Heterogeneity-Aware Resource Allocation and Topology Design for Hierarchical Federated Edge Learning

Zhidong Gao, *Student Member, IEEE,* Zhenxiao Zhang, *Student Member, IEEE,*
Yu Zhang, *Student Member, IEEE,* Yanmin Gong, *Senior Member, IEEE,* Yuanxiong Guo, *Senior Member, IEEE*

***Abstract*—Federated Learning (FL) provides a privacy-preserving framework for training machine learning models on mobile edge devices. Traditional FL algorithms, e.g., FedAvg, impose a heavy communication workload on these devices. To mitigate this issue, Hierarchical Federated Edge Learning (HFEL) has been proposed, leveraging edge servers as intermediaries for model aggregation. Despite its effectiveness, HFEL encounters challenges such as slow convergence rates and high resource consumption, particularly in the presence of system and data heterogeneity. However, existing works are mainly focused on improving the efficiency of traditional FL training, leaving the efficiency of HFEL largely unexplored. In this paper, we consider a two-tier HFEL system, where edge devices are connected to edge servers, and edge servers are interconnected through peer-to-peer (P2P) edge backhauls. Our goal is to enhance the training efficiency of the HFEL system through strategic resource allocation and topology design. Specifically, we formulate an optimization problem to minimize total training latency by allocating computation and communication resources, as well as adjusting P2P connections. To ensure convergence under dynamic topologies, we analyze the convergence error bound and introduce a model consensus constraint into the optimization problem. The proposed problem is then decomposed into several subproblems, enabling us to alternatively solve it online. Our method facilitates the efficient implementation of large-scale FL in edge networks under data and system heterogeneity. Comprehensive experiment evaluation on benchmark datasets validates the effectiveness of the proposed method, demonstrating significant reductions in training latency while maintaining the model accuracy compared to various baselines.**

***Index Terms*—Federated learning, resource allocation, topology design, mobile edge**

I. Introduction

THE widespread adoption of edge devices, such as smartphones and Internet-of-things (IoT) devices, each possessing advanced sensing, computing, and storage capabilities, results in an enormous amount of data being produced daily at the network edge. Concurrently, the rapid advancements in artificial intelligence (AI) and machine learning (ML) facilitate the extraction of valuable knowledge from extensive data. The integration of 5G networks with AI/ML technologies is driving the development of numerous innovative applications that have significant economic and social impacts, such as autonomous driving [1], augmented reality [2], real-time video analytics [3], mobile healthcare [4], and smart manufacturing [5]. A key characteristic of these new applications is the substantial and continuously streaming data they produce, which requires efficient processing for real-time learning and decision-making. However, despite these advancements, data sharing is impeded by privacy regulations, such as the General Data Protection Regulation (GDPR), and hardware constraints, like limited communication bandwidth. Federated learning (FL) [6] emerges as a promising solution to these challenges by enabling model training directly on mobile devices in a decentralized manner. FL not only enhances privacy protection but also leverages the computational resources of edge devices.

In a standard FL architecture, the system comprises a cloud-based Parameter Server (PS) and multiple clients[1]. The PS orchestrates the training procedure while the clients carry out model training. The training process consists of multiple communication rounds between the clients and the PS. In each round, each selected client downloads the most recent global model from PS and updates the model using its local data. Then, these updated local models are uploaded to the PS, where they are aggregated into a new global model. However, the process of transferring models between the numerous clients and the PS imposes a considerable data traffic load, leading to increased training latency and network congestion.

To unlock the full potential of FL on mobile edge networks, recent works [7]–[12] have explored Hierarchical Federated Edge Learning (HFEL) by leveraging multi-server collaboration for model training. These works typically adopt a hierarchical architecture that integrates the advantage of cloud-based and decentralized federated learning (DFL) [13]–[16] to improve the speed and reliability of the FL system. In these works, clients are connected to a proximal edge server via wireless networks, while edge servers either connect to a central cloud server via an edge-to-cloud (E2C) network or form a peer-to-peer (P2P) network without connecting to a central cloud server. Each edge server acts as a local PS,

Z. Gao, Z. Zhang, Y. Zhang, and Y. Gong are with the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX, 78249. E-mail: {zhidong.gao@my., yu.zhang@my., yanmin.gong@}utsa.edu

Y. Guo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX, 78249. E-mail: yuanxiong.guo@utsa.edu

[1]Note that we use clients and mobile devices interchangeably in this paper.

orchestrating the training process with its connected clients. To facilitate collaborative training over broader distances and ensure a unified model across the network, edge servers engage in periodic synchronization, sharing, and updating their models with each other via E2C or P2P networks.

However, implementing FL efficiently in realistic edge systems presents significant challenges due to two key factors. 1) System heterogeneity: Edge devices possess limited resources such as battery life, communication bandwidth, and CPU frequency. In addition, these resources are not uniformly distributed among different devices. In traditional HFEL, the faster devices must remain idle, waiting for the slower ones to finish training in each communication round. 2) Statistical heterogeneity: The data that local devices collect are inherently influenced by their specific geographic locations or operational contexts, leading to a non-IID data distribution. This discrepancy can significantly hinder the model convergence speed and accuracy.

Recent research [17]–[20] has studied optimization-based strategies to deal with the challenges introduced by system and statistical heterogeneity in FL environments. Typically, these approaches first construct an optimization problem that encapsulates resource cost, system constraints, and training performance. Then, an adaptive control strategy is developed by solving the optimization problem to optimize device resource allocation and learning task scheduling in real time. Specifically, to guarantee the performance of the learned model under devised control decisions, these studies usually perform the convergence analysis for their learning algorithm. This allows them to derive several key insights which are subsequently applied to the design or solution of their optimization problems. However, these studies primarily focus on cloud-based FL or DFL, and less effect has been made for the FL system with hierarchical architectures.

In this paper, we investigate the efficiency of a two-tier HFEL system, where edge devices are connected to nearby edge servers and edge servers are interconnected via P2P networks. Each edge server with its connected edge devices forms one cluster. Our study focuses on the strategic regulation of CPU processing on edge devices, wireless communication bandwidth, and P2P network configuration to uncover potential efficiency gains. In particular, we formulate an optimization problem that characterizes the inherent resource constraints of the HFEL system and the relationship between resource consumption and training latency. Additionally, we derive a consensus distance constraint that captures the influence of topology design on the model convergence. This enables us to safely remove the unimportant links from the graph to reduce training latency while preserving the model's utility.

