MATCHA: A Matching-Based Link Scheduling Strategy to Speed up Distributed Optimization

Jianyu Wang, Anit Kumar Sahu, Gauri Joshi, *Member, IEEE*, and Soummya Kar, *Fellow, IEEE*

***Abstract*—In this paper, we study the problem of distributed optimization using an arbitrary network of lightweight computing nodes, where each node can only send/receive information to/from its direct neighbors. Decentralized stochastic gradient descent (SGD) has been shown to be an effective method to train machine learning models in this setting. Although decentralized SGD has been extensively studied, most prior works focus on the error-versus-iterations convergence, without taking into account how the topology affects the communication delay per iteration. For example, a denser (sparser) network topology results in faster (slower) error convergence in terms of iterations, but it incurs more (less) communication time per iteration. We propose MATCHA, an algorithm that can achieve a win-win in this error-runtime trade-off for any arbitrary network topology. The main idea of MATCHA is to communicate more frequently over connectivity-critical links in order to ensure fast convergence, and at the same time minimize the communication delay per iteration by using other links less frequently. It strikes this balance by decomposing the topology into matchings and then optimizing the set of matchings that are activated in each iteration. Experiments on a suite of datasets and deep neural networks validate the theoretical analyses and demonstrate that MATCHA takes up to 5x less time than vanilla decentralized SGD to reach the same training loss. The idea of MATCHA can be applied to any decentralized algorithm that involves a communication step with neighbors in a graph.**

***Index Terms*—Decentralized SGD, Distributed Training, Communication-efficient Methods.**

I. INTRODUCTION

A MAJORITY of supervised machine learning problems are solved using the empirical risk minimization framework [1], [2], where the goal is to minimize the empirical risk objective function $F(\mathbf{x}) = \sum_{s\in\mathcal{D}} \ell(\mathbf{x}, \boldsymbol{s})/|\mathcal{D}|$, where $\mathcal{D}$ is the training dataset, and $\ell(\mathbf{x}, \boldsymbol{s})$ is the composite loss function for a sample $\boldsymbol{s}$. The most common algorithm to optimize $F(\mathbf{x})$ is stochastic gradient descent (SGD), where we compute the gradient of $\ell(\mathbf{x}, s)$ over small, randomly chosen subsets (called mini-batches) of b samples each [3], [4] and update $\mathbf{x}$ according to: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \eta \sum_{\boldsymbol{s}\in\mathcal{B}} \nabla \ell(\mathbf{x}^{(k)}; \boldsymbol{s})/b$, where η is referred to as the learning rate or step size. Although designed for convex objectives, mini-batch SGD has been shown to perform well even on non-convex loss functions due to its ability to escape saddle points and converge to minima which generalize well. Therefore, it is the dominant training algorithm for state-of-the-art machine learning models.

Distributed and Local-update SGD: Classical SGD was designed to run on a single computing node, and its error-convergence with respect to the number of iterations has been extensively analyzed and further improved via accelerated SGD methods. Due to the massive training datasets and neural network architectures used today, it has became imperative to design distributed SGD implementations where gradient computation and aggregation are split across multiple worker nodes. A standard way to parallelize gradient computation is the parameter server framework [5], consisting of a central server and m worker nodes which store partitions $\mathcal{D}_1, \ldots, \mathcal{D}_m$ of the training dataset $\mathcal{D}$. In each iteration, the parameter waits for the m workers to return one mini-batch gradient each and aggregates them to update the model $\mathbf{x}$. However, in bandwidth-limited computing environments the need for constant communication between the parameter server and worker nodes can be prohibitively expensive and slow [6]. A simple way to improve communication-efficiency is to use local-update or periodic averaging SGD [7], [8], which is the core of the emerging field of federated learning [9], [10]. Local-update SGD divides the training into communication rounds. In the t-th communication round, the m nodes read the current global model $\mathbf{x}$ and make τ local SGD updates to optimize their local objective $F_i(\mathbf{x}) = \sum_{\boldsymbol{s}\in\mathcal{D}_i} \ell(\mathbf{x}; \boldsymbol{s})/|\mathcal{D}_i|$. The resulting models are then sent to the central server, which averages them and updates the global model.

Decentralized SGD Training: Besides the temporal communication reduction achieved by local-update SGD, one can achieve spatial communication reduction by eliminating the central aggregating server, and instead perform training in a decentralized sparse node topology where nodes can only communicate with their neighbors [11]. Decentralized SGD, which is the focus of this paper, is especially suitable for training outside a controlled data-center setting, for example, multi-agent systems or

Manuscript received 24 February 2022; revised 22 July 2022 and 13 September 2022; accepted 21 September 2022. Date of current version 9 November 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ketan Rajawat. This work was supported in part by NSF under Grants CCF-1850029, CCF-2045694, and CCF-2112471, in part by 2018 IBM Faculty Research Award, and in part by the Qualcomm Innovation Fellowship (Jianyu Wang). *(Corresponding authors: Jianyu Wang; Gauri Joshi.)*

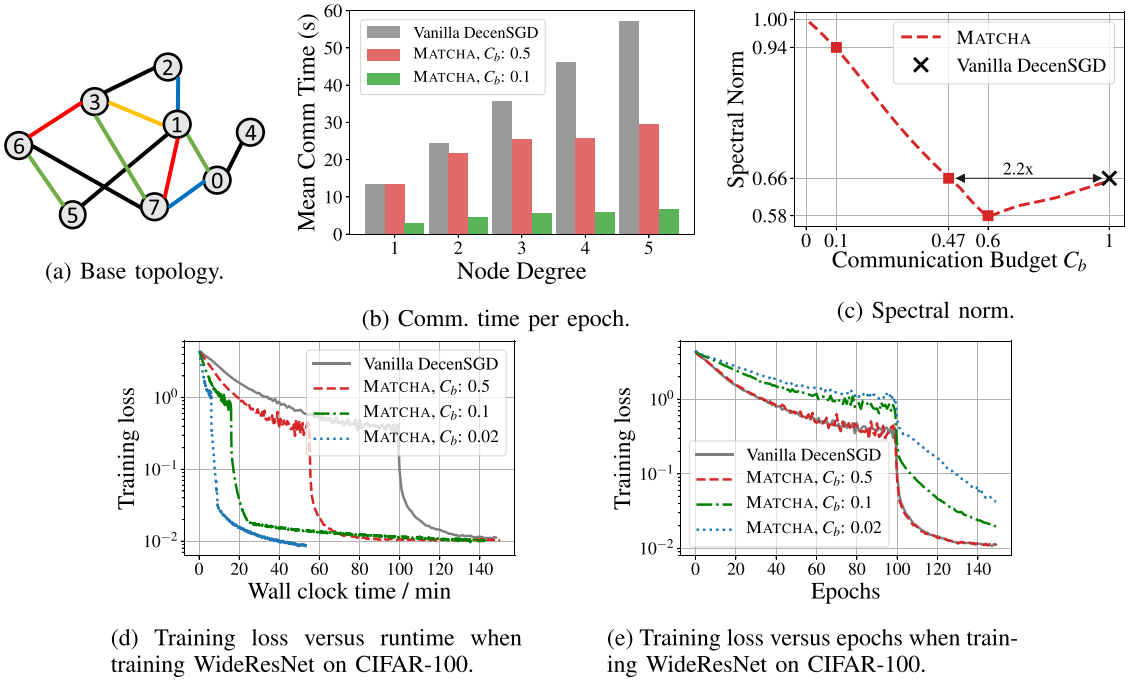
Jianyu Wang was with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA. He is now with Meta Platforms, Menlo Park, CA 94025 USA (e-mail: jianyuw1@andrew.cmu.edu).

Gauri Joshi and Soummya Kar are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: gaurij@andrew.cmu.edu; soummyak@andrew.cmu.edu).

Anit Kumar Sahu is with the Amazon Alexa AI, Pittsburgh, PA 15232 USA (e-mail: anit.sahu@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2022.3212536, provided by the authors.

Digital Object Identifier 10.1109/TSP.2022.3212536



(a) Base topology.

(b) Comm. time per epoch.

(c) Spectral norm.

(d) Training loss versus runtime when training WideResNet on CIFAR-100.

(e) Training loss versus epochs when training WideResNet on CIFAR-100.

Fig. 1. (a) Base network topology used to compare MATCHA with vanilla decentralized SGD (DecenSGD). (b) MATCHA reduces the communication time more sharply for higher degree nodes (for instance node 1), which tend to have fewer connectivity-critical links. (c) Lower spectral norm yields better convergence rate. Communication budget C_b represents the average frequency of communication over the links in the network. (d,e) Loss-versus-time and loss-versus-epoch curves when training WideResNet on CIFAR-100.

federated learning [9], [10], [12] on edge devices. Not having a central parameter server [5], [13], [14], [15] avoids bandwidth bottleneck and makes the system scalable to a large number of nodes. In vanilla decentralized SGD [11], [16], each node makes only one local SGD update before exchanging model updates with its neighbors. The case of $\tau > 1$ local updates is considered in only a few works such as [8], [17], which propose and analyze the periodic decentralized SGD algorithm which combines temporal and spatial communication reduction.

Need to Consider Error-versus-runtime Convergence: Although decentralized SGD has been extensively studied in distributed optimization community, most works analyzing its convergence [8], [11], [16], [18], [19], [20], [21], [22], [23], [24] only focus on the number of iterations or communication rounds required to achieve a target error. They do not explicitly consider or demonstrate how the topology affects the training runtime, that is, wall-clock time required to complete each iteration. However, there is a fundamental trade-off between the runtime per iteration and the error-versus-iterations convergence depending on the communication and aggregation mechanism used by the algorithm. For instance, denser networks typically give faster error convergence but they incur a higher communication delay per iteration. Thus, in order to achieve the fastest error-versus-wallclock time convergence, it is critical to jointly optimize the iteration complexity as well as the runtime per iteration by juxtaposing optimization and scheduling techniques. Only a few previous works such [15], [25], [26] study the error-versus-wallock time convergence from a theoretical perspective, but only for the parameter server model. We adopt this system-aware approach in the decentralized SGD setting and seek to design inter-node communication strategies that achieve the fastest error-versus-runtime convergence.

