Adaptive Configuration for Heterogeneous Participants in Decentralized Federated Learning

Yunming Liao[1,2] *Yang Xu[1,2] Hongli Xu[1,2] Lun Wang[1,2] Chen Qian[3]
[1]School of Computer Science and Technology, University of Science and Technology of China, China
[2]Suzhou Institute for Advanced Study, University of Science and Technology of China, China
[3]Department of Computer Science and Engineering, Jack Baskin School of Engineering, University of California, Santa Cru.

***Abstract*—Data generated at the network edge can be processed locally by leveraging the paradigm of edge computing (EC). Aided by EC, decentralized federated learning (DFL), which overcomes the single-point-of-failure problem in the parameter server (PS) based federated learning, is becoming a practical and popular approach for machine learning over distributed data. However, DFL faces two critical challenges, *i.e.*, system heterogeneity and statistical heterogeneity introduced by edge devices. To ensure fast convergence with the existence of slow edge devices, we present an efficient DFL method, termed FedHP, which integrates adaptive control of both local updating frequency and network topology to better support the heterogeneous participants. We establish a theoretical relationship between local updating frequency and network topology regarding model training performance and obtain a convergence upper bound. Upon this, we propose an optimization algorithm, that adaptively determines local updating frequencies and constructs the network topology, so as to speed up convergence and improve the model accuracy. Evaluation results show that the proposed FedHP can reduce the completion time by about 51% and improve model accuracy by at least 5% in heterogeneous scenarios, compared with the baselines.**

***Index Terms*—Edge Computing, Decentralized Federated Learning, Peer-to-Peer, Heterogeneity.**

I. Introduction

The past few years have witnessed remarkable advancements in mobile computing and the Internet of Things. Mobile devices constantly generate massive data, such as photos and voices, which are of great value for developing intelligent applications [1], [2]. Meanwhile, edge computing (EC) systems have been deployed to store data locally and push more computing power to the network edge for data analysis [3]–[5]. With the emergence of EC, federated learning (FL) [6]–[9] has been developed to perform distributed model training at the network edge or end devices close to the data source. FL does not only prevent personal privacy from being exposed but also fully utilizes plenty of computation resources at the network edge.

Traditional FL requires a parameter server (PS) to communicate with the edge nodes (*i.e.*, participants) [7], [10], [11], and involves model transmission from a certain (possibly large) number of nodes for model aggregation, which brings enormous amount of traffic workload to the PS. Consequently, the PS may become the system bottleneck, leading to the risk of network congestion and poor scalability. In comparison, decentralized federated learning (DFL) [6], [12]–[14] is becoming an attractive solution by disseminating information through peer-to-peer (P2P) communication, to avoid the communication bottleneck at the centralized server. Moreover, since there is no need to forward the local models from nodes to the PS, the potential of single point failure can be avoided and the system scalability will be significantly improved. This work focuses on DFL and explores its communication and computation efficient learning strategies so as to enhance model training at the network edge.

There are two important features in EC systems making it difficult to implement efficient DFL. 1) *System Heterogeneity.* In EC, the capabilities of edge nodes are usually limited and heterogeneous [1], [15]. There could be a tenfold difference in computing capabilities (*e.g.*, CPU frequency) or communication capabilities (*e.g.*, bandwidth, throughput) among edge nodes [16]–[18]. Due to system heterogeneity, fast edge nodes may have to wait for the stragglers in a synchronous manner, which incurs non-negligible waiting time and deteriorates training efficiency. 2) *Statistical Heterogeneity.* The local data collected by edge nodes usually depends on their functions and/or locations, resulting in non- independent and identically distributed (non-IID) local data across all edge nodes. The non-IID data (known as statistical heterogeneity) will decelerate the convergence rate and even compromise the accuracy of trained models [2], [19], [20].

In general, edge nodes always update the models with their globally-synchronized neighbor models, which is proven to achieve similar convergence rate (*w.r.t.* the number of rounds/iterations) as the parallel mini-batch SGD, and will converge to satisfied solutions with high test accuracy [13]. Besides, given limited capabilities on edge nodes, a synchronous DFL method, named LD-SGD [21], has been proposed, which alternates the frequencies of local updating and global updating to significantly reduce the communication resource consumption. As for statistical heterogeneity, Onoszko *et al.* [22] proposed a synchronous method named performance-based neighbor selection (PENS), where nodes with similar data distributions communicate with each other. However, the synchronization barrier of these methods often leads to idle time for staying and waiting for the stragglers (*i.e.*, the slow participants) before model aggregation, especially in the heterogeneous system. Moreover, PENS always suffers from more computing time for neighbor selection (*i.e.*, network topology construction) and model training at each communication round due to system heterogeneity. Although the asynchronous DFL [23]–[26] contributes to addressing the challenge of system heterogeneity and accelerating the convergence rate *w.r.t.* time, each node receives and aggregates the stale models, which amplifies the negative impact of non-IID data on test accuracy and even leads to model divergence [17]. Herein, we focus on the synchronous implementation of DFL to cope with the potential problems, such as delayed convergence time and compromised model accuracy, caused by system and statistical heterogeneities.

In this paper, we investigate the benefits of controlling local updating frequency and network topology, which are jointly optimized to adequately address the two heterogeneity issues for synchronous DFL. Unlike the identical local updating frequency and fixed neighbors (*i.e.*, network topology) for all edge nodes [12], [21], we explore to adaptively assign different local updating frequencies for heterogeneous nodes and adjust network topology to eliminate the idle time incurred by synchronization. The coupled relationship between local updating frequency and network topology will be elaborated in Sec. II-D. According to our theoretical analysis and pretest in Sec. III, a relatively smaller or larger local updating frequency will lead to more communication rounds or lower model accuracy. Therefore, as training progresses, it is necessary yet challenging to simultaneously determine the appropriate local updating frequencies and neighbors for different edge nodes so as to well balance the trade-off between convergence rate and model accuracy. The main contributions of this paper are summarized as follows:

- We design an efficient DFL method, called FedHP, which integrates adaptive control of local updating frequency and network topology to better overcome the challenges of system and statistical heterogeneities in EC systems.
- We theoretically analyze the convergence rate and obtain a convergence upper bound related to local updating frequency and network topology. Upon this, we propose a control algorithm, which adaptively determines appropriate local updating frequencies and neighbors for different edge nodes, so as to speed up training and improve the model accuracy.
- The performance of our method is evaluated through extensive simulation experiments. The evaluation results show that our method can reduce the convergence time by about 51% and improve model accuracy by at least 5% in heterogeneous scenarios, compared to existing DFL methods.

The rest of this paper is organized as follows. Sec. II formalizes the optimization problem in FedHP. Sec. III gives the convergence analysis of FedHP. Based on the analysis, we propose an efficient algorithm in Sec. IV. Then in Sec. V, we report our experimental results. We discuss some related works in Sec. VI and conclude the paper in Sec. VII.

II. Preliminaries and Problem Formulation

A. Network Model

An EC system includes a set of distributed workers (*e.g.*, IoT devices or small base stations) $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$, with $|\mathcal{V}| = N > 1$. In DFL, the workers collaboratively train deep learning models on their local datasets, and each worker needs to exchange models with its neighbors rather than sharing its original data. A control node (*i.e.*, coordinator) is still needed to collect the global information about model training statuses and network conditions in DFL [20], [26]–[28]. However, unlike the parameter server in FL, the coordinator does not aggregate the models and hence will not become the bandwidth bottleneck. Furthermore, any worker can act as the coordinator. Since the size of these information (*e.g.*, 100-300KB [29]) is much smaller than that of model parameters, it is reasonable to ignore the cost (*e.g.*, bandwidth consumption and time cost) for information collection [30].