To address this problem, we propose FedRT, an online algorithm that decomposes the original optimization into subproblems solved across successive global rounds, with each global round further partitioned into sequential edge rounds. The final edge-round subproblem constitutes a challenging mixed-integer non-convex optimization (combining continuous resource variables and discrete topology decisions). To solve it efficiently, FedRT employs a greedy alternating optimization: iteratively updating the communication graph topology via consensus-aware link pruning, then optimizing resource allocations (CPU, bandwidth), while dynamically adapting these decisions based on real-time system states, environmental conditions, and resource availability. This approach enables FedRT to balance training latency reduction and model convergence in dynamic HFEL environments.

We conducted comprehensive evaluations on three benchmark datasets, CIFAR-10 [21], FEMNIST [22], and FM-NIST [22] under various data distributions and resource configurations to validate the effectiveness of our proposed method. The experimental results illustrate that our method outperforms baselines in terms of the total training latency and convergence speed. In summary, our main contributions are stated as follows:

- We formulated an optimization problem that captures the two trade-offs for the HFEL system: i) model convergence rate versus network topology link density by limiting consensus distance during training. ii) total training latency versus energy cost by finding the optimal communication and computation resource allocation scenario for edge devices.
- We propose FedRT, which jointly optimizes the edge backhaul topology and resource allocation for a two-tier HFEL system, considering both system and data heterogeneity. FedRT achieves high accuracy, low latency, and resource-efficient model training in mobile edge FL.
- We present a detailed analysis of our control strategy and provide several insights on adjusting the control variables to achieve a better trade-off between resource consumption and convergence speed.
- We conduct extensive experiments on benchmark datasets. The results demonstrate the advantage of our method compared with baselines without client scheduling and/or resource allocation.

The remainder of this paper is organized as follows. Section III describes the system model and formulated optimization problem. Sectio II discusses the related works. Section IV presents the convergence result and discusses the insights of topology design. Section VI introduces the proposed solution. Then, in Section VII, we demonstrate the experiment results. Finally, we conclude this paper in Section VIII.

II. Related Work

FL at mobile edge networks faces several challenges, such as high training latency and resource cost due to the presence of system and data heterogeneity. To mitigate these issues, several resource-optimization-based algorithms have been proposed to enhance both training latency and resource utilization of FL. For instance, Luo et al. [23] introduced an adaptive client sampling algorithm aimed at minimizing convergence time in heterogeneous systems. Perazzone et al. [24] proposed a joint client sampling and power allocation scheme to reduce convergence error and average communication time under a

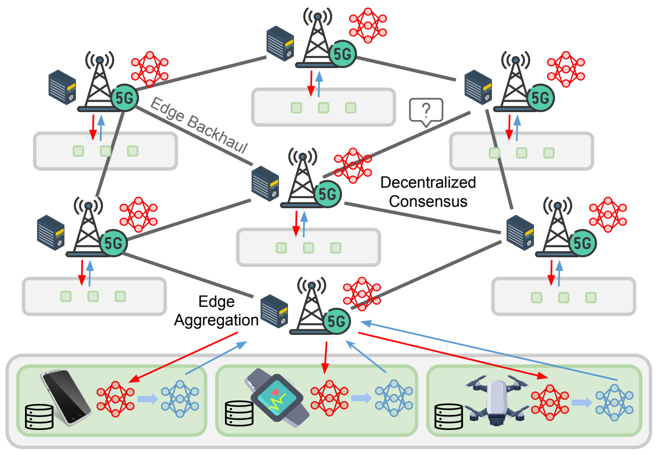


Fig. 1. Flow-chart of a two-tier federated learning system.

power constraint, though their approach does not consider local computation. Wang et al. [25] formulated an optimization problem to minimize convergence bounds by adaptively compressing model updates and determining the probability of local updates. However, these works predominantly focus on cloud-based FL, which relies on a single cloud server to coordinate the model training.

A few studies [26]–[30] have explored resource-optimization of HFEL at mobile networks, where multiple edge servers are responsible for aggregating model updates from a subset of devices. In particular, Zhang et al. [26] proposed a framework for wireless networks, which enhances training efficiency and convergence in multi-cell scenarios by combining intra-cell device-to-device consensus with inter-cell aggregation while optimizing resources to minimize training latency and energy consumption. Feng et al. [29] proposed to optimize computation and communication resources, alongside edge server associations, to minimize global costs while enhancing training performance in HFEL systems. Zhou et al. [30] propose a joint scheduling and resource allocation scheme to improve convergence rates and reduce energy consumption in the mobile edge. The work closest to us is [20], which considers the cost efficiency of a Multi-Cell FL system. In their setting, the client could establish communication links with multiple servers. Their focus is client association and coordinator node selection. Different from these studies, our goal in this paper is to minimize the total learning latency while preserving the model accuracy by jointly optimizing the topology for edge servers, as well as the communication and computation resource allocations for edge devices, under both system and data heterogeneity.

III. Preliminaries

A. Federated Learning over Mobile Edge

We consider a hierarchical federated learning system as shown in Fig. 1. Assume there are C clusters in the system. Every cluster $c \in [C]$ possesses a server colocated with a base station. Each cluster owns a set of devices $\mathcal{S}_c$, and the number of devices is $N_c = |\mathcal{S}_c|$. Note the devices in $\mathcal{S}_c$ only communicate with the server within the same cluster via the wireless communication links, e.g., 5G. We define the set of all devices in the system as $\mathcal{S} = \cup_{c=1}^{C} \mathcal{S}_c$. The total number of devices is $N = |\mathcal{S}|$.

The edge backhaul communication pattern is defined as $\mathcal{G}_b = \{\mathcal{V}, \mathcal{E}_b\}$, which is an undirected and connected graph. Here $\mathcal{V}$ denotes the set of all servers, and $\mathcal{E}_b$ is the set of possible communication links. Moreover, let $\mathbf{A}_b$ be the adjacency matrix of $\mathcal{G}_b$, where $(\mathbf{A}_b)_{c,c'} = 1$ if there is an edge between server c and server c' and $(\mathbf{A}_b)_{c,c'} = 0$ otherwise.