All Links should Not be Treated Equally: Recent works have proposed various methods to reduce the communication delay when performing decentralized SGD. One such simple way is to perform periodic decentralized SGD, also called *Periodic DecenSGD (P-DecenSGD)* [8], [17]. In P-DecenSGD, each node makes $\tau > 1$ local model updates before synchronizing with its neighbors. Thus, all links in the base topology are activated simultaneously after every τ iterations. Other methods include gradient quantization [27], asynchronous model aggregation [28], stochastic gradient push [29] and pairwise gossip [18], [30], [31] (see Section VII for a detailed comparison). *All these methods to improve the communication-efficiency of decentralized SGD method have one common drawback – they are agnostic to the contribution of each inter-node link towards the overall connectivity of the graph.* However, each link affects the error convergence and the wall-clock time per iteration differently. For example, in the topology shown in Fig. 1(a), the $0-4$ link is critical to maintaining the graph's connectivity and in turn ensuring fast error convergence, whereas removing or infrequently using the $1-3$ link has little impact on the average graph connectivity but can help reduce the communication delay. This gives us the novel insight that connectivity-critical links such as $0-4$ should be activated more frequently.

Main Contributions: MATCHA uses this novel insight to develop a principled approach to optimize the communication frequency of each link depending on how it affects the expected connectivity of the topology and the delay per iteration. In contrast to existing communication-efficient SGD methods which

have to compromise on error-versus-iterations convergence in order to reduce the communication delay per iteration, MATCHA can flexibly reduce the communication delay per iteration while preserving (or even improving) the speed of error-versus-iterations convergence. To the best of our knowledge, this is the first work that attempts to strike the best error-runtime trade-off in decentralized SGD for an arbitrary network topology. The main contributions of this paper are summarized below.

1) *Saving Communication Time by Using Bipartite Matchings:* The communication delay of the model synchronization step in decentralized SGD is an increasing function of the maximal node degree, as we demonstrate in Section II below. To reduce this delay without hurting convergence, we decompose the graph into *matchings*. Each matching has degree one and is a set of disjoint links that communicate in parallel, as illustrated by the colored links in Fig. 1(a). The probability of activating each matching is optimized so as to maximize the algebraic connectivity of the expected topology (captured by the second smallest eigenvalue λ_2 of the graph Laplacian). This results in more frequent communication over matchings that contain connectivity-critical links (ensuring fast error-versus-iterations convergence) and less frequent over others (saving the communication time per iteration).
2) *Flexible Communication Budget:* MATCHA allows the system designer to set a flexible communication budget C_b, which represents the relative communication time of MATCHA compared to vanilla DecenSGD. When $C_b = 1$, MATCHA reduces to vanilla decentralized SGD studied in [16]. When we set $C_b < 1$, MATCHA carefully reduces the communication frequency of each link, depending upon its importance in maintaining the overall connectivity of the graph. For example, observe in Fig. 1(b) that by setting $C_b = 0.1$, MATCHA achieves a $1/0.1 = 10\times$ reduction in expected communication time per iteration. The communication reduction is much larger for higher-degree nodes, as shown in Fig. 1(b). This judicious asymmetry in communication reduction is the key ingredient that enables MATCHA to preserve fast error-versus-iterations convergence.
3) *Same or Faster Error Convergence than Vanilla Decentralized SGD:* In Section V we present a convergence analysis of MATCHA for non-convex objectives and illustrate the dependence of the error on ρ, the spectral norm of the mixing matrix (defined formally later). This analysis shows that for a suitable communication budget, MATCHA achieves the same or smaller ρ as vanilla decentralized SGD — a smaller ρ implies faster error convergence. For example, observe in Fig. 1(c) that MATCHA has the same spectral norm as vanilla decentralized SGD (DecenSGD) with a $2.2\times$ less communication budget per iteration, and if we set $C_b = 0.6$ then the spectral norm is even lower. *In this case, contrary to intuition,* MATCHA*not only reduces the per-iteration communication delay but also gives faster error-versus-iterations convergence.*
4) *Experimental Results on Error-versus-wallclock Time Convergence:* In Section VI we evaluate the performance of MATCHA on various deep learning tasks including image classification and language modeling, and for several base topologies including Erdős-Rényi and geometric graphs. The empirical results consistently corroborate theoretical analyses and show that MATCHA can get up to $5.2\times$ reduction in wall-clock time (computation plus communication time) to achieve the same training accuracy as vanilla decentralized SGD, as illustrated in Fig. 1(d). Moreover, MATCHA achieves test accuracy that is comparable or better than vanilla decentralized SGD.
5) *Extendable to Other Subgraphs and Computations:* While we currently decompose the topology into matchings, our approach can be extended to other sub-graphs such as edges or cliques. It is also complementary to and can be combined with other methods such as gradient compression or quantization [27], [32], asynchronous model aggregation [28], and gradient tracking and variance reduction techniques [33], [34], [35], as we review in Section VII. Furthermore, going beyond decentralized SGD, the matching-based link scheduling strategy, which is the core idea of MATCHA, is extendable to any distributed computation or consensus algorithm that requires frequent synchronization between neighboring nodes.

II. PRELIMINARIES AND PROBLEM FORMULATION

A. Node Topology and Graph Theory Preliminaries

Consider a network of m worker nodes, which correspond to the vertex set $\mathcal{V} = \{1, 2, \ldots, m\}$. The communication links connecting the nodes (vertices) are represented by an arbitrary undirected graph $\mathcal{G}$ with the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each node i can only communicate with its neighbors, that is, it can communicate with node j if and only if $(i, j) \in \mathcal{E}$. We use $\mathcal{N}_i = \{j | (j, i) \in \mathcal{E}, j \in \mathcal{V}, j \neq i\}$ to denote the neighbor index set of node i. The degree of node i is defined as $d_i = |\mathcal{N}_i|$, and the maximal node degree is denoted by $\Delta_{\mathcal{G}} = \max_{i \in \mathcal{V}} d_i$. We assume that the graph $\mathcal{G}$ is connected, that is, for any pair of nodes, there exists a path linking them.

The communication graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ (*i.e.*, network topology connecting nodes) can also be represented using the adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$. In particular, $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$; $A_{ij} = 0$ otherwise. The *graph Laplacian* matrix $\mathbf{L}$ is defined as $\mathbf{L} = \text{diag}(d_1, \ldots, d_m) - \mathbf{A}$. The graph Laplacian is a positive semi-definite matrix, hence its eigenvalues can be ordered and represented as $0 = \lambda_1(\mathbf{L}) \leq \lambda_2(\mathbf{L}) \leq \cdots \leq \lambda_m(\mathbf{L})$. The graph $\mathcal{G}$ is a connected graph if and only if the second smallest eigenvalue $\lambda_2(\mathbf{L})$ is strictly greater than 0. In spectral graph theory [36], [37], the algebraic connectivity of a graph is defined as follows.

Definition 1 (Algebraic Connectivity $\lambda_2(\mathbf{L})$): Given a connected graph $\mathcal{G}$ and its corresponding Laplacian matrix $\mathbf{L}$, the algebraic connectivity of $\mathcal{G}$ is defined as the second smallest eigenvalue of $\mathbf{L}$. A larger value of $\lambda_2(\mathbf{L})$ implies a denser graph.

Next, we define the notions of a matching subgraph and matching decomposition, which are central to this paper.

Definition 2 (Matching): A matching is a subgraph of $\mathcal{G}$, in which each vertex is incident with at most one edge.

Definition 3 (Matching Decomposition): A matching decomposition is a set M disjoint matching sub-graphs $\{\mathcal{G}_j(\mathcal{V},\mathcal{E}_j)\}_{j=1}^{M}$ of $\mathcal{G}$ such that $\mathcal{E}=\bigcup_{j=1}^{M}\mathcal{E}_j$ and $\mathcal{E}_i\bigcap\mathcal{E}_j=\varnothing,\forall i\neq j$.

The matching decomposition is not necessarily unique, and there are polynomial-time edge-coloring algorithms [38] to decompose a given graph into matchings. A nice property of matching decomposition is that the number of matchings M is closely related to the maximal degree $\Delta_{\mathcal{G}}$ of the graph. In particular, the edge coloring algorithm in [38] provably guarantees that the number of matchings M equals to either $\Delta_{\mathcal{G}}$ or $\Delta_{\mathcal{G}}+1$.

B. Decentralized Stochastic Optimization Preliminaries

We consider that each worker node only has access to its own local dataset $\mathcal{D}_i$ where $i\in\{1,2,\ldots,m\}$. Our objective is to use this network of m nodes to train one common model $\mathbf{x}$) using the joint dataset and the given network topology. In particular, we seek to minimize the objective function $F(\mathbf{x})$, which is defined as follows:

$$F(\mathbf{x})\triangleq\frac{1}{m}\sum_{i=1}^{m}F_i(\mathbf{x})=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{D}_i|}\sum_{\boldsymbol{s}\in\mathcal{D}_i}\ell(\mathbf{x};\boldsymbol{s}) \tag{1}$$

where $\mathbf{x}$ denotes the model parameters (for instance, the weights and biases of a neural network), $F_i(\mathbf{x})$ is the local objective function, $\boldsymbol{s}$ denotes a single data sample, and $\ell(\mathbf{x};\boldsymbol{s})$ is the loss function for sample $\boldsymbol{s}$, defined by the learning model.