The P2P network topology at the h-th communication round can be expressed as a connected undirected graph $\mathcal{G}^h = (\mathcal{V}, E^h)$, where $\mathcal{V}$ denotes the worker set and E^h denotes the set of links connecting workers at communication round h. Specifically, the P2P network topology at round h can be expressed as a symmetric adjacency matrix $\mathbf{A}^h = \{a^h_{i,j} \in \{0,1\}, 1 \le i,j \le N\}$, where $a^h_{i,j} = 1$ if $e^h_{i,j} \in E^h$, otherwise 0. The neighbor set of worker i at round h is represented as $\mathcal{N}^h_i$, whose cardinality is denoted as $|\mathcal{N}^h_i| = \sum_{j \in \mathcal{N}^h_i} a^h_{i,j}$. The degree matrix $\mathbf{D}^h = \{d^h_{i,j}, 1 \le i,j \le N\}$ is defined as a diagonal matrix, where $d^h_{i,i} = |\mathcal{N}^h_i|$. Combining the adjacency matrix and the degree matrix, the Laplacian matrix $\mathbf{L}^h$ can be expressed as follows:

$$\mathbf{L}^h = \mathbf{D}^h - \mathbf{A}^h. \tag{1}$$

According to the spectral graph theory [31], $\lambda_2(\mathbf{L}^h) > 0$ if and only if the topology is connected, where $\lambda_m(\mathbf{L}^h)$ denotes the m-th smallest eigenvalue of matrix $\mathbf{L}^h$.

B. Model Training Process

In DFL, worker i updates the local model parameter x_i at the h-th communication round based on a mini-batch ξ_i sampled from its local dataset $\mathcal{D}_i$. Let $f_i(x_i)$ and $F_i(x_i;\xi_i)$ (for ease of description, written as $F_i(x_i)$) denote the local loss function and the loss function over mini-batch ξ_i, respectively. Generally, model training can be formally described as optimizing the following objective function [32]:

$$f^* := \min_{x \in \mathbb{R}^d} \; [\; f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x_i) \;], \tag{2}$$

where $f_i(x_i) := \mathbb{E}_{\xi_i \sim \mathcal{D}_i} F_i(x_i)$ and x denotes the global model parameter. This setting covers the important cases of empirical risk minimization in DFL [32].

The model will be updated by applying the decentralized stochastic gradient descent (DSGD) algorithm [33], which provides an effective way to optimize the loss function in a decentralized manner. For the mini-batch stochastic gradient descent, a gradient descent step over a mini-batch on each worker is regarded as a local iteration (or a local update). After performing one or multiple local iterations, each worker exchanges local models or gradients with its neighbors and aggregates these models. Such a training process is regarded as a communication round. $x^{h,k}_i$ denotes the local model of worker i at the k-th local iteration within communication round h. At the beginning of communication round h, by setting $x^{h,0}_i = x^h_i$, worker i updates its local model by gradient descent as follows [18], [20]:

$$x^{h,k+1}_i = x^{h,k}_i - \eta \nabla F_i(x^{h,k}_i),\ 0 \le k < \tau, \tag{3}$$

where η is the local learning rate, τ is the local updating frequency, and $\nabla F_i(x^{h,k}_i)$ is the gradient. The local updates of worker i at round h is denoted as $g^h_i = \sum_{k=0}^{\tau-1} \nabla F_i(x^{h,k}_i)$. Then the local updating of worker i can be rewritten as:

$$x^{h+1}_i = x^h_i - \eta \cdot g^h_i. \tag{4}$$

After local updating, workers send local models to their neighbors. Based on the received model parameters, worker i will aggregate these models from neighbors:

$$x_i^{h+1} = x_i^h + \sum_{j \in \mathcal{N}_i^h} w_{i,j}^h (x_j^h - x_i^h), \tag{5}$$

where $\mathcal{N}_i^h$ is the neighbor set of worker i at round h and $w_{i,j}^h, j \in \mathcal{N}_i^h$, is the mixing weight for aggregating the model of neighbor j. Defining u_{max}^h as the maximum of $|\mathcal{N}_i^h|$ over workers at round h, a simple suboptimal choice of $w_{i,j}^h$ is [34]:

$$w_{i,j}^h = \frac{1}{u_{max}^h + 1}. \tag{6}$$

C. Consensus Distance

Unlike the traditional PS architecture, there is no global model in DFL, and local models hosted by different workers are not always the same. We introduce the *consensus distance* metric to measure the discrepancy among local models [20], [32], [35]. Firstly, the consensus distance between model of worker i and model of worker j at the h-th communication round is defined as:

$$D_{i,j}^h = \left\| x_i^h - x_j^h \right\|. \tag{7}$$

Then the consensus distance between local model of worker i and "global model" (*i.e.*, the average of all workers' models) at round h is defined as:

$$D_i^h = \left\| \overline{x}^h - x_i^h \right\|, \tag{8}$$

where $\overline{x}^h = \frac{1}{N}\sum_{i=1}^N x_i^h$ denotes the average of all workers' models at round h. It is worth noting that $\overline{x}^h$ is not available in practice because there is no PS to collect all workers' models in DFL. To this end, we would estimate D_i^h using consensus distance between the local model of worker i and the models of its neighbors (*i.e.*, $D_{i,j}^h, j \in \mathcal{N}_i^h$), which will be elaborated in Sec. IV-A. Accordingly, the average consensus distance of all workers' models is:

$$D^h = \frac{1}{N}\sum_{i=1}^N D_i^h. \tag{9}$$

Similar to the weight divergence [19], [36] in the PS architectures, the consensus distance is correlated to data distribution and is the key factor that captures the joint effect of decentralization [35], which motivates us to apply consensus distance for topology construction to overcome the challenge introduced by non-IID data.

D. Relationship between Local Updating Frequency and Network Topology

In this section, we explain the coupled relationship between local updating frequencies and network topologies. On the one hand, the computing time of one local iteration and the transmission time of one model among workers are highly different due to system heterogeneity. However, in traditional synchronous schemes, local updating frequencies among workers are usually identical or fixed at each communication round. Accordingly, fast workers have to wait for slow ones, incurring non-negligible idle time and significantly reducing the training efficiency [15], [16]. Considering the heterogeneous computing capabilities of workers, before aggregation, the workers with higher computing capabilities will perform more local iterations while the workers with lower computing capabilities only perform fewer local iterations. On the other hand, data samples across all workers may be non-IID, which seriously affects the convergence rate and even compromises the accuracy of trained model [2], [19]. To deal with the statistical heterogeneity, the workers with significantly different data distributions (*i.e.*, with large consensus distance) can be connected preferentially and frequently. After that, the training performance over non-IID data can be guaranteed meanwhile the waiting time and training time among workers would be significantly reduced.

Furthermore, the local models trained with different local updating frequencies are discrepant, which requires to select suitable neighbors for model aggregation to achieve satisfied model accuracy. Meanwhile, the completion time of each communication round (including computing time and communication time) varies with dynamic network topology, which requires to assign appropriate local updating frequencies for heterogeneous workers to reduce the waiting time. Accordingly, we propose to jointly optimize the local updating frequency and network topology to address the system heterogeneity and statistical heterogeneity in DFL.