B. Model Training Process

The objective of FL is to find a model $\mathbf{w} \in \mathbb{R}^d$ that minimize the following global objective function:

$$\min_{\mathbf{w}} F(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} F_n(\mathbf{w}), \tag{1}$$

where $F_n(\mathbf{w}) = \mathbb{E}_{x\sim\mathcal{D}_n}[\ell_n(\mathbf{w}; x)]$ is the local objective function of device n, $\mathcal{D}_n$ is the data distribution of device n. Here ℓ_n is the loss function, e.g., cross-entropy, and x denotes a data sample drawn from the distribution $\mathcal{D}_n$. We define the cluster-level objective function as

$$\min_{\mathbf{w}} f_c(\mathbf{w}) = \frac{1}{N_c} \sum_{n\in\mathcal{S}_c} F_n(\mathbf{w}), \tag{2}$$

Here, f_c is the objective function of the c-th cluster, which is the average of the local objective of all devices from cluster c. Then we can rewrite the global objective function (1) as:

$$\min_{\mathbf{w}} F(\mathbf{w}) = \sum_{c=1}^{C} \frac{N_c}{N} f_c(\mathbf{w}). \tag{3}$$

We summarize the detailed training process in **Algorithm 1**. Specifically, we assume there is a central controller that will coordinate the operations of the HFEL system. It collects the device-specific parameters, including the CPU cycle per sample c_n, the size of the training data D_n, energy budget $\bar{\mathcal{E}}_n$, the capacitance coefficient α_n, decision boundaries $f_n^{\min}$, $f_n^{\max}$, $p_n^{\min}$, $p_n^{\max}$, and background noise power N_0, before the training starts. The extra costs (e.g., bandwidth consumption and time cost) for information collection are ignored as in prior works [19], [20], [23], [25], [31]–[33]. Other inputs are hyper-parameters, e.g., sampling frequency K, batch size I, and local iterations S depending on user specification.

At the beginning of t-th global round, each server collects its real-time communication bandwidth to other servers $\mathbf{B}^t$ and sends it to the coordinator (line 2). Each global round includes R edge rounds. At the beginning of r-th edge round, each edge device records its observed signal-to-noise ratio $\text{SNR}_n^{t,r}$ and communication bandwidth to connected edge server $b_{c'}^{t,r}$, and then sends them to the coordinator (line 5). Coordinator determines the allocated bandwidth for each client $b_n^{t,r}$, CPU frequency $f_n^{t,r}$, and graph topology $\mathcal{G}^t$ by solving **Algorithm 2** (line 6). The solving details will be elaborated in Section VI.

Algorithm 1 Training Process of FL over Mobile Edge

```
Input: Initial global model $\mathbf{u}_c^{0,0}$, learning rate $\eta$, mixing matrix $M$, $T, R, S$, $\psi$
Output: Final learned model $\mathbf{u}^{T-1}$
1: for each global round $t = 0, \dots, T-1$ do
2:     for each cluster $c \in [C]$ in parallel do
3:         for each edge round $r = 0, \dots, R-1$ do
4:             Server $c$ broadcasts $\mathbf{u}_c^{t,r}$ to all devices in $\mathcal{S}_c$
5:             for Device $n \in \mathcal{S}_c$ in parallel do
6:                 $\mathbf{w}_n^{t,r,0} \leftarrow \mathbf{u}_c^{t,r}$
7:                 for $s = 0, \dots, S-1$ do
8:                     Compute a stochastic gradient $\mathbf{g}_n$ over a mini-batch $\xi_n$ drawn from $\mathcal{D}_n$
9:                     $\mathbf{w}_n^{t,r,s+1} \leftarrow \mathbf{w}_n^{t,r,s} - \eta \mathbf{g}_n(\mathbf{w}_n^{t,r,s})$
10:                end for
11:                Upload $\mathbf{w}_n^{t,r,S}$ to server $c$
12:            end for
13:            $\mathbf{u}_c^{t,r+1} \leftarrow \frac{1}{N_c} \sum_{n \in \mathcal{S}_c} \mathbf{w}_n^{t,r,S}$
14:        end for
15:        $\mathbf{u}_c^{t+1,0} \leftarrow \mathbf{u}_c^{t,R-1} + \sum_{c' \in \{c\} \cup \mathcal{N}_c^t} \mathbf{M}_{c,c'}^{\psi} (\mathbf{u}_{c'}^{t,R-1} - \mathbf{u}_c^{t,R-1})$
16:    end for
17: end for
```

After that, server c broadcasts the latest global model $\mathbf{u_c^{t,r}}$ to all devices in $\mathcal{S}_c$ (line 7). Then, devices receive $\mathbf{u_c^{t,r}}$ and initialize their local model to be the received global model (line 9). Next, the device runs S iterations local update (SGD) on its local training dataset (lines 10-13). After local training, the device uploads the model to the server c (line 14). Server c aggregates the local models from devices in $\mathcal{S}_c$ (line 16). The edge training lasts for R rounds. After edge training, server c communicates ψ times with its neighbor server $c' \in \mathcal{N}_c$ to synchronize the global model (lines 18-19).

C. Cost Analysis of HFEL

1) Device Communication Time: We assume the communication follows the Frequency Division Multiple Access (FDMA) protocol, and each server adaptively assigns its communication bandwidth to all devices within the cluster. Let $b_n^{t,r}$ be the assigned bandwidth for device n at edge round r and global round t. Then, the communication rate of device n is

$$r_n^{t,r} = b_n^{t,r} \log_2(1 + \text{SNR}_n^{t,r}), \tag{4}$$

Here, $\text{SNR}_n^{t,r}$ denotes the signal-to-noise ratio between device n and the corresponding server at edge round r and global round t.

The totally assigned bandwidth for all devices in $\mathcal{S}_c$ should not exceed the available bandwidth of server c, therefore we have

$$\begin{aligned} &\sum_{n \in \mathcal{S}_c} b_n^{t,r} \le b_c^{t,r}, \forall c, \forall t \\ &b_n^{t,r} > 0, \forall t, \forall r, \forall n, \end{aligned} \tag{5}$$

where $b_c^{t,r}$ denotes the available bandwidth of the server c at edge round r and global round t. The communication time $\mathcal{T}_n^{t,r,\text{com}}$ of device n can be expressed as

$$\mathcal{T}_n^{t,r,\text{com}} = \frac{\Lambda}{r_n^{t,r}} = \frac{\Lambda}{b_n^{t,r} \log_2(1 + \text{SNR}_n^{r,t})}. \tag{6}$$

Here, Λ denotes the model size. Note that we only consider the upload time cost since the download time cost is not the bottleneck for the practical HFEL system.

2) Server Communication Time: Define $\mathbf{B}^t \in \mathbb{R}^{C \times C}$ as the matrix of communication bandwidth between servers at global round t, where $\mathbf{B}_{c,c'}^t$ is the available bandwidth between server c and c'. If there is no available link between c and c', we set $\mathbf{B}_{c,c'}^t = 0, \forall (c,c') \notin \mathcal{E}_b$. It is worth noting that the communication between servers is accomplished through the edge backhaul, which operates independently from the communication between the server and devices (e.g., through the base station).

In this paper, we consider the topology design approach that adaptively selects and deletes the slow communication links from the base graph $\mathcal{G}_b$. Specifically, we aim to find an edge backhaul communication pattern $\mathcal{G}^t = \{V, E^t\}$ at each global round. Here $E^t \subset E_b$ denotes the set of remaining edges in $\mathcal{G}^t$. Note different choices of $\mathcal{G}^t$ have an impact on model convergence. We will discuss it in a later section. Then, the communication time between server c and its neighbors depends on the slowest links, which can be expressed as

$$\mathcal{T}_c^t = \frac{\psi \Lambda}{\min_{c' \in \mathcal{N}_{c,}^t} \{\mathbf{B}_{c,c'}^t\}}. \tag{7}$$

Here $\mathcal{N}_c^t$ is the set of neighbors for server c in $\mathcal{G}^t$.