Vanilla Decentralized SGD Update Rule: In order to minimize the objective function (1), many decentralized optimization algorithms have been developed starting from the seminal work of [11], [39]. In a typical decentralized optimization algorithm, each worker node alternates between local computation and communication with its neighboring nodes. For example, in vanilla decentralized SGD (DecenSGD) [16], [19], [24], all nodes iteratively run the following:

1) *Parallel Local Computation:* Node i computes the stochastic gradient with respect to its local version of model parameters: $g_i(\mathbf{x}_i^{(k)})=\frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{s}\in\mathcal{B}}\nabla\ell(\mathbf{x}_i^{(k)};\boldsymbol{s})$ where $\mathcal{B}\subseteq\mathcal{D}_i$ is a randomly sampled mini-batch $\mathcal{B}$ from the local dataset. Then, node i updates its local parameters using the stochastic gradient: $\mathbf{x}_i^{(k+\frac{1}{2})}=\mathbf{x}_i^{(k)}-\eta g_i(\mathbf{x}_i^{(k)})$ where η denotes the learning rate.
2) *Communication with Neighbors:* In order to achieve consensus, worker nodes need to exchange model parameters with neighbors. This procedure involves a two-way communication.
   - Node i sends its local model parameters $\mathbf{x}_i^{(k+\frac{1}{2})}$ to its neighboring nodes $\mathcal{N}_i$;
   - Node i receives model parameters from each of its neighbors. After this step, node i should have all neighbors' model parameters $\left\{\mathbf{x}_j^{(k+\frac{1}{2})}\right\}_{j\in\mathcal{N}_i}$.
3) *Mixing Local Model with Neighbors:* Since the nodes seek to jointly train a common model $\mathbf{x}$, they perform a consensus step by taking a weighted average of their model $\mathbf{x}_i^{(k+\frac{1}{2})}$ with neighboring models. Thus, node i updates its model as: $\mathbf{x}_i^{(k+1)}=\sum_{j\in\mathcal{N}_i\cup\{i\}}W_{ji}\mathbf{x}_j^{(k+\frac{1}{2})}$, where W_{ji} is the (j,i)-th element of the mixing matrix $\mathbf{W}\in\mathbb{R}^{m\times m}$. It represents the weight assigned to neighbor j when updating node i's model. Observe that $W_{ij}\neq 0$ if and only if node i and node j are connected, i.e., $(i,j)\in\mathcal{E}$.

Putting the above steps together, we can write the update rule of vanilla DecenSGD as follows:

$$\mathbf{x}_i^{(k+1)}=\underbrace{\sum_{j=1}^{m}W_{ij}}_{\text{consensus step}}\underbrace{\left[\mathbf{x}_j^{(k)}-\eta g_j\left(\mathbf{x}_j^{(k)}\right)\right]}_{\text{parallel local computation}}. \tag{2}$$

In order to enforce consensus among the nodes in (2), it is common practice to use a symmetric and doubly stochastic (*i.e.*, the sum of each column/row is 1) mixing matrix $\mathbf{W}$. A common choice is equal weight matrix, defined as follows:

$$\mathbf{W}=\mathbf{I}-\alpha\mathbf{L} \tag{3}$$

where α is a tunable parameter and $\mathbf{L}$ denotes the Laplacian matrix of graph $\mathcal{G}$. The definition (3) makes $\mathbf{W}$ symmetric and doubly-stochastic by construction and is widely used in literature [18], [40]. A larger α assigns a higher weight to neighboring models. For example, if node 1 is only connected to nodes 2 and 3, then the first row of graph Laplacian $\mathbf{L}$ is $[2,-1,-1,0,\ldots,0]$ and the corresponding first row of $\mathbf{W}$ is $[1-2\alpha,\alpha,\alpha,0,\ldots,0]$. Although we use a fixed $\mathbf{W}$ to describe decentralized SGD in this section, it can also vary across iterations. For example, in periodic decentralized SGD [8], [17] where each node performs τ local SGD updates before communicating with neighbors, $\mathbf{W}^{(k)}=\mathbf{W}$ when $k \mod \tau=0$ and $\mathbf{W}^{(k)}=\mathbf{I}$ otherwise. Our proposed MATCHA algorithm described in Section III also constructs a time-varying mixing matrix.

Convergence Analysis of Decentralized SGD: When the objective function (1) is a non-convex function, then the typical metric for convergence is the gradient norm $\|\nabla F(\mathbf{x})\|$. In particular, we have the following informal lemma.

Lemma 1 (Convergence guarantee of DecenSGD, [8], [16]): Suppose $\overline{\mathbf{x}}$ denote the averaged model across all worker nodes and all local models at nodes are initialized from the same point $\overline{\mathbf{x}}^{(1)}$. Then under standard assumptions on the stochastic gradients (formally stated in Section V), we have after K iterations:

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla F\big(\overline{\mathbf{x}}^{(k)}\big)\|^2=\mathcal{O}\left(\frac{1}{\sqrt{mK}}\right)+\mathcal{O}\left(\frac{m}{K}\frac{\rho}{(1-\sqrt{\rho})^2}\right) \tag{4}$$

where K is the total iterations and ρ denotes the spectral norm (i.e., largest singular value) of matrix $\mathbf{W}^2-\mathbf{1}\mathbf{1}^\top/m$.

Lemma 1 shows that the convergence rate of DecenSGD is controlled by the spectral norm $\rho=\|\mathbf{W}^2-\mathbf{1}\mathbf{1}^\top/m\|_2<1$. A smaller ρ yields a better convergence error bound. In general, the value of ρ relates to the connectivity of the network topology. When worker nodes can communicate with any other nodes (*i.e.*, the network topology is fully connected), then $\rho=0$ and the second term in (4) achieves its minimum. And when the network topology is extremely sparse, then ρ will be close to 1, making the second term in (4) approach to infinity.

Remark 1 (Larger $\lambda_2(\mathbf{L})$ results in faster error convergence): When the mixing matrix is of the form $\mathbf{W} = \mathbf{I} - \alpha\mathbf{L}$, then (see [41] for the detailed derivation)

$$\rho = \max\{(1-\alpha\lambda_m(\mathbf{L}))^2, (1-\alpha\lambda_2(\mathbf{L}))^2\}. \tag{5}$$

$$\geq \frac{1-\frac{\lambda_2(\mathbf{L})}{\lambda_m(\mathbf{L})}}{1+\frac{\lambda_2(\mathbf{L})}{\lambda_m(\mathbf{L})}} \tag{6}$$

where the equality is achieved by setting $\alpha = 2/(\lambda_2(\mathbf{L}) + \lambda_m(\mathbf{L}))$. Thus, a graph with larger $\lambda_2(\mathbf{L})/\lambda_m(\mathbf{L})$[1] will have smaller value of ρ and hence, a better convergence error bound. The value of $\lambda_2(\mathbf{L})/\lambda_m(\mathbf{L})$ is more sensitive to $\lambda_2(\mathbf{L})$ because $\lambda_m(\mathbf{L})$ is always lower bounded by the average node degree, which is away from zero [41]. Therefore, a larger $\lambda_2(\mathbf{L})$ will result in a better error convergence bound.

We use the above observation in MATCHA, and design a topology sequence that maximizes $\lambda_2(\mathbf{L})$ of the expected graph (see Section III) and then optimize the value of α in order to minimize the corresponding spectral norm ρ (see Section IV). The resulting mixing matrix sequence reduces the communication delay per iteration while maintaining or improving the error convergence bound.

C. Understanding How Network Topology Affects the Runtime Per Iteration

Observe that Lemma 1 gives a convergence bound for decentralized SGD with respect to the number of iterations K and the spectral norm ρ, without accounting for the time spent per iteration. In this section, our goal is to understand how the network topology affects the runtime per iteration. The runtime includes the local computation time and the communication time $T_{\mathcal{G}}^{\text{comm}}$ required to perform the two-way exchange of models with neighboring nodes (Step 2 of the vanilla DecenSGD update rule described above). Since the communication time $T_{\mathcal{G}}^{\text{comm}}$ generally dominates over local computation time in bandwidth-limited environments, we focus on understanding how $T_{\mathcal{G}}^{\text{comm}}$ is affected by the network topology. In particular, we have the following proposition.

Proposition 1: The expected communication time per iteration $\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}]$ of decentralized optimization algorithms monotonically increases with the maximal degree in the graph, that is,

$$\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}] \geq t(\Delta_{\mathcal{G}}) \tag{7}$$

where $t(\cdot)$ is a monotonically increasing function and $\Delta_{\mathcal{G}} = \max_{i\in\mathcal{V}} d_i$, the maximal node degree in graph $\mathcal{G}$.

In the rest of this section, we justify why Proposition 1 is true irrespective of the link delay distribution and the inter-node communication protocol. If the reader believes Proposition 1, they may skip ahead to Section III, which describes the proposed MATCHA algorithm.

[1]The quantity $\lambda_2(\mathbf{L})/\lambda_m(\mathbf{L})$ is typically maximized by spectral expanders such as Ramanujan graphs.

Node i's per-iteration communication time T_i increases with its degree d_i: Recall that the communication step of decentralized SGD (see step 2 in Section 2.1) requires a two-way communication of models – each node needs to send its model parameters to its neighbors and then receive one model back from each of the neighbors. Without loss of generality assume that the fastest node finishes its local computation (step 1 in Section 2.1) at time 0. Starting from this time, let random variable $T_{i\to j}^{\text{send}}$ denote the time for node i to complete sending its model to node j respectively (including any idle time spent waiting for other concurrent inter-node communication). As a result, node i takes time

$$T_i = \max\left(\max_{j\in\mathcal{N}_i} T_{i\to j}^{\text{send}}, \max_{j\in\mathcal{N}_i} T_{j\to i}^{\text{send}}\right) \tag{8}$$

to communicate with its neighbors in each iteration. Since the slowest neighboring node is always the bottleneck, $\mathbb{E}[T_i] = t(d_i)$, where $t(\cdot)$ is a monotonically increasing function.

The exact form of the function t depends on the distributions of the random variables $T_{i\to j}^{\text{send}}$, which in turn depend on many factors such as the inter-node communication and collision avoidance protocols, link delays, background processes etc. For example, if node i sequentially sends its models to its d_i neighbors where each transfer takes time S_i then $\max_{j\in\mathcal{N}_i} T_{i\to j}^{\text{send}} = d_i S_i$ and $\mathbb{E}[T_i]$ increases linearly with d_i. Instead, if node i broadcasts its model to its neighbors in time $S_i \sim \text{Exp}(\mu)$ that is i.i.d. across nodes, then $\max_{j\in\mathcal{N}_i} T_{i\to j}^{\text{send}} = S_i$ and $\mathbb{E}[T_i] = \max_{j\in N_i} S_j \approx \frac{\log(1+d_i)}{\mu}$, which is again monotonically increasing in d_i.