E. Problem Formulation

This section defines the problem of efficient DFL with adaptive local updating and network topology: *minimizing the training time while requiring workers to achieve a satisfied accuracy for their models*. Given a DFL task in the EC system, we need to determine the local updating frequencies and average consensus distance of all workers to minimize the training time. First, the local updating frequency and the computing time of one local iteration at the h-th communication round on worker i are denoted as τ_i^h and μ_i^h, respectively. Let $\mathbf{B}^h = \{\beta_{i,j}^h, 1 \leq i,j \leq N\}$ denote the communicating time matrix at round h, where $\beta_{i,j}^h$ is the communicating time between worker i and worker j. Therefore, the local updating time (including computing time and communication time) of worker i at round h is formulated as:

$$t_i^h = \tau_i^h \cdot \mu_i^h + \max\{\beta_{i,j}^h\} \ \forall i \in [N], \forall j \in \mathcal{N}_i^h. \tag{10}$$

In addition, the waiting time of worker i can be expressed as $t^h - t_i^h$, where $t^h = \max\{t_i^h\}$ ($\forall i \in [N]$) denotes the local updating time of the slowest worker at round h. t^h also denotes the completion time of round h. Then the average waiting time of all workers at round h can be formulated as:

$$\mathcal{W}^h = \frac{1}{N}\sum_{i=1}^N (t^h - t_i^h). \tag{11}$$

Accordingly, we formulate the problem as follows:

$$\min \sum_{h=1}^H t^h$$
$$s.t. \begin{cases} D^{h+1} \leq D_{max}^h, \\ \lambda_2(\mathbf{L}^h) > 0, \\ t_i^h = \tau_i^h \cdot \mu_i^h + \max\{\beta_{i,j}^h\}, \forall i \in [N], \forall j \in \mathcal{N}_i^h \\ \mathcal{W}^h = \frac{1}{N}\sum_{i=1}^N (t^h - t_i^h) \leq \varepsilon \end{cases} \tag{12}$$

The first inequality expresses that the average consensus distance should not exceed the predefined threshold D_{max}^h. We set D_{max}^h as the same in [35] and the details are described in Sec. IV. The second inequality ensures a connected topology in each communication round, which is essential to guarantee the training convergence [37]. The third set of equalities denotes the formulation of the local updating completion time and communication time on worker i at the h-th communication round, where $\beta_{i,j}$ denotes the communication time between worker i and worker j. The fourth set of inequalities essentially guarantees that the average waiting time of all workers at each communication round is sufficiently small, where $\varepsilon > 0$ is the time threshold, so as to mitigate the effects of the synchronization barrier. Our objective is to minimize the training time under the constraints.

III. Convergence Analysis

In this section, we analyze the model convergence rate of our method in theory and obtain a convergence upper bound related to local updating frequency and network topology. We first make the following assumptions, which are widely used in previous works [20], [37]–[39]:

Assumption 1. *(**L-smooth**) Each local objective function $f_i : \mathbb{R}^d \to \mathbb{R}$ on workers is L-smooth:*

$$\|\nabla f_i(y) - \nabla f_i(x)\|_2 \le L\,\|y - x\|_2\,, \forall x, y \in \mathbb{R}^d. \tag{13}$$

Assumption 2. *(**Unbiased Local Gradient Estimator**) Let ξ_i^h be a random local data sample at the h-th communication round on worker i. The local gradient estimator is unbiased as follows:*

$$\mathbb{E}\left[\nabla F_i\left(x_i^h, \xi_i^h\right)\right] = \nabla f_i\left(x_i^h\right). \tag{14}$$

Assumption 3. *(**Bounded gradient variance**) The variance of stochastic gradients at each worker is bounded:*

$$\mathbb{E}\,\|\nabla F_i(x_i, \xi_i) - \nabla f_i(x_i)\|_2^2 \le \sigma^2, \forall x \in \mathbb{R}^d, \forall i \in [N], \tag{15}$$

$$\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(x_i) - \nabla f(x)\|_2^2 \le \zeta^2, \forall x \in \mathbb{R}^d, \forall i \in [N]. \tag{16}$$

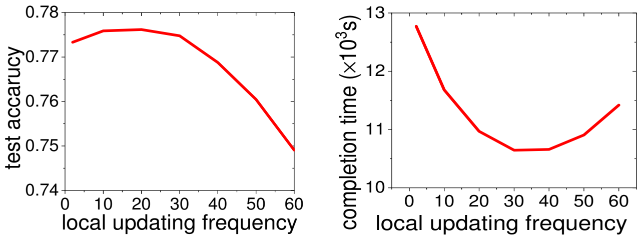
The variance in Eq. (15) denotes how far the estimated gradient over mini-batch ξ_i deviates from the true gradient of $f_i(x_i)$. In addition, ζ in Eq. (16) indicates the degree of difference between local functions on workers and the global function $f(x)$, indicating the heterogeneity of the non-IID datasets among different workers. In particular, if the data distributions across workers are IID, all functions are identical (*i.e.*, $f_i(x_i) = f_j(x_j), \forall i, j \in [N]$), thus $\zeta = 0$.

Assumption 4. *(**Spectral gap**) The weight matrix W is symmetric doubly stochastic. We define $\rho = \max\{|\lambda_2(W)|, |\lambda_N(W)|\}$ and assume $\rho < 1$.*

Lemma 1. *Under the above assumptions with $\eta \le \frac{1}{4L\tau}$, we have the following expression:*

$$\mathbb{E}f(\overline{x}^{h+1}) \le f(\overline{x}^h) - \frac{\eta\tau}{4}\left\|\nabla f(\overline{x}^h)\right\|_2^2 + \frac{\eta L^2\tau}{N}\sum_{i=1}^{N}\left\|\overline{x}^h - x_i^h\right\|_2^2 + \frac{\sigma^2\eta^2\tau^2 L}{N}, \tag{17}$$

where $\tau = \max\{\tau_i^h\}$.



(a) Accuracy *vs.* τ (b) Completion time *vs.* τ

Fig. 1: Model training on different local updating frequency τ of CIFAR-10.

Remark 1. *Summing up for all H communication rounds and rearranging the terms in Eq.* (17)*, we get:*

$$\frac{1}{H}\sum_{h=1}^{H}\left\|\nabla f(\overline{x}^h)\right\|_2^2 \le \frac{4*(f(\overline{x}^1) - f(\overline{x}^*))}{\eta\tau H} + \frac{4L^2}{NH}\sum_{h=1}^{H}\sum_{i=1}^{N}\left\|\overline{x}^h - x_i^h\right\|_2^2 + \frac{4L\eta\tau\sigma^2}{N}. \tag{18}$$

Lemma 2. *Under the above assumptions with $\frac{27L\eta^2}{(1-\rho)^2} < 1$, we have the following formulation:*

$$\sum_{h=1}^{H}\sum_{i=1}^{N}\mathbb{E}\left\|\overline{x}^h - x_i^h\right\|_2^2 \le \frac{2N\eta^2(\sigma^2 + 3\zeta^2)H}{(1-\rho)^2 - 3\eta^2L^2} + \frac{6N\eta^2}{(1-\rho)^2 - 3\eta^2L^2}\sum_{h=1}^{H}\mathbb{E}\left\|\nabla f(\overline{x}^h)\right\|_2^2. \tag{19}$$

Due to space limitations, we omit the proofs of Lemmas 1 and 2. The detail proofs are presented in [40].