Let $\mathbf{A}^t$ be the adjacency matrix of $\mathcal{G}^t$, and $\mathbf{D}^t$ be the degree matrix. Here $\mathbf{D}^t$ is a diagonal matrix, and each element denotes the number of neighbors $\mathbf{D}_{c,c}^t = |\mathcal{N}_c^t|$. Then, the Laplacian matrix $\mathbf{L}^t$ of $\mathcal{G}^t$ can be expressed as

$$\mathbf{L}^t = \mathbf{D}^t - \mathbf{A}^t. \tag{8}$$

We always require the graph $\mathcal{G}^t$ to remain connected. According to the spectral graph theory [34], it can be translate into following constraint:

$$\lambda_2(\mathbf{L}^t) > 0, \forall t \tag{9}$$

where $\lambda_l(\mathbf{L}^t)$ denotes the l-th smallest eigenvalue of Laplacian matrix $\mathbf{L}^t$.

3) Edge Device Computation Time: Let μ_n represent the number of CPU cycles required by device n to process a single training example. The value of μ_n can either be measured offline or known as a prior. The total number of CPU cycles required to train one edge round is $SI\mu_n$, where I denotes the batch size. The computation time for one edge round of device n can be formulated as

$$\mathcal{T}_n^{t,r,\text{cmp}} = \frac{SI\mu_n}{f_n^{t,r}}, \tag{10}$$

where $f_n^{t,r}$ is the CPU frequency of device n at edge round r and global round t.

4) Time Model: In **Algorithm 1**, one global round comprises R edge rounds conducted within the cluster, followed by ψ times synchronization between the clusters. The time required for one edge round depends on the slowest device in that round. Therefore, we have

$$\mathcal{T}_c^{t,r} = \max_{n \in \mathcal{S}_c} \{\mathcal{T}_n^{t,r,\text{cmp}} + \mathcal{T}_n^{t,r,\text{com}}\}, \tag{11}$$

where $\mathcal{T}_c^{t,r}$ is the time cost of cluster c at edge round r and global round t.

After R edge rounds training within the cluster, the server starts to communicate with neighbor servers. Similarly, the slowest cluster determines the final completion time for one global round

$$\mathcal{T}^t = \max_{c \in [C]} \{\textstyle\sum_{r=0}^{R-1} \max_{n \in \mathcal{S}_c} \{\mathcal{T}_n^{t,r,\text{cmp}} + \mathcal{T}_n^{t,r,\text{com}}\} + \mathcal{T}_c^t\}. \tag{12}$$

Here, $\mathcal{T}^t$ is time consumption at global round t.

5) Edge Device Computation Energy: Following [35], the CPU energy cost for one edge round training can be formulated as

$$\mathcal{E}_n^{t,r,\text{cmp}} = \frac{\alpha_n}{2} SI\mu_n (f_n^{t,r})^2, \tag{13}$$

where $\alpha_n/2$ denotes the effective capacitance coefficient of the computing chipset on device n.

We assume the devices could adjust their CPU frequency by leveraging the Dynamic Voltage and Frequency Scaling (DVFS) technique. Due to the hardware limitation, the CPU frequency satisfy

$$f_n^{\min} \leq f_n^t \leq f_n^{\max} \tag{14}$$

Here $f_n^{\min}, f_n^{\max}$ denote the minimum and maximum CPU frequency of device n.

6) Edge Device Communication Energy: For devices, the energy used for downloading is usually negligible. Therefore, we only consider the communication energy usage during uploading

$$\mathcal{E}_n^{t,r,\text{com}} = p_n \mathcal{T}_n^{t,r,\text{com}} = \frac{p_n \Lambda}{b_n^{t,r} \log_2(1 + \text{SNR}_n^{r,t})}, \tag{15}$$

where p_n denotes the communication power of device n.

7) Energy Model: As the server is usually equipped with a plug-in power supply, the energy cost of the server is not our focus in this paper. For devices, the energy cost has two sources: the energy used for local training and the energy used for wireless transmission between the devices and the server. Thus, we have

$$\mathcal{E}_n^{t,r} = \frac{p_n \Lambda}{b_n^{t,r} \log_2(1 + \text{SNR}_n^{r,t})} + \frac{\alpha_n}{2} SI\mu_n (f_n^{t,r})^2, \tag{16}$$

where $\mathcal{E}_n^{t,r}$ is the energy consumption of device n at edge round r and global round t.

D. Inter-Cluster Model Consensus

In **Algorithm 1**, there is no actual global model, and each server hosts an edge model that serves as the "global model" within the cluster. The edge models belonging to different clusters are usually not the same. We introduce the consensus distance to measure the discrepancy between any two edge models

$$\Upsilon_{c,c'}^t = \|\mathbf{u}_c^{t,R-1} - \mathbf{u}_{c'}^{t,R-1}\|. \tag{17}$$

Here, $\Upsilon_{c,c'}^t$ denotes the consensus distance between the edge model c and the edge model c' in the global round t.

We define the consensus distance between the edge model and the actual global model

$$\Upsilon_c^t = \|\bar{\mathbf{u}}^{t,0} - \mathbf{u}_c^{t,0}\|, \tag{18}$$

where $\bar{\mathbf{u}}^{t,0} = \frac{1}{C} \sum_{c=1}^{C} \mathbf{u}_c^{t,0}$ denotes the average of all edge models. Note that $\bar{\mathbf{u}}^{t,0}$ is not available in our system. Moreover, the average consensus distance of all edge models is

$$\Upsilon^t = \tfrac{1}{C} \textstyle\sum_{c=1}^{C} \Upsilon_c^t. \tag{19}$$

Similar to the weight divergence and the consensus distance in prior works [36]–[38], the consensus distance depends on the data distribution across the clusters, which is the key factor in capturing the joint effect of decentralization.

IV. Convergence Constraint

A. Convergence Result

For the convergence of HFEL systems, several studies [8], [11], [39] have been carried out. However, these works mainly focus on the convergence analysis of the learning algorithm in HFEL and assume that all edge devices are homogeneous (i.e., their computation, communication, and storage capabilities are the same) and the edge backhaul topology is given prior. To analyze the impact of topology design on the convergence of proposed algorithms, we re-elaborate the convergence results from prior work [8].