The communication time $T_{\mathcal{G}}^{comm}$ increases with the maximal node degree: Now let us express the communication time per iteration $T_{\mathcal{G}}^{\text{comm}}$ in terms of T_i. In an ideal network where all nodes can simultaneously communicate with their respective neighbors, $T_{\mathcal{G}}^{\text{comm}} = \max_{i\in\mathcal{V}} T_i$. However, in practice,

$$\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}] \geq \mathbb{E}[\max_{i\in\mathcal{V}} T_i] \tag{9}$$

$$\geq \max_{i\in\mathcal{V}} \mathbb{E}[T_i] \tag{10}$$

$$= t(\Delta_{\mathcal{G}}). \tag{11}$$

The last equality (11) comes from the fact that $t(\cdot)$ is an increasing function. Thus, we have justified that Proposition 1 is true. Furthermore, the equality holds in (10) only when all nodes can perform their send and receive communications in parallel. However, achieving this in practice requires careful allocation of time/frequency resources to the send/receive communications on each link in order to avoid conflicts. Below, we show examples of how the inter-node communications can be orchestrated.

- *Frequency Division Multiplexing model (FDM):* To avoid conflicting inter-node communication, each node can have a dedicated frequency channel to send its model to neighbors and d_i receive channels corresponding to its neighbors' send frequencies. In this case with a total of m frequencies, all nodes can indeed communicate in parallel and thus $\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}]$ will achieve the lower bound in (10). Alternately, we can decompose the topology into M matchings and assign one send and one receive frequency to each matching. This strategy again achieves the lower bound in

(10) but $2\Delta_{\mathcal{G}}$ (which is typically smaller than m) frequency channels.

- *Time Division Multiplexing model (TDM):* In time division multiplexing, all nodes in the system use the same frequency to send and receive messages. To avoid conflicts, each inter-node transmission needs to be assigned a dedicated time slot. A naive approach is to assign one time slot each node to broadcast its model to its neighbors. This would result in $\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}]$ to scale linearly with the number of nodes m. Instead, our link-scheduling algorithm MATCHA proposes a more efficient TDM implementation. We decompose the graph into $M = \Delta_{\mathcal{G}}$ or $M = \Delta_{\mathcal{G}} + 1$ matchings and then assign one time slot to each matching. Since a matching contains non-conflicting links, all links in a matching can be activated at the same timeslot. As a result, $\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}] \propto \Delta_{\mathcal{G}}$, which also achieves the lower bound in (10) with equality.

III. MATCHA: LINK SCHEDULING VIA MATCHING DECOMPOSITION SAMPLING

In Section II, we showed that the communication delay per iteration increases with the maximal node degree $\Delta_{\mathcal{G}}$. On the other hand, we also showed that a better-connected graph (which often has a higher $\Delta_{\mathcal{G}}$) gives faster error-versus-iterations convergence. Our proposed link-scheduling strategy MATCHA seeks the best trade-off between these opposing forces. MATCHA generates a time-varying communication topology that maximizes the algebraic connectivity of the expected graph, while keeping the expected maximal degree small. By doing so, we automatically communicate more frequently over connectivity-critical links (preserving fast error convergence) and less frequently other links (reducing the communication delay per iteration).

In this section we describe a *matching decomposition sampling* (illustrated in Fig. 2) procedure to generate a time-varying topology sequence. Later, in section 4, we will discuss how to optimize the mixing matrix weights W_{ji} for this topology sequence. All the steps described in Section III and Section IV are pre-processing steps that can be performed before training starts. Thus MATCHA does not add any per-iteration overhead to the total training time.

A. Step 1: Matching Decomposition.

First, we decompose the base node topology into total M disjoint sub-graphs i.e., $\{\mathcal{G}_j(\mathcal{V}, \mathcal{E}_j)\}_{j=1}^{M}$, $\mathcal{E} = \bigcup_{j=1}^{M} \mathcal{E}_j$ and $\mathcal{E}_i \bigcap \mathcal{E}_j = \varnothing, \forall i \neq j$, as defined in Section II. Only a subset of these sub-graphs is activated in each training iteration. It has been justified in Proposition 1 that the communication delay is a monotonically increasing function of the maximal node degree. Thus, in order to save the communication delay, it is of particular interest for the activated topology to have smaller maximal degree than the base node topology. As a consequence, a key design question here is: which kind of decomposition $\{\mathcal{G}_j\}$ of the base topology to use so that we can easily and flexibly control the maximal degree of the activated topology?

In this paper, we choose to use *matchings* as the decomposition basis. All nodes in a matching have at most degree one.

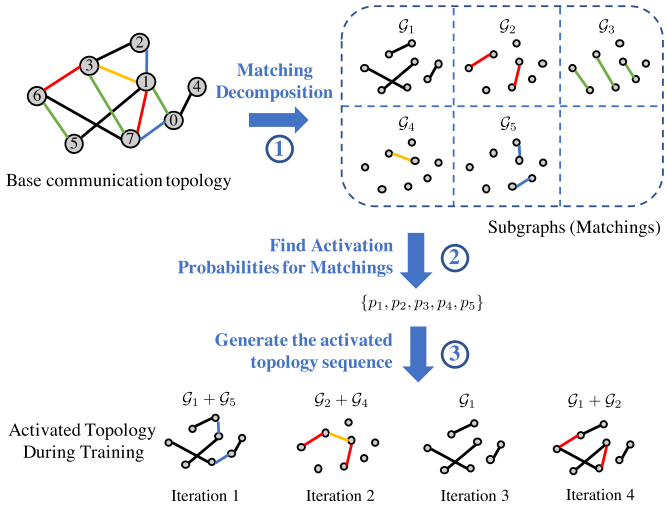


Fig. 2. Illustration of MATCHA. We decompose the base communication graph into disjoint subgraphs (in particular, matchings, in order to parallelize communication). At each communication round, we activate a subset of these matchings to construct a sparse subgraph of the base topology. Worker nodes are synchronized only through the activated topology. If one node is not involved in the activated subgraphs, it just performs one local update.

The inter-node links are disjoint and can operate in parallel. As shown in Section II, a nice property of matching decomposition is that, using the edge-coloring algorithm [38], one can provably guarantees that the number of matchings equals to either $\Delta_{\mathcal{G}}$ or $\Delta_{\mathcal{G}} + 1$, where $\Delta_{\mathcal{G}}$ is the maximal node degree. For almost all graphs, it has been proved in [42] that there exists a decomposition scheme such that $M = \Delta_{\mathcal{G}}$. That is, the node(s) with highest degree appears in all M matchings. Therefore, if there are N matchings are activated for communication at certain iteration, then the maximal degree of the activated topology is exactly N.

Another critical benefit of using matching decomposition is that it automatically construct a link scheduling scheme that can achieve the lower bound (7) of communication time per iteration. As discussed in Section II-C, in TDM communication model, one can assign one time slot for each matching and communicate them sequentially. In FDM communication model, one can assign one frequency to each matching and communicate them in parallel. As a consequence, by using matching, one can easily control the communication delay $\mathbb{E}[T_{\mathcal{G}}^{\text{comm}}] = t(\Delta_{\mathcal{G}}) = t(N)$ by adjusting N, the number of matchings to be activated at each iteration.

B. Step 2: Computing Matching Activation Probabilities.

In order to activate a subset of the M matchings, in each training iteration, we assign an independent Bernoulli random variable B_j, which is 1 with probability p_j and 0 otherwise, to each matching $\mathcal{G}_j, \forall j \in \{1, \ldots, M\}$. The links in matching j will be used for information exchange in the consensus step only when the realization of B_j is 1. Using this random sampling scheme, one can write the expected communication time per

iteration as

$$\text{Expected Comm. Time} = \mathbb{E}\left[t(\Delta_{\mathcal{G}^{(k)}})\right] = \mathbb{E}\left[t\left(\sum_{j=1}^{M} \mathsf{B}_j\right)\right] \tag{12}$$

where $\mathcal{G}^{(k)}$ denotes the activated topology at the k-th iteration. We further define $\mathbb{E}[\mathsf{B}_j] = p_j$ as the *activation probability* of the j-th matching. When all p_j's equal to 1, the algorithm reduces to vanilla DecenSGD and takes $t(M)$ time to finish one consensus step. To reduce the communication delay, we define *communication budget* $C_b > 0$, and impose the constraint $\mathbb{E}\left[t\left(\sum_{j=1}^{M} \mathsf{B}_j\right)\right] \leq C_b \cdot t(M)$. For example, $C_b = 0.1$ means that the expected communication time per iteration of MATCHA be 10% of the time per iteration of vanilla DecenSGD. Eqn. (12) can be further simplified in the TDM model. When we ignore all constants, it follows that $\mathbb{E}\left[t\left(\sum_{j=1}^{M} \mathsf{B}_j\right)\right] = \mathbb{E}\left[\sum_{j=1}^{M} \mathsf{B}_j\right] = \sum_{j=1}^{M} p_j$.

As mentioned before, the key idea of MATCHA is to give more importance to critical links. This is achieved by choosing a set of activation probabilities that maximize the connectivity of the expected graph given a communication time constraint. That is, we solve the optimization problem:

$$\begin{aligned} \max_{p_1,\ldots,p_M} \quad & \lambda_2\left(\sum_{j=1}^{M} p_j \mathbf{L}_j\right) \\ \text{subject to} \quad & \mathbb{E}\left[t\left(\sum_{j=1}^{M} \mathsf{B}_j\right)\right] \leq C_b \cdot t(M), \\ & 0 \leq p_j \leq 1,\ \forall j \in \{1, 2, \ldots, M\}, \end{aligned} \tag{13}$$

where $\mathbf{L}_j$ denotes the Laplacian matrix of the j-th subgraph and $\sum_{j=1}^{M} p_j \mathbf{L}_j$ can be considered as the Laplacian of the expected graph. Moreover, recall that λ_2 represents the algebraic connectivity and is a concave function [36], [40]. The optimization problem (13) can be solved via numerical methods. In the TDM communication model, the constraint in (13) reduces to $\sum_{j=1}^{M} p_j \leq C_b M$. Thus, it follows that (13) is a convex problem and can be solved efficiently. Typically, a larger value of λ_2 implies a better-connected graph and leads to a better rate to reach consensus in many decentralized applications [40].