Remark 2. *Inserting Eq.* (19) *into Eq.* (18)*, we obtain the following convergence bound:*

$$\frac{1}{H}\sum_{h=1}^{H}\left\|\nabla f(\overline{x}^h)\right\|_2^2 \le \frac{4*(f(\overline{x}^1) - f(\overline{x}^*))((1-\rho)^2 - 3\eta^2L^2)}{\eta\tau H((1-\rho)^2 - 27\eta^2L^2)} + \frac{8L^2\eta^2(\sigma^2 + 3\zeta^2)}{(1-\rho)^2 - 27\eta^2L^2} + \frac{(1-\rho)^2 - 3\eta^2L^2}{(1-\rho)^2 - 27\eta^2L^2}\frac{4L\eta\tau\sigma^2}{N}. \tag{20}$$

The communication topology weight matrix W (reflected by ρ), local updating frequency τ and data distribution (reflected by ζ) all have impacts on the convergence rate with Eq. (20). The sparser the topology is, the larger ρ is. For example, ρ is 0 for the fully-connected topology while ρ is 0.99 for the ring topology with 36 workers. Thus, with the increasing of topology sparsity, the above convergence bound will increase. When $\tau \le \sqrt{\frac{N(f(\overline{x}^1) - f(\overline{x}^*))}{LH\eta^2\sigma^2}}$, the above convergence bound will decrease as local updating frequency τ increases. On the contrary, when $\tau > \sqrt{\frac{N(f(\overline{x}^1) - f(\overline{x}^*))}{LH\eta^2\sigma^2}}$, the trend of convergence bound and local updating frequency is opposite. As the degree of non-IID data distribution increases (*i.e.*, larger ζ), the upper bound of Remark 2 will get looser and looser.

According to the above analysis, a very large local updating frequency may make the decentralized models converge to the local optimal solutions rather than the global optimum. However, a relatively smaller local updating frequency will lead to more communication rounds until convergence, incurring more computing time and communication time. To observe the impact of local updating frequency on model training, we conduct a pre-experiment for training AlexNet on CIFAR-10 and record the model accuracy and completion time with different local updating frequencies. As shown in Fig. 1(a), the model accuracy decreases with increasing of local updating frequency when $\tau > 27$. Besides, Fig. 1(b) shows that the completion time of model training decreases with increasing of local updating frequency when $\tau < 30$. These results are consistent with our analysis in Eq. (20). Therefore, it is critical to determine the appropriate local updating frequencies for different workers to accelerate model training.

Corollary 1. *Let the local learning rate η satisfy the following constraint:*

$$\eta = (\frac{6L}{\sqrt{(1-\rho)^2}} + \sigma N^{-\frac{1}{2}}\tau H^{\frac{1}{2}} + \zeta^{\frac{2}{3}} H^{\frac{1}{3}})^{-1}. \tag{21}$$

The convergence upper bound can be transformed as:

$$\frac{1}{H}\sum_{h=1}^{H}\mathbb{E}\|\nabla f(\overline{x}^h)\|_2^2 \le \frac{\sigma}{\sqrt{NH}} + \frac{1}{(1-\rho)^2}(\frac{\zeta}{H})^{\frac{2}{3}} + \frac{1}{H\tau^2(1-\rho)^2}. \tag{22}$$

With Corollary 1, our method can achieve a linear speedup of convergence rate $\mathcal{O}(\frac{1}{\sqrt{HN}})$ as stated in many previous works [12], [32], indicating that our method will contribute to speeding up the training without loss of convergence performance.

IV. Algorithm Design

A. Consensus Distance Estimation

We first analyze how the network topology and local updating frequency affect the consensus distance between the model of worker i and the average of all workers' models. According to the update rule in Eq. (5) and the definition in Eq. (8), the consensus distance $\|\overline{x}^{h+1} - x_i^{h+1}\|_2$ at round $h+1$ can be formulate as:

$$\begin{aligned} D_i^{h+1} &= \|\overline{x}^{h+1} - x_i^{h+1}\|_2 \\ &= \left\| \frac{1}{N}\sum_{j=1}^{N} x_j^{h,\tau_j^h} - (x_i^{h,\tau_i^h} + w_{i,j}^h \sum_{j=1}^{N} a_{i,j}^h (x_j^{h,\tau_j^h} - x_i^{h,\tau_i^h})) \right\|_2 \\ &= \left\| \sum_{j=1}^{N} (\frac{x_j^{h,\tau_j^h} - x_i^{h,\tau_i^h}}{N} - w_{i,j}^h a_{i,j}^h (x_j^{h,\tau_j^h} - x_i^{h,\tau_i^h})) \right\|_2 . \end{aligned} \tag{23}$$

According to $w_{i,j}^h = \frac{1}{u_{max}^h + 1}$ in Eq. (6), we set $u_{max}^h = N-1$ for simplicity, which is the possible maximum value [34]. Thus, it follows:

$$\begin{aligned} \mathbb{E}D_i^{h+1} &= \left\| \sum_{j=1}^{N} \frac{(1-a_{i,j}^h)(x_j^{h,\tau_j^h} - x_i^{h,\tau_i^h})}{N} \right\|_2 \\ &\le \frac{1}{N}\sum_{j=1}^{N}(1-a_{i,j}^h)D_{i,j}^h, \end{aligned} \tag{24}$$

where $D_{i,j}^h = \|x_i^{h,\tau_i^h} - x_j^{h,\tau_j^h}\|_2$ ($\forall i,j \in [N]$) is the consensus distance between two models of worker i and worker j. The last step of Eq. (24) follows the triangle inequality. After receiving local models of neighbors, worker i can locally calculate the consensus distance $D_{i,j}^h$, $\forall j \in \mathcal{N}_i^h$. As a result, the upper bound of the average consensus distance in Eq. (9) can be expressed as:

$$\mathbb{E}D^{h+1} \le \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(1-a_{i,j}^h)D_{i,j}^h. \tag{25}$$

Note that when we set $a_{i,j}^h = 1$, $\forall i,j \in [N]$, the upper bound of average consensus distance D^{h+1} is 0, *i.e.*, if each worker receives local models from all others, the updated models among workers are identical.

To solve the problem in Eq. (12) with Eq. (25), we still need to know the consensus distances among models of all workers. However, if worker i and worker j are not connected at round h, it is infeasible to obtain their consensus distance directly since each worker only receives local models from its neighbors. Thus, we need to estimate the consensus distance between unconnected workers with the help of those of the connected workers. Firstly, when the coordinator has collected consensus distance $D_{i,p}^h$ and $D_{p,j}^h$, $\forall p \in N \setminus \{i,j\}$, $D_{i,j}^h$ can be estimated as:

$$\begin{aligned} D_{i,j}^h &= \left\| x_i^{h,\tau_i^h} - x_p^{h,\tau_p^h} + x_p^{h,\tau_p^h} - x_j^{h,\tau_j^h} \right\|_2 \\ &\le \left\| x_i^{h,\tau_i^h} - x_p^{h,\tau_p^h} \right\|_2 + \left\| x_p^{h,\tau_p^h} - x_j^{h,\tau_j^h} \right\|_2 \\ &= D_{i,p}^h + D_{p,j}^h, \end{aligned} \tag{26}$$

where the second step follows the triangle inequality. Thus we can estimate $D_{i,j}^h$ as $\hat{D}_{i,j}^h$:

$$\hat{D}_{i,j}^h = \min_{p \in [N]\setminus\{i,j\}} (D_{i,p}^h + D_{p,j}^h). \tag{27}$$

Secondly, if there is no common neighbor between worker i and worker j at round h (*i.e.*, $\mathcal{N}_i^h \cap \mathcal{N}_j^h = \emptyset$), we can use Eq. (26) and Eq. (27) iteratively to obtain $\hat{D}_{i,j}^h$. Since the network topology is a connected graph, the above problem is equivalent to the shortest path problem, which can be solved efficiently by the Floyd-Warshall algorithm [41] at the coordinator. As the triangle inequality may amplify consensus distance among workers, the historical consensus distance is used to make our estimation more stable and accurate. Specifically, we use the exponential moving average to smooth the consensus distance, with $\beta_1 \in [0,1]$, as follows:

$$D_{i,j}^h = (1-\beta_1)D_{i,j}^{h-1} + \beta_1 \hat{D}_{i,j}^h, \text{ if } a_{i,j}^h = 0. \tag{28}$$