Let L be the Lipschitz constant, $\varphi = tRS + rS + s$ denote the global iteration index, and $\Phi = RTS$ represent the total number of iterations. Additionally, let σ be the bound of unbiased stochastic gradient variance, ϵ_c be the intra-cluster divergence, ϵ be the inter-cluster divergence, and ζ be the second largest eigenvalue of the mixing matrix. Furthermore, we define the following constants

$$\Omega_1 = \frac{\zeta^{2\psi}}{1 - \zeta^{2\psi}}, \quad \Omega_2 = \frac{1}{1 - \zeta^{2\psi}} + \frac{2}{1 - \zeta^{\psi}} + \frac{\zeta^{\psi}}{(1 - \zeta^{\psi})^2}, \tag{20}$$

Let $\eta \leq \min\{\frac{1}{2LS}, \frac{1}{2\sqrt{2\Omega_2}LRS}\}$, we have

$$\begin{aligned} \frac{1}{\Phi} \sum_{\varphi=0}^{\Phi-1} \mathbb{E}\|\nabla F(\mathbf{u}^{\varphi})\|^2 &\leq \frac{2(F(\mathbf{u}^1) - F_{\inf})}{\eta\Phi} + \frac{\eta L \sigma^2}{N} \\ &+ 8\eta^2 L^2 (\Omega_1 RS + \frac{C-1}{N} RS)\sigma^2 + 16\eta^2 L^2 R^2 S^2 \Omega_2 \epsilon^2 \\ &+ 8\frac{N-C}{N} \eta^2 L^2 S \sigma^2 + 16 L^2 \eta^2 S^2 \sum_{c=1}^{C} \frac{N_c}{N} \epsilon_c^2. \end{aligned} \tag{21}$$

The fourth term $16\eta^2L^2R^2S^2\Omega_2\epsilon^2$ captures the inter-cluster error bound. Here, Ω_2 captures the influence of varying topologies on the convergence error bound, while ϵ^2 quantifies the effect of inter-cluster non-IID data distribution [8]. A sparser topology results in a larger value of Ω_2, thereby increasing the error in the convergence bound. Similarly, a higher degree of inter-cluster non-IID distribution leads to a larger value of ϵ, amplifying the error caused by inter-cluster data heterogeneity. Therefore, removing certain links from the graph increases overall sparsity, enlarging convergence errors and potentially leading to longer training times and degradation in model performance, especially when data across clusters are highly non-IID.

Note that our goal is to reduce the training latency of the HFEL system. To achieve this, we aim to identify and remove links that contribute minimally to model convergence while imposing high communication overhead between edge servers. Although the above convergence results do not directly quantify the impact of individual links on the error bound, they highlight the necessity of jointly considering model convergence when optimizing the graph topology. To further analyze how individual links impact the error bound, we leverage the concept of consensus distance to assess the influence of different links on model convergence.

Intuitively, links in the graph primarily influence the P2P communication process among edge servers. For instance, in a fully connected graph, all edge servers have identical edge models after P2P communication. The consensus distance between any two edge models is zero. In contrast, if the graph contains no links, the edge servers become isolated, each maintaining an independent model. This leads to a non-zero consensus distance, which depends on the data distributions within each cluster. Therefore, the consensus distance serves as an effective metric to reflect the impact of links in the model convergence. Specifically, when two clusters have similar data distribution, their corresponding edge models tend to be similar as well, resulting in smaller consensus distances. In such cases, removing the link between their edge servers is less critical and can be executed safely. However, if two edge clusters have very different data distributions, their edge models become quite different, resulting in larger consensus distances. In this scenario, removing the link between these edge servers may hinder the convergence of the model.

B. Consensus Distance Constraint

Integrating the consensus distance into our problem presents two key challenges. First, we can only calculate the consensus distance between directly connected edge server. For unconnected servers, direct computation is not possible, making it difficult to use consensus distance as a guiding metric for topology design. Second, consensus distance does not have an explicit functional dependence on the graph topology, complicating its integration into optimization problems alongside other decision variables.

Instead of directly computing the consensus distance between two edge models, we estimate the consensus distance between the server model $\mathbf{u}_c^{t,R-1}$ and the global averaged model $\bar{\mathbf{u}}^{t,R-1}$. Based on the definition of consensus distance (17) and the aggregation rule (15), we have

$$\begin{aligned}&\Upsilon_c^{t+1} = \left\|\bar{\mathbf{u}}^{t+1,0} - \mathbf{u}_c^{t+1,0}\right\| \\ &= \Big\|\frac{1}{C}\sum_{c'=1}^{C}\mathbf{u}_{c'}^{t,R-1} - (\mathbf{u}_c^{t,R-1} + \sum_{c'\in\{c\}\cup\mathcal{N}_c^t}\mathbf{M}_{c,c'}^{(\psi)}(\mathbf{u}_{c'}^{t,R-1} - \mathbf{u}_c^{t,R-1}))\Big\| \\ &= \Big\|\textstyle\sum_{c'=1}^{C}\frac{\mathbf{u}_{c'}^{t,R-1} - \mathbf{u}_c^{t,R-1}}{C} - \mathbf{A}_{c,c'}\mathbf{M}_{c,c'}^{(\psi)}(\mathbf{u}_{c'}^{t,R-1} - \mathbf{u}_c^{t,R-1}))\Big\| \end{aligned} \tag{22}$$

For simplicity, we set $\mathbf{M}_{c,c'}^{(\psi)} = 1/N$. Note that it is the possible maximum value [36], [40], [41]. Then we have

$$\begin{aligned}\mathbb{E}\Upsilon_c^{t+1} &= \Big\|\textstyle\sum_{c'=1}^{C}\frac{(1-\mathbf{A}_{c,c'})(\mathbf{u}_{c'}^{t,R-1} - \mathbf{u}_c^{t,R-1})}{C}\Big\| \\ &\leq \tfrac{1}{C}\textstyle\sum_{c'=1}^{C}(1-\mathbf{A}_{c,c'})\Upsilon_{c,c'}^{t}.\end{aligned} \tag{23}$$

We take the average across all edge models on both sides of (23)

$$\mathbb{E}\Upsilon^{t+1} \leq \frac{1}{C^2}\textstyle\sum_{c=1}^{C}\sum_{c'=1}^{C}(1-\mathbf{A}_{c,c'})\Upsilon_{c,c'}^{t}. \tag{24}$$

Next, we expect the upper bound (24) is smaller than the threshold. This leads to the following consensus constraint:

$$\frac{1}{C^2}\textstyle\sum_{c=1}^{C}\sum_{c'=1}^{C}(1-\mathbf{A}_{c,c'})\Upsilon_{c,c'}^{t} \leq \Upsilon_{\max}^{t}, \forall t. \tag{25}$$

The inequality (25) explicitly establishes the connection between the graph topology and the consensus distance. It allows us to capture the influence of topology design on model convergence. Specifically, a relatively high value of $\Upsilon_{c,c'}^{t}$ indicates a large consensus distance between the server c and c', suggesting a substantial disparity of their edge models. In such cases, removing the communication link between these servers may negatively impact the rate of model convergence. In contrast, a smaller consensus distance implies minimal differences between the edge models. The link between them can be removed to reduce the communication overhead without significantly affecting the convergence rate. Note that if two servers are not connected, directly computing $\Upsilon_{c,c'}^{t}$ is infeasible. In this case, we adopt the approach in the previous work [36], [37] to estimate $\Upsilon_{c,c'}^{t}$.