C. Step 3: Generating Random Topology Sequence.

Given the activation probabilities obtained by solving (13), in each iteration k, we generate an independent Bernoulli random variable $\mathsf{B}_j^{(k)}$ for each matching $j = 1, \ldots M$. Thus, in the k-th iteration, the activated topology $\mathcal{G}^{(k)}(\mathcal{V}, \mathcal{E}^{(k)})$, in which $\mathcal{E}^{(k)} = \bigcup_{j=1}^{M} \mathsf{B}_j^{(k)} \mathcal{E}_j$, is sparse or even disconnected. Note that if a node has no activated links, then it still continues to make local updates to the current local models. Besides the topology sequence, one can also obtain the corresponding Laplacian matrix sequence: $\mathcal{L}^{(k)} = \sum_{j=1}^{M} \mathsf{B}_j^{(k)} \mathbf{L}_j, \forall k \in \{1, 2, \ldots\}$. *All of these information can be obtained and assigned apriori to worker nodes before starting the training procedure*. In practice, the server does not need to send the entire Laplacian matrix sequence $\{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \ldots \mathcal{L}^{(K)}\}$ to worker nodes, the size of which can be up to MMK. Instead, the server can first broadcast the matching decomposition $\{\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_M\}$ and then send the activated index sequence to all nodes, the size of which is only up to MK. Also, this additional communication cost before training is negligible compared to the that of transferring neural network parameters during training (size of neural network times number of iterations). Besides, there is another simple way to eliminate the communication of the network topologies – we can assign the same random seed to all clients and let them generate the sequence of activated graphs by themselves. By doing this, we can avoid sending the whole sequence of activated graphs.

D. Extension to Other Design Choices.

Combining all the previous steps together, we provide a pseudo code of MATCHA in Algorithm 1. One can observe that the proposed MATCHA framework of activating different subgraphs in each iteration is very general – it can be extended to various other delay models, graph decomposition methods and algorithms involving decentralized averaging. For example, instead of activating all matchings independently, one can choose to activate only one matching at each iteration. Also, instead of using matchings as the decomposition basis, one can also use other sub-graphs, such as edges, trees or cliques.

Finally, although this paper focuses on using mini-batch stochastic gradients, the local computation step can take many other forms, such as updating local models using compressed/quantized gradient, variance-reduced or stale asynchronous gradient. The core idea of MATCHA can still apply as long as the decentralized optimization algorithm involves a consensus step (communication then mixing) as marked in (2). Later in Section VI, we provide an example of combining MATCHA with gradient compression methods.

IV. Optimizing the Mixing Matrix Weights of the Topology Sequence

In order to make the best use of the sequence of activated subgraphs generated by MATCHA when running decentralized SGD with a time-varying topology, we need to optimize the proportions in which the local models are averaged together in the consensus step of (2). As mentioned in Section II, a common practice is to use an equal weight mixing matrix as [18], [40], [43]:

$$\mathbf{W}^{(k)} = \mathbf{I} - \alpha \mathcal{L}^{(k)} = \mathbf{I} - \alpha \sum_{j=1}^{M} \mathsf{B}_j^{(k)} \mathbf{L}_j, \tag{14}$$

where $\mathcal{L}^{(k)} = \sum_{j=1}^{M} \mathsf{B}_j^{(k)} \mathbf{L}_j$ denotes the graph Laplacian at the k-th iteration. Each matrix $\mathbf{W}^{(k)}$ is symmetric and doubly stochastic by construction, and the parameter α represents the weight of neighbor's information in the consensus step. To guarantee sufficient decrease after every iteration of decentralized SGD for smooth and non-convex losses, the *spectral norm* $\rho = \|\mathbb{E}[\mathbf{W}^{(k)\top}\mathbf{W}^{(k)}] - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\|_2$ must be less than 1. In

Algorithm 1: MATCHA: Link-Scheduling Algorithm for Decentralized SGD.

Input: Network topology $\mathcal{G}(\mathcal{V},\mathcal{E})$ and communication budget C_b.

Part 1 `Preprocessing`$(\mathcal{G}, C_b)$**:**

  Matchings $\{\mathcal{G}_j\}_{j=1}^M$, Laplacian matrices $\{\mathbf{L}_j\}_{j=1}^M$ = `Decomposition`$(\mathcal{G})$

  Obtain activation probabilities $\{p_j\}_{j=1}^M$ by solving optimization problem (13)

  Obtain the best value of α by solving optimization problem (15)

  $\{\mathcal{G}^{(k)}, \mathcal{L}^{(k)}\}_{k=1}^K$ = `RandomSampling`$(\{p_j\}_{j=1}^M, \{\mathbf{L}_j\}_{j=1}^M)$

Part 2 `DecentralizedTraining()`**:**

  for $k \in \{0, 1, \ldots, K-1\}$ **at worker** $i \in \mathcal{V}$ **in parallel do**

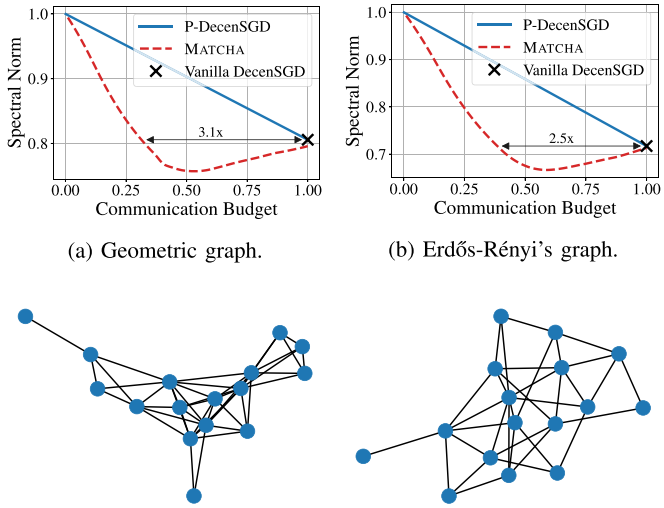
    $\mathbf{x}_i^{(k+\frac{1}{2})}$ = `LocalUpdate`$(\mathbf{x}_i^{(k)})$

    Obtain mixing matrix $\mathbf{W}^{(k)} = \mathbf{I} - \alpha\mathcal{L}^{(k)}$

    Send & Receive local models to/from all neighbors in $\mathcal{G}^{(k)}$

    Get next iterate by mixing local models: $\mathbf{x}_i^{(k+1)} = \sum_{p=1}^m W_{pi}^{(k)} \mathbf{x}_p^{(k+\frac{1}{2})}$

  end



(a) Geometric graph. (b) Erdős-Rényi's graph.

Fig. 3. Examples on how the spectral norm ρ varies over communication budget in MATCHA. In both (a) and (b), there are 16 worker nodes. MATCHA typically costs $2-3\times$ less communication time than vanilla DecenSGD (black crosses) while maintaining the exactly same or even lower value of ρ (i.e., same or better error upper bound).

general, a smaller value of ρ leads to a smaller error bound (formally stated in Theorem 2 of Section V).

In the following theorem, we optimize α by formulating a semi-definite programming problem and show that for MATCHA with arbitrary communication budget $C_b > 0$, one can always find a value of α for which the resulting spectral norm $\rho < 1$.

Theorem 1: Let $\{\mathcal{L}^{(k)}\}$ denote the sequence of Laplacian matrices generated by MATCHA with arbitrary communication budget $C_b > 0$ for a connected base graph $\mathcal{G}$. If the mixing matrix is defined as $\mathbf{W}^{(k)} = \mathbf{I} - \alpha\mathcal{L}^{(k)}$, then there exists a range of α such that $\rho = \|\mathbb{E}[\mathbf{W}^{(k)\top}\mathbf{W}^{(k)}] - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\|_2 < 1$, which guarantees the convergence of MATCHA to a stationary point.

The value of α can be obtained by solving the following semi-definite programming problem:

$$\begin{aligned} \min_{\rho,\alpha,\beta} \quad & \rho, \\ \text{subject to} \quad & \alpha^2 - \beta \le 0, \\ & \mathbf{I} - 2\alpha\overline{\mathbf{L}} + \beta\left[\overline{\mathbf{L}}^2 + 2\widetilde{\mathbf{L}}\right] - \frac{1}{m}\mathbf{1}\mathbf{1}^\top \preceq \rho\mathbf{I} \end{aligned} \tag{15}$$

where β is an auxiliary variable, $\overline{\mathbf{L}} = \sum_{j=1}^M p_j\mathbf{L}_j$ and $\widetilde{\mathbf{L}} = \sum_{j=1}^M p_j(1-p_j)\mathbf{L}_j$.

Similar to the optimization problem (13), the semi-definite programming problem (15) needs to be solved only once at the beginning of training, and this additional computation time (few seconds or even less) is negligible compared to the total training time (hours).

Note that the activation probabilities in MATCHA implicitly influence the spectral norm as well. Ideally, given a communication budget, one should jointly optimize p_i's and α via a formulation like (15). However, the resulting optimization problem is non-convex and cannot be solved efficiently. Therefore, in MATCHA, we separately optimize the p_i's and the parameter α. Optimizing p_i's via (13) can be thought of as minimizing an upper bound of the spectral norm ρ.

Dependence on Communication Budget C_b: While it is difficult to get the analytical form of ρ in terms of the communication budget C_b, in Fig. 3, we present some numerical results obtained by solving the optimization problems (13) and (15) in the TDM communication model. Recall that a lower spectral norm ρ means better error-convergence in terms of iterations. Observe that one can always find a value of C_b that preserves the same spectral norm as vanilla DecenSGD but only takes $2-3\times$ less communication time. By setting a proper budget (for instance $C_b \approx 0.5$ in Fig. 5(b)), MATCHA*can have even lower spectral norm than vanilla DecenSGD. In other words,* MATCHA*achieves a win-win in the error-runtime trade-off – it gives a $2-3\times$ communication delay reduction as compared to vanilla DecenSGD but with the same or even better convergence guarantee!* The numerical results Fig. 3 can serve as a guideline when selecting C_b in practice. These theoretical findings are corroborated by extensive experiments in Section VI.