B. Algorithm Description

Firstly, to minimize the average waiting time of all workers, we let the t_i^h among workers be approximately equal. Then we can have the following formulation:

$$\lfloor \frac{\tau_l^h \cdot \mu_l^h + \max\{\beta_{l,j}^h\}}{\tau_i^h \cdot \mu_i^h + \max\{\beta_{i,j}^h\}} \rfloor = 1, \tag{29}$$

where l denotes the index of the fastest worker with the largest local updating frequency at round h. Thus, $\tau = \tau_l^h$. Then the total training time can be formulated as follows:

$$T(H,\tau) = \sum_{h=1}^{H}(\tau \cdot \mu_l^h + \max\{\beta_{l,j}^h\}). \tag{30}$$

Secondly, the problem in Eq. (12) is a non-linear mixed integer programming problem, which is hard to solve [42], [43]. However, given a specific network topology, we can take

Algorithm 1 Procedure at worker i

```
1: for $h = 1$ to $H$ do
2:     Receive $\mathcal{N}_i^h$ and $\tau_i^h$ from the coordinator;
3:     Perform local updating of $\tau_i^h$ times by Eq. (3);
4:     Estimate $L_i \leftarrow \frac{\|\nabla f_i(x_i^{h+1}) - \nabla f_i(x_i^h)\|}{\|x_i^{h+1} - x_i^h\|}$;
5:     Estimate $\sigma_i \leftarrow \mathbb{E}\left[\|\nabla F_i(x_i^h, \xi_i^h) - \nabla f_i(x_i^h)\|^2\right]$;
6:     Send local model to workers in $\mathcal{N}_i^h$;
7:     Receive models from workers in $\mathcal{N}_i^h$;
8:     Aggregate models by Eq. (5) and obtain $x_i^{h+1}$;
9:     Record computing time $\mu_i^h$ and communication time $\beta_{i,j}^h$, $\forall j \in \mathcal{N}_i^h$;
10:    Compute consensus distance $D_{i,j}^h$, $\forall j \in \mathcal{N}_i^h$;
11:    Send $\mu_i^h$, $\beta_{i,j}^h$, $D_{i,j}^h$, $L_i$, $\sigma_i$ to the coordinator;
Output: $x_i^H$.
```

Algorithm 2 Procedure at coordinator

```
1: for $h = 1$ to $H$ do
2:     Send $\mathcal{N}_i^h$ and $\tau_i^h$ to worker $i$, $\forall i \in [N]$;
3:     Receive $\mu_i^h$, $\beta_{i,j}^h$, $D_{i,j}^h$, $L_i$, $\sigma_i$ from worker $i$, $\forall i \in [N]$;
4:     $L \leftarrow \frac{1}{N}\sum_i^N L_i$;
5:     $\sigma \leftarrow \frac{1}{N}\sum_i^N \sigma_i$;
6:     Determine the local updating frequency and network topology for each worker by the proposed algorithm in Alg. 3;
```

Algorithm 3 Adaptive control algorithm of FedHP

```
Input: $\mu_i^h$, $D_{i,j}^h$, $\beta_{i,j}^h$, $\forall i,j \in [N]$; $L$, $\sigma$; $D_{max}^h$; $\mathbf{A}_b$.
1: Initialize adjacent matrix $\mathbf{A}^h = \mathbf{A}_b$, search step $s = N$ and $Flag = True$;
2: Minimize $T_i(H, \tau_i^h) = \sum_{h=1}^H (\tau_i^h \cdot \mu_i^h + \max\{\beta_{i,j}^h\})$ and obtain $T_i$ and $\tau_i^h$ of worker $i$, $\forall i \in [N]$;
3: $l \leftarrow \arg\min_i(T_i)$, $T \leftarrow T_l$ and $\tau \leftarrow \tau_l$;
4: while $True$ do
5:     if $Flag$ then
6:         $s = \lfloor\sqrt{\sum_{i,j} a_{i,j}^h}\rfloor$;
7:     else
8:         $s = \lfloor s/2 \rfloor$;
9:     Select $s$ slowest links under the threshold of Eq. (31) into $E$;
10:    Initialize $\mathbf{A}' \leftarrow \mathbf{A}^h$;
11:    for each link $e_{i,j} \in E$ do
12:        Set $a_{i,j} \in \mathbf{A}'$ as 0;
13:        if $\mathbf{A}'$ is not connected then
14:            Set $a_{i,j} \in \mathbf{A}'$ as 1;
15:    Minimize $T_i(H, \tau_i^h) = \sum_{h=1}^H (\tau_i^h \cdot \mu_i^h + \max\{\beta_{i,j}^h\})$ and obtain $T_i$ and $\tau_i^h$ of worker $i$, $\forall i \in [N]$;
16:    $l' \leftarrow \arg\min_i(T_i)$, $T' \leftarrow T_{l'}$ and $\tau \leftarrow \tau_{l'}$;
17:    if $T' < T$ then
18:        $l$, $T$, $\tau$, $\mathbf{A}^h$, $Flag \leftarrow l'$,$T'$, $\tau_{l'}$, $\mathbf{A}'$ ,$True$;
19:    else
20:        $Flag \leftarrow False$;
21:    if not $Flag$ and $s == 1$ then
22:        Break;
23: Calculate $\tau_i^h$ for each worker by Eq. (29), where $\tau_l^h = \tau$;
Output: $\tau_i^h$, $\forall i \in [N]$, $\mathbf{A}^h$.
```

the upper bound of D^{h+1} in Eq. (25) as the estimation and transform Eq. (12) into a linear programming problem as:

$$\min T(H, \tau)$$
$$s.t. \begin{cases} \frac{1}{N^2}\sum_{i=1}^N \sum_{j=1}^N (1 - a_{i,j}^h) D_{i,j}^h \le D_{max}^h \\ \lfloor \frac{\tau_l^h \cdot \mu_l^h + \max\{\beta_{l,j}^h\}}{\tau_i^h \cdot \mu_i^h + \max\{\beta_{i,j}^h\}} \rfloor = 1 \end{cases} \tag{31}$$

In terms of Eq. (31), we propose an efficient algorithm, that adaptively determines local updating frequency for each worker and constructs the network topology. And the coordinator is responsible for monitoring the network condition and recording the model training status.

We present the procedure for workers (Alg. 1) and the coordinator (Alg. 2) while the proposed algorithm is formally described in Alg. 3. In Alg. 1, at the beginning of round h, each worker i requests the information about its neighbor set $\mathcal{N}_i^h$ and local updating frequency τ_i^h from the coordinator. Then worker i performs local updating of τ_i^h times by Eq. (3) and estimates the parameters L_i and σ_i. After local updating is finished, worker i sends the local model to its neighbors and waits for receiving the models from its neighbors for aggregation. The local updating frequency of each worker is associated with its computing and communicating capabilities. For instance, the workers with high performance are assigned with larger local updating frequencies, so that each worker does not need to waste too much waiting time. After receiving models from the neighbors, worker i computes consensus distance $D_{i,j}^h$, $\forall j \in \mathcal{N}_i^h$. Finally, worker i sends network conditions, model training statuses, and other parameters to the coordinator and starts the next communication round.

In Alg. 2, the coordinator waits for receiving the parameters (*i.e.*, L_i and σ_i), consensus distance (*i.e.*, $D_{i,j}^h$), computing time (*i.e.*, μ_i^h) and communication time (*i.e.*, $\beta_{i,j}^h$) from workers, and takes average of parameters L_i and σ_i to get L and σ. Then the coordinator calls Alg. 3 to get local updating frequencies and network topology of different workers for the next communication round.

