For any randomized mixing matrix $\boldsymbol{W}$ that is symmetric with every row/column summing to one, p defined in (4) satisfies $p = 1 - \rho$ for $\rho := \|\mathbf{E}[\boldsymbol{W}^\top\boldsymbol{W}] - \boldsymbol{J}\|$.

Lemma 3.1 implies that the lower-level optimization should minimize ρ. While it is possible to formulate this minimization in terms of the link weights $\boldsymbol{\alpha}$, the resulting optimization problem, with a form similar to [6, (18)], will be intractable due to the presence of non-linear matrix inequality constraint. We thus further simplify the objective as follows.

Lemma 3.2. For any mixing matrix $\boldsymbol{W} := \boldsymbol{I} - \boldsymbol{L}$, where $\boldsymbol{L}$ is a randomized Laplacian matrix,

$$\rho \le \mathbf{E}\left[\|\boldsymbol{I}-\boldsymbol{L}-\boldsymbol{J}\|^2\right] \tag{5}$$
$$= \mathbf{E}\left[\max\left((1-\lambda_2(\boldsymbol{L}))^2, (1-\lambda_m(\boldsymbol{L}))^2\right)\right], \tag{6}$$

where $\lambda_i(\boldsymbol{L})$ denotes the i-th smallest eigenvalue of $\boldsymbol{L}$.

By Lemma 3.2, we relax the objective of the lower-level optimization to designing a randomized $\boldsymbol{\alpha}$ by solving

$$\min_{\boldsymbol{\alpha}} \quad \mathbf{E}\left[\|\boldsymbol{I}-\boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J}\|^2\right] \tag{7a}$$
$$\text{s.t.} \quad \mathbf{E}[c_i(\boldsymbol{\alpha})] \le \Delta, \quad \forall i \in V. \tag{7b}$$

3.2. Idea on Leveraging Graph Sparsification

We propose to solve the relaxed lower-level optimization (7) based on graph (spectral) sparsification. First, we compute the optimal link weight vector $\boldsymbol{\alpha}'$ without the budget constraint (7b) by solving the following optimization:

$$\min_{\boldsymbol{\alpha}} \ \tilde{\rho} \tag{8a}$$
$$\text{s.t.} \ -\tilde{\rho}\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \tilde{\rho}\boldsymbol{I}. \tag{8b}$$

Constraint (8b) ensures $\tilde{\rho} = \|\boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J}\|$ at the optimum, i.e., $\boldsymbol{\alpha}'$ minimizes $\|\boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J}\|$. Optimization (8) is a semi-definite programming (SDP) problem that can be solved in polynomial time by existing algorithms [17]. The vector $\boldsymbol{\alpha}'$ establishes a lower bound on the relaxed objective: if $\boldsymbol{\alpha}^*$ is the optimal randomized solution for (7), then $\mathbf{E}\left[\|\boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}^*)\boldsymbol{B}^\top - \boldsymbol{J}\|^2\right] \ge \|\boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}')\boldsymbol{B}^\top - \boldsymbol{J}\|^2$.

Then, we use a graph sparsification algorithm to sparsify the weighted graph with link weights $\boldsymbol{\alpha}'$ to satisfy the budget constraint. As $\|\boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}')\boldsymbol{B}^\top - \boldsymbol{J}\|^2 = \max\left((1 - \lambda_2(\boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}')\boldsymbol{B}^\top))^2,\ (1 - \lambda_m(\boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}')\boldsymbol{B}^\top))^2\right)$, and graph sparsification aims at preserving the original eigenvalues $\left(\lambda_i(\boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}')\boldsymbol{B}^\top)\right)_{i=1}^m$ [18], the sparsified link weight vector $\boldsymbol{\alpha}_s$ is expected to achieve an objective value $\|\boldsymbol{I} - \boldsymbol{B}\operatorname{diag}(\boldsymbol{\alpha}_s)\boldsymbol{B}^\top - \boldsymbol{J}\|^2$ that approximates the optimal for (7).

3.3. Algorithm Design

We now apply the above idea to develop algorithms for mixing matrix design.

3.3.1. Ramanujan-Graph-based Design for a Special Case

Consider the special case when the base topology G is a complete graph and all transmissions by a node have the same cost, i.e., $c_{ij}^b \equiv c_i^b$ for all j such that $(i,j) \in E$. Let $d := \min_{i\in V}\lfloor(\Delta - c_i^a)/c_i^b\rfloor$. Then any graph with degrees bounded by d satisfies the budget constraint. The complete graph has an ideal sparsifier known as *Ramanujan graph*. A d-regular graph H is a Ramanujan graph if all the non-zero eigenvalues of its Laplacian matrix $\boldsymbol{L}_H$ lie between $d - 2\sqrt{d-1}$ and $d + 2\sqrt{d-1}$ [19]. By assigning weight $1/d$ to every link of a Ramanujan graph H, we obtain a weighted graph H', whose Laplacian $\boldsymbol{L}_{H'}$ satisfies $\lambda_1(\boldsymbol{L}_{H'}) = 0$ and

$$1 - \frac{2\sqrt{d-1}}{d} \le \lambda_2(\boldsymbol{L}_{H'}) \le \cdots \le \lambda_m(\boldsymbol{L}_{H'}) \le 1 + \frac{2\sqrt{d-1}}{d}.$$