V. ERROR CONVERGENCE ANALYSIS

In this section, we provide a theoretical analysis of how the error convergence of MATCHA depends on the spectral norm and how it compares to the convergence of vanilla DecenSGD.

To facilitate the analysis, we define the averaged iterate as $\overline{\mathbf{x}}^{(k)} = \frac{1}{m}\sum_{i=1}^m \mathbf{x}_i^{(k)}$ and the lower bound of the objective function as F_{inf}. Since, we focus on general non-convex loss functions, the quantity of interest is the averaged gradient norm: $\frac{1}{K}\sum_{k=1}^K \mathbb{E}[\|\nabla F(\overline{\mathbf{x}}^{(k)})\|^2]$. When it approaches zero, the algorithm converges to a stationary point. The analysis is centered around the following common assumptions.

Assumption 1: Each local objective function $F_i(\mathbf{x})$ is differentiable and its gradient is L-Lipschitz: $\|\nabla F_i(\mathbf{x}) - \nabla F_i(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|, \forall i \in \{1,2,\ldots,m\}$.

Assumption 2: Stochastic gradients at each worker node are unbiased estimates of the true gradient of the local objectives: $\mathbb{E}[g(\mathbf{x}_i^{(k)})|\mathcal{F}^{(k)}] = \nabla F_i(\mathbf{x}_i^{(k)}), \forall i \in \{1,2,\ldots,m\}$, where $\mathcal{F}^{(k)}$ denotes the sigma algebra generated by noise in the stochastic gradients and the graph activation probabilities until iteration k.

Assumption 3: The variance of stochastic gradients at each worker node is uniformly bounded: $\mathbb{E}[\|g_i(\mathbf{x}_i^{(k)}) - \nabla F_i(\mathbf{x}_i^{(k)})\|^2|\mathcal{F}^{(k)}] \leq \sigma^2, \forall i \in \{1,2,\ldots,m\}$.

Assumption 4: The deviation of local objectives' gradients are bounded by a non-negative constant: $\frac{1}{m}\sum_{i=1}^m \|\nabla F_i(\mathbf{x}) - \nabla F(\mathbf{x})\|^2 \leq \zeta^2$.

Assumptions 1 to 3 are common assumptions used for SGD analysis [4]. Assumption 4 is a standard assumption in distributed optimization to measure the dissimilarity among local gradients. It has been widely used in most related works, such as [16], [44] and many others. There is recent trend to relax Assumption 4 to let it hold only for the optimum point [45]. Although this relaxation can offer us a less restrictive assumption, it does not change the convergence rate at all and will not provide new insights. Therefore, we choose to conduct analysis under Assumption 4, which is enough to provide a reasonable convergence guarantee for MATCHA.

We first provide a non-asymptotic convergence guarantee for MATCHA, the proof of which is provided in the Appendix.

Theorem 2 (Non-asymptotic Convergence of MATCHA): Suppose that all local models are initialized with $\overline{\mathbf{x}}^{(1)}$ and $\{\mathbf{W}^{(k)}\}_{k=1}^K$ is an i.i.d. mixing matrix sequence generated by MATCHA. Under Assumptions 1 to 4, and if the learning rate satisfies $\eta L \leq \min\{1, (\sqrt{\rho^{-1}}-1)/4\}$, where ρ is the spectral norm (i.e., largest singular value) of matrix $\mathbb{E}[\mathbf{W}^{(k)\top}\mathbf{W}^{(k)}] - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$, then after K iterations, the expected gradient norm $\frac{1}{K}\sum_{k=1}^K \mathbb{E}[\|\nabla F(\overline{\mathbf{x}}^{(k)})\|^2$ will be upper bounded by

$$\left(\frac{2[F(\overline{\mathbf{x}}^{(1)}) - F_{inf}]}{\eta K} + \frac{\eta L \sigma^2}{m}\right)\frac{1}{1-2D} + \frac{2\eta^2 L^2 \rho}{1-\sqrt{\rho}}\left(\frac{\sigma^2}{1+\sqrt{\rho}} + \frac{3\zeta^2}{1-\sqrt{\rho}}\right)\frac{1}{1-2D} \tag{16}$$

where $D = 6\eta^2 L^2 \rho/(1-\sqrt{\rho})^2 < 1/2$. It is guaranteed that $\rho < 1$ for arbitrary communication budget $C_b > 0$.

Note that if $\mathbf{W}^{(k)} = \mathbf{1}\mathbf{1}^\top/m$, then $\rho = 0$ and the error bound in Theorem 2 reduces to that for fully synchronous SGD derived in [4]. When $\mathbf{W}^{(k)}$ is fixed across iterations, then Theorem 2 reduces to the case of vanilla DecenSGD and recovers the bound in [8], [16]. Theorem 2 reveals that the only difference in the optimization error upper bound between MATCHA and vanilla DecenSGD is the value of spectral norm ρ. A smaller value of ρ yields a lower optimization error bound.

Furthermore, if the learning rate is configured properly, MATCHA can achieve a linear speedup in terms of number of worker nodes.

Corollary 1 (Linear speedup): Under the same conditions as Theorem 2, if the learning rate is set as $\eta = \sqrt{\frac{m}{K}}$, then after total K iterations, we have

$$\frac{1}{K}\sum_{k=1}^K \mathbb{E}\left[\left\|\nabla F\left(\overline{\mathbf{x}}^{(k)}\right)\right\|^2\right] = \mathcal{O}\left(\frac{1}{\sqrt{mK}}\right) + \mathcal{O}\left(\frac{m}{K}\right) \tag{17}$$

where all the other constants are subsumed in $\mathcal{O}$.

It is worth noting that when the total iteration K is sufficiently large ($K \geq m^3$), the convergence of MATCHA will be dominated by the first term $1/\sqrt{mK}$, matching the same rate as vanilla DecenSGD and fully synchronous SGD. Using the same technique as in Corollary 1 and existing literature [29], one can also obtain the convergence rate of the consensus error $\mathbb{E}\|\overline{\mathbf{x}} - \mathbf{x}_i\|^2$ from the intermediate steps in the proof of Theorem 2. Due to space limitations, we omit this result here.

VI. EXPERIMENTAL RESULTS

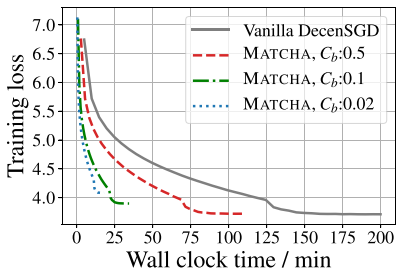
In this section, we evaluate the performance of MATCHA in multiple deep learning tasks: (1) image classification on CIFAR-10 and CIFAR-100 [46]; (2) Language modeling on Penn Treebank corpus (PTB) dataset [47]. More specifically,

- CIFAR-10 and CIFAR-100 consist of 60,000 color images in 10 and 100 classes, respectively. We set the initial learning rate as 0.8 and it decays by 10 after 100 and 150 epochs.[2] The mini-batch size per worker node is 64. We train vanilla DecenSGD for 200 epochs and all other algorithms for the same wall-clock time as vanilla DecenSGD.
- The PTB dataset contains 923,000 training words. A two-layer LSTM with 1500 hidden nodes in each layer [48] is adopted. We set the initial learning rate as 40 and it decays by 4 when the training procedure saturates. The mini-batch size per worker node is 10. The embedding size is 1500. All algorithms are trained for 40 epochs.

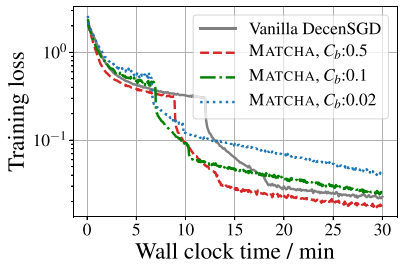
All training datasets are evenly partitioned over a network of workers. The learning rate is fine-tuned for vanilla DecenSGD and then used for all other algorithms, since we treat MATCHA as an in-place replacement of vanilla DecenSGD. Note that this will in fact disadvantage MATCHA in the comparison with vanilla DecenSGD. Moreover, each node is equipped with one NVIDIA TitanX Maxwell GPU and has a 40 Gbps (5000 MB/s) Ethernet interface. MATCHA is implemented with PyTorch and MPI4Py and uses the TDM communication model. Our code is made public here: github.com/JYWa/MATCHA.

Error-versus-wallclock time Speed-up Achieved by MATCHA: In Fig. 4, we evaluate the performance of MATCHA with various communication budgets ($2\%, 10\%, 50\%$ of vanilla DecenSGD). The base communication topology is shown in Fig. 2. From Figs. 1(e), 4(e) and 4(f), one can observe that when the communication budget is set to 0.5 (reducing expected communication time per iteration by 50%), MATCHA has nearly identical training losses as vanilla DecenSGD at every epoch. This empirical finding reinforces the claim in Section 5 regarding the similarity of the algorithms' performance in terms of iterations. When we continue to reduce the communication budget, MATCHA attains significantly faster convergence with respect to wall-clock time in communication-intensive tasks. In particular, on CIFAR-100

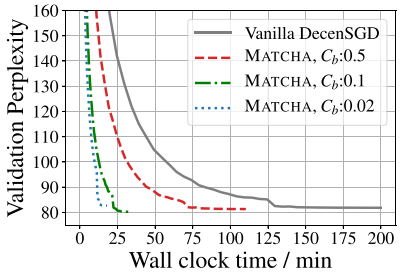
[2]One epoch means one traverse over the whole dataset



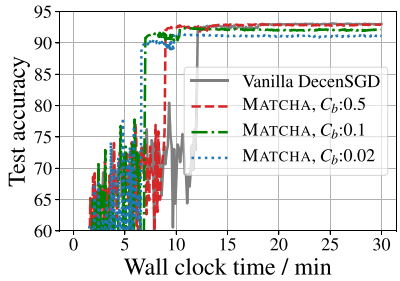
(a) Training loss versus time on PTB.