As indicated in Eq. (30), the completion time of model training depends on the slowest link and the slowest worker. Thus we mainly use the greedy algorithm to remove the slow links in the current network topology to reduce the completion time under the threshold of consensus distance in Eq. (31). The procedure executes iteratively until the completion time cannot be reduced after removing any slow links. Specifically, we take the network conditions, model training statuses of workers, and other parameters as the algorithm input. Firstly, we start from the base topology (*i.e.*, $\mathbf{A}_b$) which includes all available links for P2P communication. Then we set $\tau_i^h = \sqrt{\frac{Nf(\overline{x}^1)}{LH\eta^2\sigma^2}}$ and minimize $T_i(H, \tau_i^h)$ by using an LP solver to obtain T_i and τ_i^h for worker i, $\forall i \in [N]$. We obtain the minimum of completion time T_l in the base topology and get the local updating frequency τ_l^h of worker l at round h (Line 1-3), where

$l = \arg\min_i(T_i)$. In order to search the optimal topology and local updating frequencies efficiently, we first take a large search step. Concretely, we set the search step s as the square root of the number of links in the current topology (Line 5-6). At round h, since the slow links may become the system bottleneck in terms of time, we use a greedy algorithm to remove s slowest links and obtain the new network topology A' (Line 10-14). Then we minimize $T_i(H, \tau_i^h)$ again to obtain the new minimum of completion time $T_{l'}$ in the new topology and get the new local updating frequency $\tau_{l'}$ (Line 15-16). If a better solution (*i.e.*, shorter completion time) is found, the current network topology and local updating frequency are updated (Line 17-18). If we cannot find a better solution at the current search step, the search step is reduced by half. If the completion time T cannot be further reduced by removing any link, we stop searching and obtain the final network topology as well as local updating frequency of worker l. It is worth noting that we only remove the links that will not affect the connectivity of the network topology and exceed the constraint of consensus distance D_{max}^h in Eq. (31). In our algorithm, we follow [35] to set the threshold of D_{max}^h adaptively. Specifically, D_{max}^h is the exponential moving average of the gradient norm:

$$D_{max}^h = (1-\beta_2)D_{max}^{h-1} + \frac{\beta_2}{N}\sum_{i=1}^{N}\left\|g_i^h\right\|_2, \tag{32}$$

where $\frac{1}{N}\sum_{i=1}^{N}\left\|g_i^h\right\|_2$ denotes the average norm of local updates at round h among all workers and $\beta_2 \in [0,1]$.

Herein, we analyze the time complexity of Alg. 3. As described above, the proposed algorithm reduces the search step s by half if a better solution cannot be found at the current search step. As a result, there are at most $\lceil \log N \rceil$ iterations, where N is the number of workers. In each iteration, the linear programming can be solved in polynomial time according to [44]. Actually, since the base topology in real world is usually sparse, the practical time cost for Alg. 3 will be further reduced at the coordinator, which is usually deployed in cloud or cloudlet with high computing power. Therefore, the time for solving the joint optimization problem can be negligible, compared with that for model training and transmission.

V. Experimentation and Evaluation

A. Datasets and Models

Datasets: We conduct extensive experiments on three real-world datasets: (i) EMNIST, (ii) CIFAR-10, and (iii) ImageNet. Specifically, EMNIST [45] is a handwritten character dataset that contains 731,668 training samples and 82,587 test samples from 62 categories (10 digits, 52 characters with lowercase and uppercase). CIFAR-10 is an image dataset composed of 60,000 32×32 colour images (50,000 for training and 10,000 for test) in 10 categories. ImageNet [46] is a dataset for visual recognition which consists of 1,281,167 training images, 50,000 validation images and 100,000 test images from 1,000 categories. To cope with the constrained resource of edge devices, we create IMAGE-100, a subset of ImageNet that contains 100 out of 1,000 categories, and each sample is resized with the shape of 64×64×3.

To simulate the non-IID setting, we propose to create synthesized non-IID datasets with different *class distribution skews* as in [2], [19], *e.g.*, a single user can possess more data for one class or a couple of classes than others. Concretely, p (*e.g.*, 0.1, 0.2, 0.4, 0.6 and 0.8) of a unique class is divided equally for every three workers and the remaining samples of each class are partitioned to other workers uniformly. Accordingly, the non-IID levels of the above datasets are denoted as 0.1, 0.2, 0.4, 0.6 and 0.8, respectively. Note that p = 0.1 is a special case, where the distribution of training dataset is IID for 30 workers. For fair comparisons, the full test datasets are used across all workers.

Models: Three models with different types and structures are implemented on the above three real-world datasets for performance evaluation: (i) CNN on EMNIST, (ii) AlexNet on CIFAR-10, (iii) VGG-16 on IMAGE-100. Firstly, The plain CNN model [7] specialized for the EMNIST dataset has two 5×5 convolutional layers, a fully-connected layer with 512 units, and a softmax output layer with 62 units. Secondly, An 8-layer AlexNet [47], which is composed of three 3×3 convolutional layers, one 7×7 convolutional layer, one 11×11 convolutional layer, two fully-connected hidden layers, and one fully-connected output layer, is adopted for CIFAR-10. Thirdly, a famous model VGG-16 [48], that consists of 13 convolution layers with kernel of 3×3, two dense layers and a softmax output layer, is utilized to classify the images in IMAGE-100.

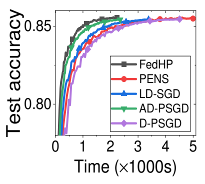
B. Baselines and Metrics

Baselines: We choose four classical algorithms as baselines for performance comparison, which are summarized as follows. (i) D-PSGD [12] is a synchronous DFL algorithm using a ring network topology and the same local updating frequency for workers. (ii) AD-PSGD [23] is an asynchronous DFL algorithm, where workers randomly send local models to one of their neighbors immediately after performing local updating to speed up the training process. (iii) LD-SGD [21] alternates the frequencies of local updating and global updating for efficient decentralized communication. (iv) PENS [22] with adaptive network topology allows workers with similar data distributions to communicate with each other to deal with statistical heterogeneity.

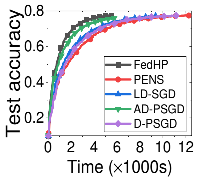
Metrics: The following metrics are adopted to evaluate the performance of FedHP and the baselines. (i) *Test accuracy* is measured by the proportion between the amount of the right data predicted by the model and that of all data. Specifically, at each communication round, we evaluate the average test accuracy of all workers' models trained with different algorithms on the test datasets. (ii) *Completion time* is defined as the total training time until the average model of all workers converges to the target accuracy. Concretely, we record the completion time of each communication round and sum up to get the total training time. (iii) *Average waiting time* is introduced to reflect the training efficiency of different algorithms. Specifically, the waiting time of worker i at round h can be represented by $t^h - t_i^h$, then the average waiting time of all workers at round h is expressed as $\frac{1}{N}\sum_{i=1}^{N}(t^h - t_i^h)$.

C. Experiments

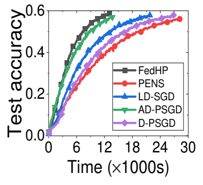
1) Experimental Setup: We evaluate the performance of FedHP through extensive simulation experiments, which are



(a) EMNIST

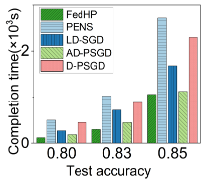


(b) CIFAR-10

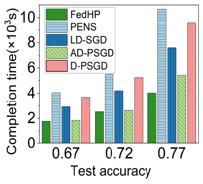


(c) IMAGE-100

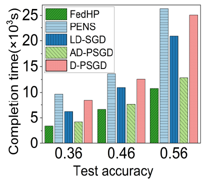
Fig. 2: Test accuracy of five algorithms on the three IID datasets.