By Lemma 3.2, the deterministic mixing matrix $\boldsymbol{W}_{H'} := \boldsymbol{I} - \boldsymbol{L}_{H'}$ achieves a ρ-value $\rho_{H'}$ that satisfies

$$\rho_{H'} \le \max\left((1-\lambda_2(\boldsymbol{L}_{H'}))^2, (1-\lambda_m(\boldsymbol{L}_{H'}))^2\right)$$
$$\le \frac{4(d-1)}{d^2} = O\left(\frac{1}{d}\right) = O\left(\frac{1}{\Delta}\right).$$

Ramanujan graphs can be easily constructed by drawing random d-regular graphs until satisfying the Ramanujan definition [20]. By the result of [21], for $d \le m^{1/3}$, we can generate random d-regular graphs in polynomial time. Thus, the above method can efficiently construct a deterministic mixing matrix with guaranteed performance in solving the lower-level optimization for a given budget Δ such that $d \le m^{1/3}$.

3.3.2. Greedy Heuristic for General Case

For the general case, ideally we want to sparsify a weighted graph G' with link weights $\boldsymbol{\alpha}'$ such that the sparsified graph

with link weights $\boldsymbol{\alpha}_s$ will approximate the eigenvalues of G' while satisfying the constraint $c_i(\boldsymbol{\alpha}_s) \leq \Delta$ for each $i \in V$. While this remains an open problem for general graphs, we propose a greedy heuristic based on the intuition that the *importance of a link is reflected in its absolute weight*. Specifically, we will find the link (i, j) with the minimum absolute weight according to the solution to (8) such that the cost for either node i or node j exceeds the budget Δ, set $\alpha_{(i,j)} = 0$, and then find the next link by re-solving (8) under this additional constraint, until all the nodes satisfy the budget.

4. PERFORMANCE EVALUATION

We evaluate the proposed solution for the general case based on a real dataset and the topology of a real wireless network. We defer the evaluation in the special case to [14].

Experiment setting: 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. We use the topology of Roofnet [22] at data rate 1 Mbps as the base topology, which contains 33 nodes and 187 links. To evaluate the cost, we set the computation energy as $c_i^a = 0.0003342$ (Wh) and the communication energy as $c_{ij}^b = 0.0138$ (Wh) based on our parameters and the parameters from [23][4]. Following [6], we set the learning rate as 0.8 at the beginning and reduce it by 10X after 100, 150, 180, 200 epochs, and the mini-batch size to 32.

Benchmarks: We compare the proposed solution with with four benchmarks: 'Vanilla D-PSGD' [5] where all the neighbors communicate in all the iterations, 'Periodic' where all the neighbors communicate periodically, 'MATCHA' [7] which was designed to minimize training time, and Algorithm 1 in [6] ('Greedy total') for the cost model (3) which was designed to minimize the total energy consumption[5]. In 'Vanilla D-PSGD', 'Periodic', and 'MATCHA', identical weights are assigned to every activated link, whereas in 'Greedy total' and the proposed algorithm, heterogeneous link weights are parts of the designs. We first tune MATCHA to minimize its loss at convergence, and then tune the other benchmarks to activate the same number of links on the average. We evaluate two versions of the proposed algorithm ('Greedy per-node'): one with the same maximum energy consumption per node as the best-performing benchmark (leading to a budget that amounts to 55% of maximum degree) and the other with the same accuracy as the best-performing benchmark at convergence (leading to a budget that amounts to 25% of maximum degree).

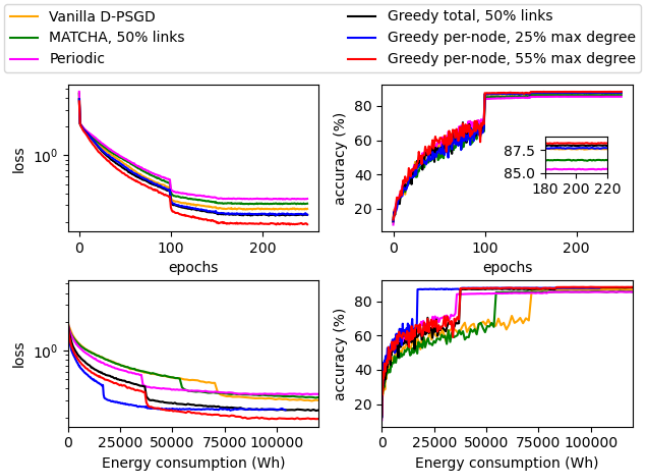

Fig. 1. Training loss and testing accuracy for decentralized learning over Roofnet.