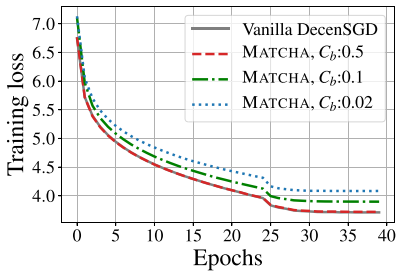
(b) Training loss versus time on CIFAR-10.



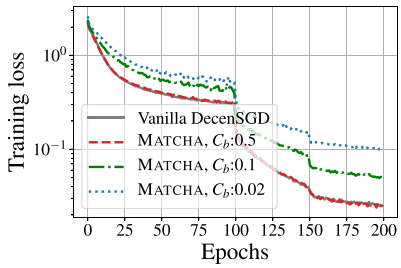
(c) Test accuracy versus time on PTB.



(d) Test accuracy versus time on CIFAR-10.



(e) Training loss versus epoch on PTB.



(f) Training loss versus epoch on CIFAR-10.

Fig. 4. Varying communication budgets C_b in MATCHA. The base communication topology is the 8-node topology in Fig. 1(a). As predicted by the theoretical result in Fig. 1(c), when the communication budget is 0.5, MATCHA has nearly the same loss-versus-iteration curves as vanilla DecenSGD (see (e) and (f)) but requires only half communication time per iteration.

(see Fig. 1(d)), MATCHA with $C_b = 0.02$ exhibits a $5\times$ reduction in wall-clock time than vanilla DecenSGD to reach a training loss of 0.1.

Effect of the Base Communication Topology: To further demonstrate the applicability of MATCHA to arbitrary graphs, we evaluate it on different topologies with varying levels of connectivity. In Fig. 5, we present experimental results on random geometric graph topologies that have different maximal degrees. In particular, in each experiment, we tune the communication budget C_b according to the numerical results like Fig. 3 so that it yields the lowest spectral norm. When the maximal degree is 10 (see Figs. 5(e), 5(h) and 5(k)), MATCHA with communication budget $C_b = 0.4$ not only reduces the mean communication time per iteration by $1/0.4 = 2.5\times$ but also has lower error than vanilla DecenSGD. This result corroborates the corresponding spectral norm versus communication budget curve in Fig. 3.

Moreover, note that for denser graphs (see Figs. 5(f), 5(i) and 5(l)), one can set a much smaller C_b and obtain a greater communication reduction. MATCHA takes less and less time to achieve a training loss of 0.1 when the base topology becomes denser, in contrast to vanilla DecenSGD. This is because denser graphs tend to have more redundant links and allow more flexibility in scheduling.

Comparison to Periodic DecenSGD: A naive way to reduce the communication time per iteration is to use the whole base graph for synchronization after every few iterations [8], [17]. Instead, in MATCHA, we allow different matchings to have different communication frequencies. In Fig. 5, we evaluate the performance of these two algorithms on three random geometric graphs with 16 nodes. Similar to the theoretical simulations in Fig. 3, the results in Fig. 5 show that given a fixed communication budget C_b, MATCHA consistently achieves better convergence rate than periodic DecenSGD (see Figs. 5(g) to 5(i)) while preserving the same runtime per iteration (see Figs. 5(d) to 5(f)). Moreover, the same phenomenon also appears on the other two training tasks (PTB and CIFAR-100). The results are presented in Fig. 6.

Extension to Other Decentralized Tasks: It is worth highlighting that MATCHA not only speeds up vanilla DecenSGD but also can be viewed as a generic tool to improve the

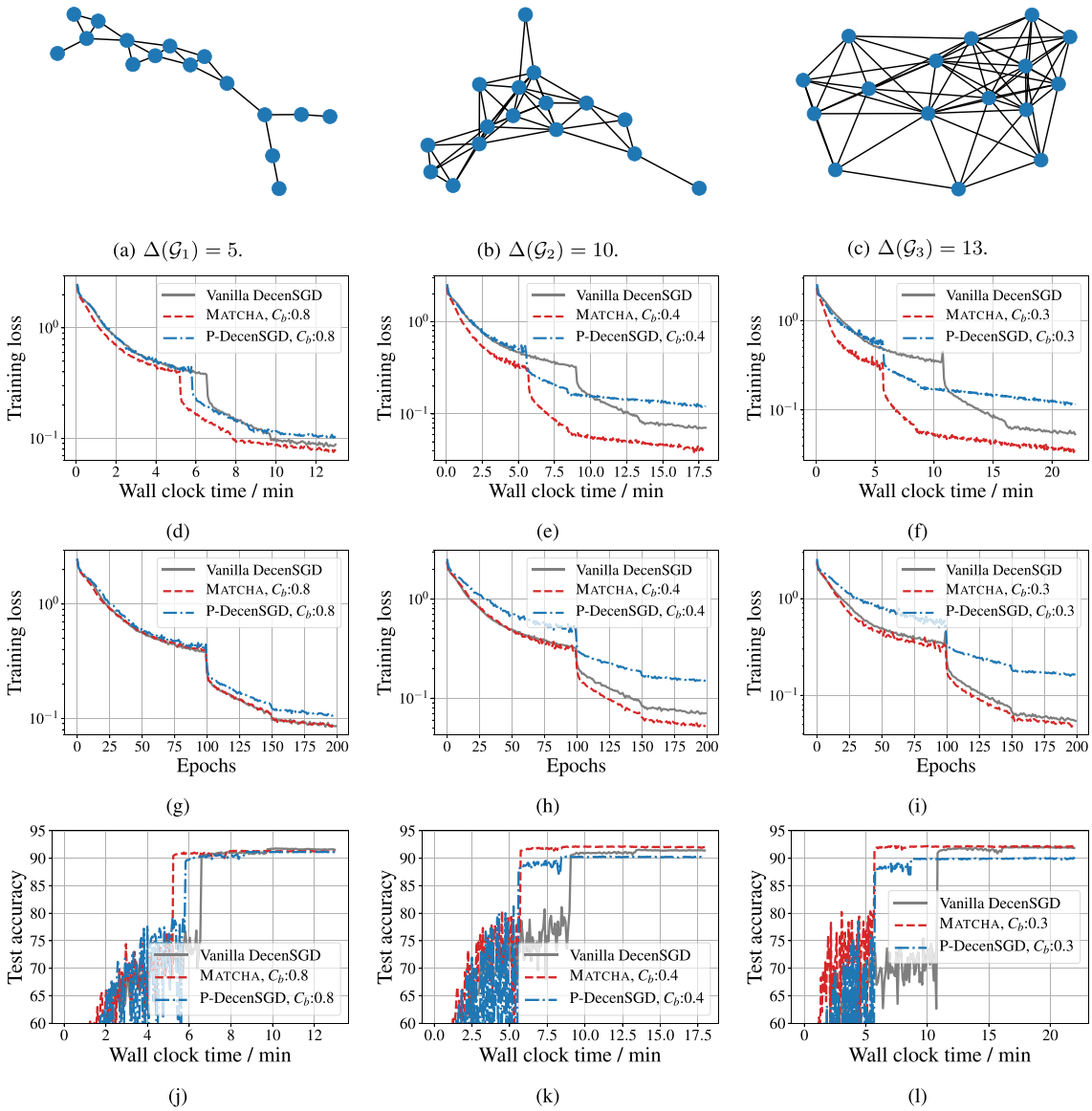


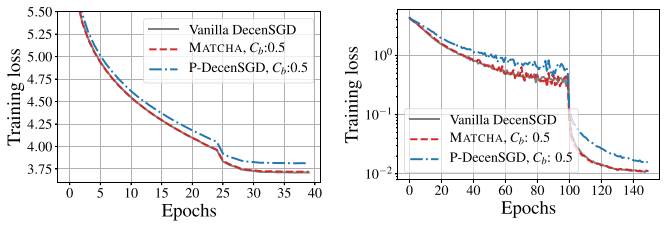
Fig. 5. Performance comparison on three random-generated geometric graphs with 16 nodes and different levels of connectivity. We train a ResNet-50 on CIFAR-10 dataset and decay the learning rate by 10 when training saturates. Each column corresponds to the results on one graph. In particular, the base topology used in (d) and (e) is the same as Fig. 5(b), which suggests that MATCHA can achieve lower spectral norm than vanilla DecenSGD when $C_b \geq 0.3$. As a consequence, (g) and (h) show that MATCHA can reach a much lower training loss than vanilla DecenSGD.

communication efficiency of any other decentralized computation task over an arbitrary network. As an example, we implement MATCHA on the top of CHOCO-SGD [27], [45], which is the state-of-the-art method to compress inter node transmissions in decentralized SGD. While CHOCO-SGD already compresses the transmitted data size to 1%, MATCHA can further achieve about $2\times$ speedup in wall-clock time over CHOCO-SGD without hurting the iteration-wise convergence (see Fig. 7).

VII. RELATED PRIOR WORK

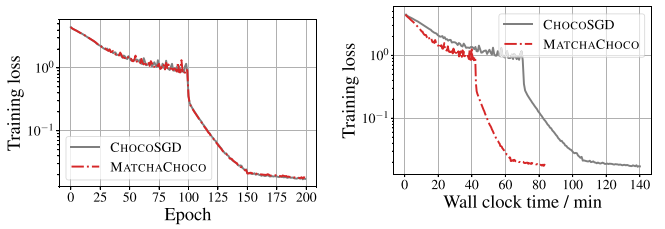
Sparsifying the Base Topology via Gossip: The idea of saving communication delay per iteration by using sparse sub-graphs of the base network topology is not new. It has been studied in the context of gossip algorithms for distributed optimization [18], [30], [31], where a subset of links are activated at random in each iteration. But the selection of these links is agnostic to its effect on the network connectivity and communication time. In contrast, to the best of our knowledge, MATCHA is the first work to tune the communication frequency of each link separately by accounting for its effect on the graph connectivity and the communication delay per iteration.

Hand-designed Network Topology Sequence: Some recent works on communication-efficient decentralized SGD use a hand-designed sequence of sub-graphs assuming that any pair of nodes can communicate with each other in the base topology. For example, [16] shows that decentralized SGD with a ring topology can sharply reduce the time spent per iteration as compared to centralized SGD (via a parameter server or usign AllReduce). Building on this idea, [29] proposes stochastic gradient push,



(a) LSTM on Penn Treebank. (b) WideResNet on CIFAR-100.