(a) EMNIST

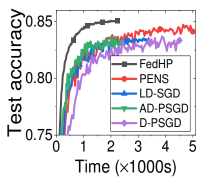


(b) CIFAR-10

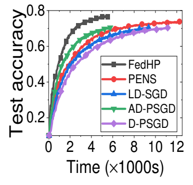


(c) IMAGE-100

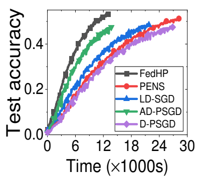
Fig. 3: Completion time of five algorithms when achieving different target accuracy



(a) EMNIST

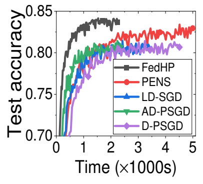


(b) CIFAR-10

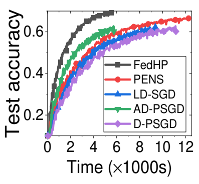


(c) IMAGE-100

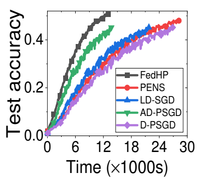
Fig. 4: Test accuracy of five algorithms on the three datasets with non-IID level p=0.6.



(a) EMNIST



(b) CIFAR-10



(c) IMAGE-100

Fig. 5: Test accuracy of five algorithms on the three datasets with non-IID level p=0.8.

conducted on an AMAX deep learning workstation equipped with an Intel(R) Xeon(R) Gold 5218R CPU, 8 NVIDIA GeForce RTX 3090 GPUs and 256 GB RAM. On the workstation, we simulate a heterogeneous EC system with 30 workers and one coordinator (each is implemented as a process in the system) for DFL. The implementation for model training on each worker is based on the PyTorch framework [49], and we use the socket library of Python to build up the communication among workers and between workers and the coordinator.

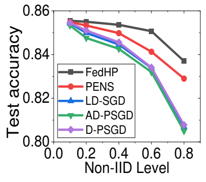
We consider the common situation where each worker communicates with its neighbors and coordinator through either LANs or WANs. To reflect the heterogeneity and dynamics of networks in our simulations, we let the bandwidth of each worker fluctuate between 1Mb/s and 10Mb/s. In addition, for simulating the computing heterogeneity, we assume that the computing time of one local iteration on a certain simulated worker is subject to the Gaussian distribution. Different simulated workers are randomly assigned with a specific Gaussian function whose mean and variance are derived from the time records of performing one local iteration on a commercial device (*e.g.*, laptop, Jetson TX, Xavier NX).

Each experiment will by default run 200, 500, and 500 communication rounds for EMNIST, CIFAR-10 and IMAGE-100, respectively, which will guarantee the convergence of the models. For CNN on EMNIST, the learning rate is initialized as 0.1 and the corresponding decay rate is specified as 0.98, while for AlexNet on CIFAR-10 and VGG-16 on IMAGE-100, the learning rates and the corresponding decay rates of them are identical, separately initialized as 0.1 and 0.993 [18]. Besides, the batch size is set as 32 for all three models.

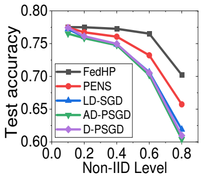
2) Overall Effectiveness: Firstly, we implement a set of experiments of these algorithms on the IID datasets. The training processes of FedHP and the baselines are presented in Fig. 2. In addition, we show the completion time of different algorithms when they achieve different target accuracy in Fig. 3. The results demonstrate that all the algorithms achieve the similar test accuracy eventually. FedHP achieves the fastest convergence, followed by AD-PSGD on all the three datasets, and they are much faster than the other methods. For example, by Figs. 2(a) and 3(a), FedHP takes 1,064s to achieve 85% accuracy for CNN on EMNIST, while PENS, LD-SGD, AD-PSGD, D-PSGD, takes 2,725s, 1,680s, 1,129s, 2,254s, respectively. Besides, by Figs. 2(b) and 3(b), FedHP reduces the completion time of training AlexNet by about 56%, 41%, 3% and 51%, compared with PENS, LD-SGD, AD-PSGD and D-PSGD. Moreover, for VGG-16 on IMAGE-100 as shown in Figs. 2(c) and 3(c), FedHP can separately speed up training by about 2.17×, 1.65×, 1.06× and 2.07×, compared with PENS, LD-SGD, AD-PSGD and D-PSGD. These results demonstrate the advantage of FedHP in accelerating model training.

Secondly, we implement two sets of experiments of these algorithms on non-IID datasets. The results of non-IID scenarios with p=0.6 and p=0.8 are presented in Fig. 4 and Fig. 5, respectively. We observe that FedHP can achieve the same convergence rate as that in the IID scenario while achieving higher accuracy than the other methods. For example, by Fig. 4(b), FedHP takes 5,015s to achieve 76.77% accuracy for AlexNet on CIFAR-10, while PENS, LD-SGD, AD-PSGD and D-PSGD takes 11,953s, 8,926s, 5,539s and 10,634s to achieve 73.52%, 70.54%, 69.29% and 70.35% accuracy, respectively. By Fig. 5(b), FedHP can improve the test accuracy by about 4.83%, 13.37%, 14.26% and 13.52% on CIFAR-10 with non-IID level of p=0.8, compared with PENS, LD-SGD, AD-PSGD and D-PSGD. The above results indicate the effectiveness of FedHP by adaptively assigning appropriate local updating frequencies and constructing network topology for heterogeneous workers.

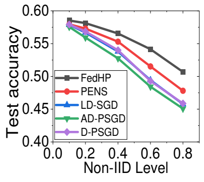
3) Effect of Statistical Heterogeneity: To demonstrate the robustness of FedHP to non-IID data, we show the test accuracies of these algorithms at different non-IID levels in Fig. 6, where the horizontal axis denotes the non-IID level of the datasets. By Fig.6, we observe that the test accuracies of models trained by the five algorithms on all datasets decrease with the increasing of non-IID level. However, FedHP can always achieve the highest model accuracy in comparison with the other algorithms. In addition, PENS with performance-based neighbor selection can achieve higher model accuracy than the algorithms without considering the challenge of statistical heterogeneity. For instance, by Fig. 6(c), FedHP and PENS achieve 50.63% and 47.81% accuracy on IMAGE-100 with



(a) EMNIST

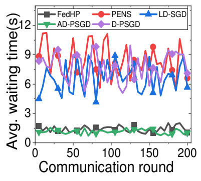


(b) CIFAR-10

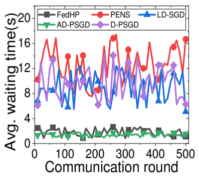


(c) IMAGE-100

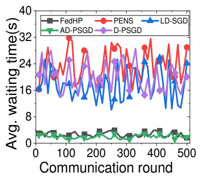
Fig. 6: Test accuracy varies with different non-IID levels.



(a) EMNIST



(b) CIFAR-10



(c) IMAGE-100

Fig. 7: Average waiting time of five algorithms on the three datasets.

non-IID level of p=0.8, while LD-SGD, AD-PSGD and D-PSGD achieve 45.69%, 45.12% and 45.83%, respectively. In AD-PSGD, each worker probably receives the stale models for aggregation, which amplifies the negative impact of non-IID data on model performance, leading to the lowest test accuracy. Both D-PSGD and LD-SGD adopt static network topologies without considering the challenge of statistical heterogeneity on model training, thus they suffer from severe loss of accuracy. Although PENS allows workers with similar data distributions to communicate with each other in order to deal with the statistical heterogeneity, it still achieves a lower test accuracy than FedHP. More specifically, by Fig. 6(c), FedHP can achieve improvement of test accuracy by about 5.90%, 10.81%, 12.22%, 10.47% for VGG-16 on IMAGE-100 with non-IID level of p=0.8, compared with the baselines (*i.e.*, AD-PSGD, LD-SGD, D-PSGD, PENS). Collectively, these results demonstrate the advantage of FedHP in addressing the challenge of statistical heterogeneity.