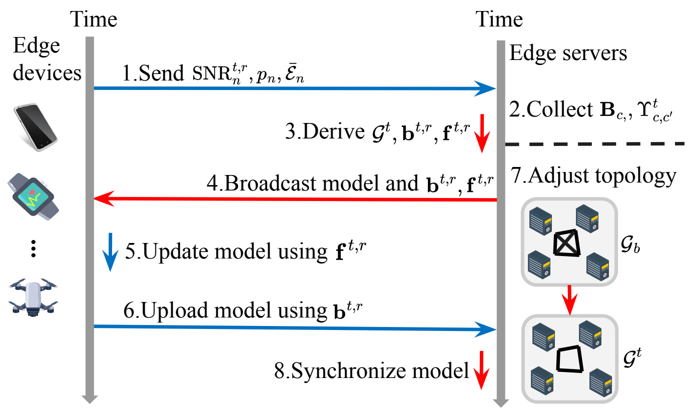
V. Problem Formulation

A. Formulated Optimization Problem

Our goal is to minimize the total time cost (during local updating and model transmission) while ensuring model convergence. Specifically, we aim to devise a control strategy by adjusting the decision variables, e.g., graph topology, communication bandwidth, and CPU frequency, which translates into the following problem:

$$\textbf{P1:} \quad \min_{\{b_n^{t,r},\forall n,t,r\},\{f_n^{r,t},\forall n,t,r\},\{\mathcal{G}^t,\forall t\}} \sum_{t=0}^{T-1}\mathcal{T}^t$$

$$\text{s.t.}\begin{cases}\Upsilon^{t+1} \leq \Upsilon_{\max}^{t}, \forall t & (26)\\ \sum_{t=0}^{T-1}\sum_{r=0}^{R-1}\mathcal{E}_n^{t,r} \leq \bar{\mathcal{E}}_n, \forall n & (27)\\ (5), (8), (14). \end{cases}$$



Time
Time
Edge devices
Edge servers
1.Send $\mathrm{SNR}_n^{t,r}, p_n, \bar{\mathcal{E}}_n$
2.Collect $\mathbf{B}_c, \Upsilon_{c,c'}^t$
3.Derive $\mathcal{G}^t, \mathbf{b}^{t,r}, \mathbf{f}^{t,r}$
4.Broadcast model and $\mathbf{b}^{t,r}, \mathbf{f}^{t,r}$
7.Adjust topology
5.Update model using $\mathbf{f}^{t,r}$
$\mathcal{G}_b$
6.Upload model using $\mathbf{b}^{t,r}$
$\mathcal{G}^t$
8.Synchronize model

Fig. 2. Flowchart of FedRT.

The objective of Problem **P1** represents the total time cost of T global rounds. Inequality (26) states that the consensus distance should not exceed a threshold $\Upsilon_{\max}^t$. The role of this constraint is two-fold. First, we use it to mitigate the negative influence caused by the non-IID data distribution between clusters. Additionally, it ensures the convergence of the model when we update the graph topology. Inequality (27) represents the constraint on energy consumption, where $\bar{\mathcal{E}}_n$ is the energy budget of the device n.

B. Challenges to address the problem

Solving Problem **P1** presents several challenges: (i) The model convergence rate is directly influenced by the graph topology. Removing slow communication links can negatively affect the convergence speed, potentially increasing training latency. (ii) Control decisions within the objective function and constraints are interconnected across time slots. However, we lack future knowledge of the system stats such as cluster communication environment and edge backhaul speed. Consequently, an online algorithm that operates without reliance on pre-existing system information or assumptions is required. (iii) The updates frequencies of the graph topology and resource allocation differ, complicating their joint online optimization.

VI. Solution Design

To address Problem **P1**, we first decompose it into a series of subproblems alongside the global training round t. Then, the sub-problem at the global round t is

$$\textbf{P2:} \quad \min_{\{b_n^{r,t},\forall n,r\},\{f_n^{r,t},\forall n,r\},\mathcal{G}^t} \mathcal{T}^t$$

$$\text{s.t.} \begin{cases} \frac{1}{C^2}\sum_{c=1}^{C}\sum_{c'=1}^{C}(1-\mathbf{A}_{c,c'}^t)\Upsilon_{c,c'}^t \le \Upsilon_{\max}^t \\ \sum_{t'=0}^{t-1}\sum_{r=0}^{R-1}\mathcal{E}_n^{t',r} + (T-t)\sum_{r=0}^{R-1}\mathcal{E}_n^{t,r} \le \bar{\mathcal{E}}_n, \forall n \\ (5),(8),(14). \end{cases}$$

Here, the second inequality uses the energy cost in the current global round t to estimate the total energy cost of T global rounds.

Algorithm 2 FedRT

```
1:  for $t = 0$ to $T-1$ do
2:    for $r = 0$ to $R-1$ do
3:      if $r = R-1$ then
4:        Server records $\mathbf{B}_c$, and $\Upsilon_{c,c'}^t$, and send them to the
          coordinator (it can be any server)
5:      end if
6:      Device observes channel statistics $\mathrm{SNR}_n^{t,r}$ and send
        it to coordinator
7:      if $r \neq R-1$ then
8:        Solve Problem **P2.1** through CVX
9:      else
10:       Solve Problem **P2.2** via **Algorithm 3**
11:     end if
12:     Run one edge round collaborative training
13:     if $r = R-1$ then
14:       Server synchronizes model with neighbor servers
15:     end if
16:   end for
17: end for
```

However, solving the above **P2** still is infeasible. One obstacle is the server only starts to communicate with neighbor servers after they finish their training task within each cluster. The graph topology should be derived based on the time cost of the R edge training rounds and the consensus distance between the server models. In the edge round $r < R-1$, the future communication topology is still unknown.

In order to address this challenge, we again break down **P2** into a sequence of subproblems across different edge rounds. Specifically, for the edge round $r < R-2$, we allocate the CPU frequency and communication bandwidth for each device such that the estimated completion time for each edge round is minimized. In the edge round $R-1$, we jointly optimize the CPU frequency, communication bandwidth, and graph topology based on the completion time and the energy used.