Fig. 6. Comparison of MATCHA and P-DecenSGD. The base communication topology is given in Fig. 1(a). While MATCHA has nearly identical error-convergence to vanilla DecenSGD, P-DecenSGD performs consistently worse in all tasks. Note that MATCHA and P-DecenSGD have the same average communication time per iteration.



(a) Training loss versus epoch. (b) Training loss versus wall-clock time.

Fig. 7. MATCHA can be directly applied to other decentralized algorithms, such as CHOCO-SGD [27], which uses gradient compression techniques to save communication delay in decentralized training. Here, we use the Top-1% compressor in CHOCO-SGD and train a WideResNet on CIFAR-100 dataset. The base node topology is the same as Fig. 1(a).

which uses a sequence of dynamic *directed* exponential graphs to reduce the communication delay. Having a fully-connected base topology gives them the flexibility to hand-design this sequence and ensure fast consensus. It is unclear how to generalize this idea to an arbitrary network topology where certain inter-node links may not exist due to practical constraints. MATCHA provides a general, principled method to identify communication-efficient subgraphs for *an arbitrary given base topology*.

Asynchronous Decentralized SGD: [28] develops an asynchronous version of decentralized SGD [16] called AD-PSGD, where workers asynchronously average their updated local model with a randomly selected neighbor. In order to avoid deadlocks, the asynchronous communication has to be conducted over a hand-designed sequence of bipartite sub-graphs of a fully-connected base topology. Moreover, asynchronous aggregation results in gradient staleness, which can lead to worse error-versus-iterations convergence. Although we focus on synchronous aggregation in this paper, MATCHA is a general link scheduling technique that is independent of the gradient aggregation mechanism (asynchronous or synchronous) and thus it is complementary to AD-PSGD.

Gradient Compression or Quantization Techniques: By reducing the frequency of inter-node communication, MATCHA effectively performs more local SGD updates at each node between model synchronization steps. Thus, MATCHA belongs to the class of local-update SGD methods recently studied in [7], [8], [25], [44]. An orthogonal way of reducing inter-node communication is to compress or quantize inter-node model updates [27], [32], [49], [50], [51]. These gradient compression techniques reduce the amount of data that is transmitted per round rather than the communication frequency. MATCHA can be easily combined with these complementary compression techniques to give a further reduction in the overall communication time, as shown in Section VI. As a sidenote, compression techniques incur an encoding/decoding overhead at each iteration. MATCHA does not incur such overhead at runtime – the sequence of subgraphs is pre-determined before the training starts.

Gradient Tracking Methods: There are extensive literature studying gradient tracking methods to accelerate the convergence of vanilla DecenSGD with respect to iterations [34], [52], [53]. However, this line of works is orthogonal to our paper. The goal of MATCHA is to reduce the communication delay per iteration in vanilla DecenSGD while maintaining its convergence rate. In terms of update rules, the gradient tracking methods changes how worker nodes perform model updates. But our method MATCHA changes how different local models (more generally, local information) are synchronized together.

VIII. CONCLUDING REMARKS

In this paper, we consider the problem of speeding-up decentralized SGD, where nodes connected by an arbitrary network topology seek to train a machine learning model by only exchanging updates with neighboring nodes. Although decentralized SGD has been extensively studied in distributed optimization literature, most previous works focus on analyzing and improving the error convergence with respect to the number of iterations. The key novelty of this work is that we take the system-aware approach of accounting for how the network topology affects the communication delay per iteration and improving the true convergence speed measured in terms of error versus wallclock runtime.

The idea of more frequent activation connectivity-critical sub-graphs, which is at the core of MATCHA, is a nascent concept with plenty of room for future work. MATCHA currently finds matching decomposition and the activation probabilities centrally before training begins. In order to use MATCHA in large and dynamic network topologies, we plan to design decentralized algorithms to find matchings and activation probabilities. Also, instead of activating the matchings independently, we could identify correlated subsets that should be activated together and consider heterogeneous and/or random communication link delays. Finally, so far we consider an undirected topology because each link represents a two-way transfer of information (sending and receiving the local models) between neighbors. Considering a directed graph can give more flexibility in scheduling the inter-node model transfers and enable overlapping communication and computation, as done successfully in [29] for a fully-connected base topology.

ACKNOWLEDGMENT

The experiments were conducted on the ORCA cluster provided by the Parallel Data Lab at CMU.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[2] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[3] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[5] J. Dean et al., "Large scale distributed deep networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1223–1231, 2012.

[6] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "Firecaffe: Near-linear acceleration of deep neural network training on compute clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2592–2600.

[7] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Representations*, 2019.

[8] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," *J. Mach. Learn. Res.*, vol. 22, no. 213, pp. 1–50, 2021.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.* 2017, pp. 1273–1282.

[10] P. Kairouz et al., "Advances and open problems in federated learning," *Foundations Trends Mach. Learn.*, vol. 14, no. 12, pp. 1–210, 2021.

[11] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[13] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2737–2745.

[14] J. Zhang, I. Mitliagkas, and C. Re, "YellowFin and the art of momentum tuning," in *Proc. 2nd MLSys Conf.*, 2019.

[15] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD," in *Proc. 21st Int. Conf. Artif. Intell. Statist.,* 2018, pp. 803–812.

[16] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 5336–5346.

[17] K. Tsianos, S. Lawlor, and M. G. Rabbat, "Communication/computation tradeoffs in consensus-based distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1943–1951, 2012.

[18] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.

[19] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. Optim.*, vol. 26, no. 3, pp. 1835–1854, 2016.

[20] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2834–2848, Jun. 2018.

[21] Z. J. Towfic, J. Chen, and A. H. Sayed, "Excess-risk of distributed stochastic learners," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5753–5785, Oct. 2016.

[22] D. Jakovetic, D. Bajovic, A. K. Sahu, and S. Kar, "Convergence rates for distributed stochastic optimization over random networks," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 4238–4245.

[23] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 2740–2749.

[24] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5906–5916.

[25] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proc. 2nd Conf. Mach. Learn. Syst.*, 2019, pp. 212–229.

[26] S. Gupta, W. Zhang, and F. Wang, "Model accuracy and runtime tradeoff in distributed deep learning: A systematic study," in *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 171–180.

[27] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3478–3487.

[28] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3043–3052.

[29] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 344–353.

[30] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. Netw.*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.

[31] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1291–1306, Jun. 2011.

[32] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 7652–7662.

[33] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[34] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4848–4856.

[35] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with accelerated convergence," 2019, *arXiv:1912.04230*.

[36] B. Bollobás, *Modern Graph Theory*, vol. 184. Berlin, Germany: Springer, 2013.

[37] M. Fiedler, "Eigenvalues of nonnegative symmetric matrices," *Linear Algebra Appl.*, vol. 9, pp. 119–142, 1974.

[38] J. Misra and D. Gries, "A constructive proof of Vizing's theorem," *Inf. Process. Letters. Citeseer*, vol. 41, pp. 131–133, 1992.

[39] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. AC-31, no. 9, pp. 803–812, Sep. 1986.

[40] S. Kar and J. M. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3315–3326, Jul. 2008.

[41] S. Kar, S. Aldosari, and J. M. Moura, "Topology for distributed inference on graphs," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2609–2613, Jun. 2008.

[42] P. Erdős and R. J. Wilson, "On the chromatic index of almost all graphs," *J. Combinatorial Theory, Ser. B*, vol. 23, no. 2-3, pp. 255–257, 1977.

[43] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.

[44] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD for non-convex optimization with faster convergence and less communication," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019.

[45] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkgGCkrKvH

[46] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[47] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computat. Linguistics-Assoc. Comput. Linguistics* vol. 19, no. 2, pp. 313–330, 1993.

[48] O. Press and L. Wolf, "Using the output embedding to improve language models," in *Proc. 15th Conf. Eur. Chapter Assoc. Computat. Linguistics*, vol. 2, pp. 157–163, 2017.

[49] H. Tang et al., "Deepsqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *Proc. Int. Conf. Mach. Learn.*, 2019.

[50] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9850–9861.

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

[52] A. Mokhtari and A. Ribeiro, "DSA: Decentralized double stochastic averaging gradient algorithm," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2165–2199, 2016.

[53] H. Li, L. Zheng, Z. Wang, Y. Yan, L. Feng, and J. Guo, "S-DIGing: A stochastic gradient tracking algorithm for distributed optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 1, pp. 53–65, Feb. 2022.

Jianyu Wang received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2017, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2022. He is currently a Research Scientist with Meta. He was a Research Interns with Google Research (2020, 2021) and Facebook AI research in 2019. His research interests include federated learning, distributed optimization, and systems for large-scale machine learning. His research has been supported by Qualcomm Ph.D. Fellowship in 2019.

Anit Kumar Sahu received the B.Tech. degree in electronics and electrical communication engineering and the M.Tech. degree in telecommunication systems engineering from the Indian Institute of Technology, Kharagpur, Kharagpur, India, in May 2013, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2018. He is currently a Senior Applied Scientist with Amazon Alexa AI. His research interests include federated learning, self-learning, and stochastic optimization. He was the recipient of the 2019 A.G. Jordan Award from the Department of Electrical and Computer Engineering at CMU.

Gauri Joshi (Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology (IIT) Bombay, Mumbai, India, in 2010, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2016. She is currently an Associate Professor with ECE Department, Carnegie Mellon University, Pittsburgh, PA, USA. She was a Research Staff Member with IBM T. J. Watson Research Center during 2016–2017. Her research interests include federated learning, distributed optimization, and coding theory. She was the recipient of the NSF CAREER Award in 2021, ACM Sigmetrics Best Paper Award in 2020, and the Institute Gold Medal of IIT Bombay in 2010.

Soummya Kar (Fellow, IEEE) received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, Kharagpur, India, in May 2005, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2010. He is currently a Professor of electrical and computer engineering with Carnegie Mellon University. From June 2010 to May 2011, he was with the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, as a Postdoctoral Research Associate. He has authored or coauthored more than 250 articles in journals and conference proceedings and holds multiple patents in his research field which include decision-making in large-scale networked systems, stochastic systems, and machine learning.