Results: Fig. 1 shows the loss and accuracy of the trained model, with respect to both the epochs and the maximum energy consumption per node. We see that: (i) instead of activating all the links as in 'Vanilla D-PSGD', it is possible to activate fewer (weighted) links without degrading the quality of the trained model; (ii) different ways of selecting the links to activate lead to different quality-cost tradeoffs; (iii) the algorithm designed to optimize the total energy consumption ('Greedy total') performs the best among the benchmarks; (iv) however, by balancing the energy consumption across nodes, the proposed algorithm ('Greedy per-node') can achieve either a better loss/accuracy at the same maximum energy consumption per node, or a lower maximum energy consumption per node at the same loss and accuracy. In particular, the proposed algorithm (at 25% maximum degree) can save 54% energy at the busiest node compared to the best-performing benchmark ('Greedy total') and 76% compared to 'Vanilla D-PSGD', while producing a model of the same quality. Meanwhile, the proposed algorithm also saves 41–71% of the total energy consumption compared to the benchmarks, as shown in [14, Table 1].

5. CONCLUSION

Based on an explicit characterization of how the mixing matrix affects the convergence rate in decentralized learning, we proposed a bi-level optimization for mixing matrix design, with the lower level solved by graph sparsification. This enabled us to develop a solution with guaranteed performance for a special case and a heuristic for the general case. Our solution greatly reduced the energy consumption at the busiest node while maintaining the quality of the trained model.

[4]Our model size is $S = 2.3$MB, batch size is 32, and processing speed is 8ms per sample. Assuming 1Mbps links and TX2 as the hardware, whose power is 4.7W during computation and 1.35W during communication [23], we estimate the computation energy by $c_i^a = 4.7 * 32 * 0.008/3600 \approx 0.0003342$Wh, and the communication energy with each neighbor by $c_{ij}^b = 2 * 1.35 * S * 8/1/3600$Wh, where the multiplication by 2 is because this testbed uses WiFi, which is half-duplex.

[5]While the final solution in [6] was randomized over a set of mixing matrices, we only use the deterministic design by Algorithm 1 for a fair comparison, as the same randomization can be applied to the proposed solution.

6. REFERENCES

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

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

[3] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jagg, "Decentralized deep learning with arbitrary communication compression," in *The International Conference on Learning Representations (ICLR)*, 2020.

[4] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin 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.

[5] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji 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.

[6] Cho-Chun Chiu, Xusheng Zhang, Ting He, Shiqiang Wang, and Ananthram Swami, "Laplacian matrix sampling for communication- efficient decentralized learning," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 887–901, 2023.

[7] 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.

[8] Yucheng Lu and Christopher De Sa, "Optimal complexity in decentralized training," in *International Conference on Machine Learning (ICML)*, 2021.

[9] Navjot Singh, Deepesh Data, Jemin George, and Suhas Diggavi, "SPARQ-SGD: Event-triggered and compressed communication in decentralized optimization," in *IEEE CDC*, 2020.

[10] Navjot Singh, Deepesh Data, Jemin George, and Suhas Diggavi, "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.

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

[12] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah, "Randomized gossip algorithms," in *IEEE Transactions on Information Theory*, 2006, vol. 52.

[13] Julien M Hendrickx, Raphaël M Jungers, Alexander Olshevsky, and Guillaume Vankeerberghen, "Graph diameter, eigenvalues, and minimum-time consensus," *Automatica*, pp. 635–640, 2014.

[14] Xusheng Zhang, Cho-Chun Chiu, and Ting He, "Energy-efficient decentralized learning via graph sparsification," 2024, https://arxiv.org/abs/2401.03083.

[15] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *ICML*, 2020.

[16] Béla Bollobás, *Modern Graph Theory*, Graduate texts in mathematics. Springer, 2013.

[17] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao 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.

[18] Daniel A. Spielman and Nikhil Srivastava, "Graph sparsification by effective resistances," in *ACM STOC*, 2008.

[19] Shlomo Hoory, Nathan Linial, and Avi Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, no. 04, pp. 439–562, Aug. 2006.

[20] Joel Friedman, "Relative expanders or weakly relatively Ramanujan graphs," *Duke Mathematical Journal*, vol. 118, no. 1, pp. 19 – 35, 2003.

[21] Jeong Han Kim and Van H. Vu, "Generating random regular graphs," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 2003, STOC '03, p. 213–222, Association for Computing Machinery.

[22] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris, "Link-level measurements from an 802.11b mesh network," in *SIGCOMM*, 2004.

[23] Xinchi Qiu, Titouan Parcollet, Javier Fernandez-Marques, Pedro P. B. Gusmao, Daniel J. Beutel, Taner Topal, Akhil Mathur, and Nicholas D. Lane, "A first look into the carbon footprint of federated learning," 2021.