4) Effect of System Heterogeneity: To further illustrate the efficiency of FedHP, the average waiting time of five algorithms on the three datasets is illustrated in Fig. 7, where we find that FedHP takes much less waiting time than both D-PSGD and PENS. For instance, by Fig. 7(b), the average waiting time of FedHP is 1.7s while PENS and D-PSGD incur average waiting time of 12.1s and 10.6s, respectively. That is because both D-PSGD and PENS assign identical local updating frequencies for workers without considering system heterogeneity, resulting in non-negligible waiting time. In addition, PENS always suffers from more computing time for neighbor selection and model training, incurring the highest average waiting time among five algorithms. As shown in Fig. 7, the average waiting time of AD-PSGD is the lowest among these algorithms, because in the asynchronous scenario, workers update their local models as soon as they receive any models from their neighbors. Besides, LD-SGD, implemented to alternate the frequencies of local updating and global updating, reduces the variance of waiting time to some extent. Concretely, by Fig. 7(c), FedHP and AD-PSGD only incur average waiting time of 3.2s and 2.9s, while LD-SGD, D-PSGD and PENS incur average waiting time of 19.2s, 21.5 and 24.7s, respectively. The above results explain why FedHP and AD-PSGD can achieve much faster converge rate than D-PSGD and PENS while LD-SGD takes less completion time than D-PSGD in Figs. 2, 4 and 5.

VI. Related Work

The concept of FL was first introduced in [7], which has demonstrated the effectiveness of performing distributed model training over distributed and isolated datasets. In order to reduce the communication resource consumption, the early works explored to optimize the local updating frequency [11], [18], [21]. As the local updating frequency increases, the frequency for global aggregation can relatively get decreased, therefore, the communication resource for model transmission can be saved to a great extent. However, these related researches mainly focus on PS-based FL [11], [18], which suffers from the single point of failure problem [12], [13]. Herein, we focus on the more attractive DFL, where Li *et al.* [21] proposed LD-SGD to alternate the frequencies of local updating and global updating to deal with the resource-constrained issue, but they could not address the challenge of system heterogeneity.

As for network topology construction in DFL, there have been many related studies [20], [22], [26]–[28]. Wang *et al.* [27] proposed MATCHA, which uses matching decomposition sampling of the base topology to parallelize inter-worker information exchange so as to significantly reduce communication delay. Besides, Xu *et al.* [28] dynamically constructed an efficient P2P topology to address the challenge of resource limitation and network dynamics. However, the above works all suffered from a drop in model accuracy without considering the negative effect of statistical heterogeneity. In order to overcome statistical heterogeneity, Wang *et al.* [20] proposed CoCo to preferentially select neighbors with large differences in data distribution, while Onoszko *et al.* [22] proposed PENS, where workers with similar data distributions communicate with each other. However, CoCo and PENS did not overcome the challenge of system heterogeneity, often resulting in idle time for staying and waiting for the stragglers before model aggregation. On the contrary, FedHP investigates the benefits of controlling local updating frequency and network topology, which are jointly optimized to adequately address the issues of system and statistical heterogeneities.

VII. Conclusion

This work focuses on system heterogeneity and statistical heterogeneity for DFL. To overcome these challenges, we have proposed FedHP to achieve fast convergence by jointly optimizing both the local updating frequency and network topology in DFL. We have analyzed the convergence rate of FedHP and proposed an efficient algorithm. We have evaluated the performance of FedHP through extensive simulations and the results have demonstrated the efficiency of FedHP.

VIII. Acknowledgement

The corresponding author of this paper is Yang Xu. This research was supported in part by the National Key Research and Development Program of China (Grant No. 2021YFB3301501); in part by the National Science Foundation of China (NSFC) under Grants 62102391, 61936015 and 62132019; in part by the Jiangsu Province Science Foundation for Youths (Grant No. BK20210122). C. Qian was partially supported by NSF grants 1750704 and 2114113.

References

[1] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[2] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[5] J. Liu, H. Xu, L. Wang, Y. Xu, C. Qian, J. Huang, and H. Huang, "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[6] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[8] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[10] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

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

[12] 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," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5693–5700.

[14] Y. Hua, K. Miller, A. L. Bertozzi, C. Qian, and B. Wang, "Efficient and reliable overlay networks for decentralized federated learning," *SIAM Journal on Applied Mathematics*, 2022.

[15] J. Zhang, H. Tu, Y. Ren, J. Wan, L. Zhou, M. Li, and J. Wang, "An adaptive synchronous parallel strategy for distributed machine learning," *IEEE Access*, vol. 6, pp. 19 222–19 230, 2018.

[16] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, 2021.

[17] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[18] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Transactions on Mobile Computing*, 2022.

[19] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[20] L. Wang, Y. Xu, H. Xu, M. Chen, and L. Huang, "Accelerating decentralized federated learning in heterogeneous edge computing," *IEEE Transactions on Mobile Computing*, 2022.

[21] X. Li, W. Yang, S. Wang, and Z. Zhang, "Communication-efficient local decentralized sgd methods," *arXiv preprint arXiv:1910.09126*, 2019.

[22] N. Onoszko, G. Karlsson, O. Mogren, and E. L. Zec, "Decentralized federated learning of deep neural networks on non-iid data," *arXiv preprint arXiv:2107.08517*, 2021.

[23] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3043–3052.

[24] M. S. Assran and M. G. Rabbat, "Asynchronous gradient push," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 168–183, 2020.

[25] Q. Luo, J. He, Y. Zhuo, and X. Qian, "Prague: High-performance heterogeneity-aware asynchronous decentralized training," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 401–416.

[26] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, "Communication-efficient decentralized machine learning over heterogeneous networks," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 384–395.

[27] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," in *2019 Sixth Indian Control Conference (ICC)*. IEEE, 2019, pp. 299–300.

[28] H. Xu, M. Chen, Z. Meng, Y. Xu, L. Wang, and C. Qiao, "Decentralized machine learning through experience-driven method in edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 515–531, 2021.

[29] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and Y. J. Guo, "Multi-timescale decentralized online orchestration of software-defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2716–2730, 2018.

[30] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2393–2406, 2019.

[31] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[32] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *International Conference on Learning Representations*, 2019.

[33] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

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

[35] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. U. Stich, "Consensus control for decentralized deep learning," in *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[36] J. Qian, X. Fafoutis, and L. K. Hansen, "Towards federated learning: Robustness analytics to data heterogeneity," *arXiv preprint arXiv:2002.05038*, 2020.

[37] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 5381–5393.

[38] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7652–7662, 2018.

[39] H. Tang, X. Lian, S. Qiu, L. Yuan, C. Zhang, T. Zhang, and J. Liu, "Deepsqueeze: Decentralization meets error-compensated compression," *arXiv preprint arXiv:1907.07346*, 2019.

[40] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," *arXiv preprint arXiv:2212.02136*, 2022.

[41] P. E. Black, "Dictionary of algorithms and data structures," 1998.

[42] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[43] C. H. Papadimitriou and M. Yannakakis, "The complexity of facets (and some facets of complexity)," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, 1982, pp. 255–260.

[44] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 385–463, 2004.

[45] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.