Let $\mathbf{b}^{t,r} = \{b_n^{t,r},\forall n\}$, $\mathbf{f}^{t,r} = \{f_n^{t,r},\forall n\}$ for $r = 0,\dots,R-1$. At global round t, the completion time from the first edge round to edge round r can be estimated as follows:

$$\mathcal{T}^{t,r} = \max_{c\in[C]}\Big\{\sum_{r'=0}^{r-1}\max_{n\in\mathcal{S}_c}\{\mathcal{T}_n^{t,r',\text{cmp}} + \mathcal{T}_n^{t,r',\text{com}}\} + \max_{n\in\mathcal{S}_c}\{\mathcal{T}_n^{t,r,\text{cmp}} + \mathcal{T}_n^{t,r,\text{com}}\}\Big\} \quad (28)$$

Then, the sub-problem at edge round r and global round t can be expressed as

$$\textbf{P2.1:} \quad \min_{\mathbf{b}^{t,r},\mathbf{f}^{t,r}} \mathcal{T}^{t,r}$$

$$\text{s.t.} \begin{cases} \sum_{t'=0}^{t-1}\sum_{r'=0}^{R-1}\mathcal{E}_n^{t',r'} + \sum_{r'=0}^{r-1}\mathcal{E}_n^{t,r'} \\ \qquad +((T-t)R + R - r)\mathcal{E}_n^{t,r} \le \bar{\mathcal{E}}_n, \forall n \\ (5),(14). \end{cases}$$

Note that the problem is convex and can be efficiently solved using convex optimization software such as CVX.

Algorithm 3 Algorithm to address **P2.2**

```
Input: $\Lambda$, $S$, $I$, $\mathbf{B}^t$, $\zeta$, $\Upsilon^t_{\max}$, $\mathcal{G}_b$, $\{\mu_n\}$, $\{p_n\}$, $\{\alpha_n\}$, $\{\mathrm{SNR}^{t,R}_n, \forall n\}$, $\{\Upsilon^t_{c,c'}, \forall c, \forall c'\}$
Output: $\mathbf{b}^{t,R-1}$, $\mathbf{f}^{t,R-1}$, $\mathcal{G}^t$
1: Initilize the optimal topology $(\mathcal{G}^{t,R-1})^* = \mathcal{G}_b$ and $Flag = True$
2: Solve P2.1 to obtain an initial resource allocation $(\mathbf{b}^{t,R-1})^*, (\mathbf{f}^{t,R-1})^*$ based on $(\mathcal{G}^{t,R-1})^*$
3: Initilize the best completion time using (29)
   $(\mathcal{T}^{t,R-1})^* \leftarrow \mathcal{T}^{t,R-1}((\mathbf{b}^{t,R-1})^*, (\mathbf{f}^{t,R-1})^*, (\mathcal{G}^{t,R-1})^*)$
4: while $True$ do
5:     if $Flag$ then
6:         $e = \lfloor \sqrt{\sum_{c,c'} (\mathbf{A}^t_{c,c'})^*} \rfloor$
7:     else
8:         $e = \lfloor e/2 \rfloor$
9:     end if
10:    Select $e$ slowest links from $(\mathcal{G}^{t,R-1})^*$ within the consensus distance threshold (25), and place them into $\bar{\mathcal{E}}$.
11:    Sort the links in $\bar{\mathcal{E}}$ in ascending order of speed
12:    Initilize temporal topology $(\mathcal{G}^{t,R-1})' \leftarrow (\mathcal{G}^{t,R-1})^*$
13:    for $E_{c,c'} \in \bar{\mathcal{E}}$ do
14:        Remove $E_{c,c'}$ from $(\mathcal{G}^{t,R-1})'$
15:        if $(\mathcal{G}^t)^*$ is not connected then
16:            Restore $E_{c,c'}$ into $(\mathcal{G}^{t,R-1})'$
17:        end if
18:    end for
19:    Solve P2.1 to obtain new resource allocation $(\mathbf{b}^{t,R-1})', (\mathbf{f}^{t,R-1})'$ based on $(\mathcal{G}^{t,R-1})'$
20:    Compute new completion time
       $(\mathcal{T}^{t,R-1})' \leftarrow \mathcal{T}^{t,R-1}((\mathbf{b}^{t,R-1})', (\mathbf{f}^{t,R-1})', (\mathcal{G}^{t,R-1})')$
21:    if $(\mathcal{T}^{t,R-1})' < (\mathcal{T}^{t,R-1})^*$ then
22:        $(\mathbf{b}^{t,R-1})^*, (\mathbf{f}^{t,R-1})^* \leftarrow (\mathbf{b}^{t,R-1})', (\mathbf{f}^{t,R-1})'$
23:        $(\mathcal{T}^{t,R-1})^*, (\mathcal{G}^{t,R-1})^* \leftarrow (\mathcal{T}^{t,R-1})', (\mathcal{G}^{t,R-1})'$
24:        $Flag \leftarrow True$
25:    else
26:        $Flag \leftarrow False$
27:    end if
28:    if not $Flag$ and $e == 1$ then
29:        Break
30:    end if
31: end while
32: $\mathcal{G}^{t,R-1}, \mathbf{b}^{t,R-1}, \mathbf{f}^{t,R-1} \leftarrow (\mathcal{G}^{t,R-1})^*, (\mathbf{b}^{t,R-1})^*, (\mathbf{f}^{t,R-1})^*$
```

In the edge round $R-1$, the backhaul communication bandwidth matrix B^t is available. The completion time of global round r can be estimated as follows:

$$\mathcal{T}^{t,R-1} = \max_{c\in[C]} \Big\{ \sum_{r'=0}^{R-2} \max_{n\in\mathcal{S}_c} \{ \mathcal{T}_n^{t,r',\mathrm{cmp}} + \mathcal{T}_n^{t,r',\mathrm{com}} \} + \max_{n\in\mathcal{S}_c} \{ \mathcal{T}_n^{t,R-1,\mathrm{cmp}} + \mathcal{T}_n^{t,R-1,\mathrm{com}} \} + \mathcal{G}^t \Big\} \tag{29}$$

Then, the sub-problem at edge round $R-1$ and global round t can be formulated as

$$\mathbf{P2.2:} \quad \min_{\mathbf{b}^{t,R-1}, \mathbf{f}^{t,R-1}, \mathcal{G}^t} \mathcal{T}^{t,R-1}$$

$$\text{s.t.} \begin{cases} \frac{1}{C^2} \sum_{c=1}^{C} \sum_{c'=1}^{C} (1 - \mathbf{A}^t_{c,c'}) \Upsilon^t_{c,c'} \le \Upsilon^t_{\max} \\ \sum_{t'=0}^{t-1} \sum_{r'=0}^{R-1} \mathcal{E}_n^{t',r'} + \sum_{r'=0}^{R-2} \mathcal{E}_n^{t,r'} \\ \quad +((T-t)R+1)\mathcal{E}_n^{t,R-1} \le \bar{\mathcal{E}}_n, \forall n \\ (5), (8), (14). \end{cases}$$

Problem **P2.2** is non-convex, involving combinatorial optimization with coupled discrete graph topology and continuous variables like device communication bandwidth and computation frequency, making it difficult to derive optimal solution with theoretical guarantees. Therefore, we tackle **P2.2** using a greedy search approach. Specifically, we alternatively update the continuous varibles, i.e., the communication bandwidth $\mathbf{b}^{t,R-1}$ and CPU frequency $\mathbf{f}^{t,R-1}$ for each device, and the discrete variable, i.e., the server communication graph topology $\mathcal{G}^t$. This iterative process continues until no further reduction in completion time can be achieved. Subsequently, the entire problem **P2** can be effectively solved online.

We summarize the detailed procedure in **Algorithm 2** and illustrate the overall solution process in Fig. 2. Specifically, in each edge round, edge devices first transmit locally collected information to the coordinator, including the signal-to-noise ratio for wireless communication $\mathrm{SNR}^{t,r}_n$, transmission power p_n, and energy budget $\bar{\mathcal{E}}_n$ (Step 1). Concurrently, edge servers record the inter-server bandwidth $\mathbf{B}_c$ and compute the consensus distance $\Upsilon^t_{c,c'}$ between edge models (Step 2). For disconnected edge servers, the consensus distance is estimated following previous work [36], [37]. Using the collected information, the coordinator solves subproblems **P2.1** and **P2.2** online to allocate communication and computation resources and to design the graph topology (Step 3). In the first $R-1$ edge rounds of each global round, the coordinator uses CVX

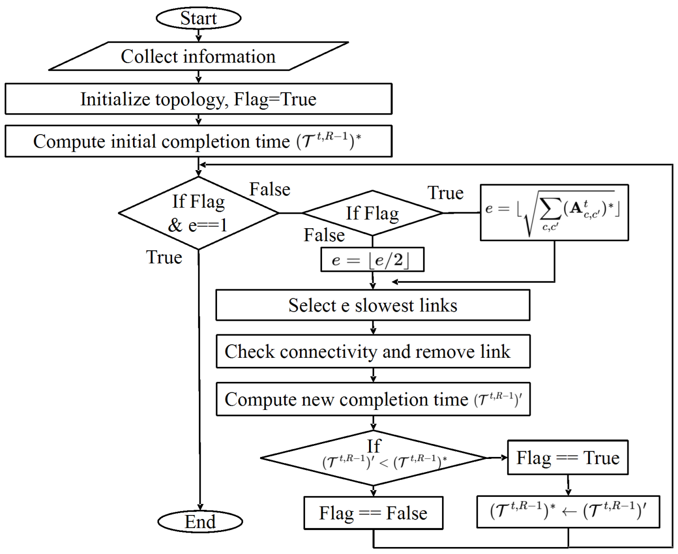

Start
Collect information
Initialize topology, Flag=True
Compute initial completion time $(\mathcal{T}^{t,R-1})^*$
If Flag & e==1
False
True
If Flag
True
False
$e = \lfloor \sqrt{\sum_{c,c'} (\mathbf{A}^t_{c,c'})^*} \rfloor$
$e = \lfloor e/2 \rfloor$
Select e slowest links
Check connectivity and remove link
Compute new completion time $(\mathcal{T}^{t,R-1})'$
If $(\mathcal{T}^{t,R-1})' < (\mathcal{T}^{t,R-1})^*$
Flag == True
Flag == False
$(\mathcal{T}^{t,R-1})^* \leftarrow (\mathcal{T}^{t,R-1})'$
End

Fig. 3. Flowchart to address **P2.2**.

to determine bandwidth allocations $\mathbf{b}^{t,r}$ and CPU frequencies $\mathbf{f}^{t,r}$ for each device, while preserving the existing topology. In the final edge round, Algorithm 3 is invoked to jointly determine the bandwidth $\mathbf{b}^{t,r}$, the CPU frequency $\mathbf{f}^{t,r}$ and the new graph topology $\mathcal{G}^t$. Subsequently, the coordinator broadcasts the solution to all edge devices and servers. Each edge server also transmits their latest server model to its connected edge devices (Step 4). Each device then adjusts its computational resources according to the assigned CPU frequency $\mathbf{f}^{t,r}$ and begin the local training (Step 5). The updated local models are later uploaded to edge servers using the allocated bandwidth $\mathbf{b}^{t,r}$ for aggregation (Step 6). Finally, in the last edge round, the edge servers synchronize the latest edge model with neighboring servers according to the new topology $\mathcal{G}^t$ to achieve global model consensus (Steps 7 and 8).

The solving procedure for Problem **P2.2** is summarized in **Algorithm 3** and illustrated through the flow chart presented in Fig. 3. Specifically, the coordinator first initializes the optimal graph topology $(\mathbf{A}^t)^*$ as the base graph topology $\mathbf{A}_b$ (Line 1). Then, it solves **P2.1** based on $(\mathbf{A}^t)^*$ to determine the initial allocation of bandwidth $(\mathbf{b}^{t,R-1})^*$ and frequency assignment $(\mathbf{f}^{t,R-1})^*$ (Line 2). The initial best completion time $(\mathcal{T}^{t,R-1})^*$ is then calculated based on the obtained $(\mathbf{b}^{t,R-1})^*$, $(\mathbf{f}^{t,R-1})^*$, $(\mathcal{G}^t)^*$ (Line 3). Subsequently, the coordinator iteratively performs the following two-step procedure until no further reduction in completion time can be achieved: (i) updates $\mathcal{G}^t$ by link speed with fixed $(\mathbf{b}^{t,R-1}, \mathbf{f}^{t,R-1})$ (Lines 5-17); (ii) updates bandwidth $(\mathbf{b}^{t,R-1})$ and frequency assignments $(\mathbf{f}^{t,R-1})$ with fixed $\mathcal{G}^t$ (Line 18).

In particular, the coordinator first selects e slowest links as candidates for removal. The initial number of e is set as the square root of the total number of links in the current graph $(\mathcal{G}^t)^*$ (Line 6). Then, the selected links are sorted in ascending order according to their speed (Line 11). For each link, the coordinator attempts to remove it from the current graph $(\mathcal{G}^t)^*$. If removing a link disconnects the graph, the link is restored to maintain connectivity (Lines 12–17). Using the updated graph, the coordinator solves Problem **P2.1** via CVX to obtain new value of $(\mathbf{b}^{t,R-1})$ and $(\mathbf{f}^{t,R-1})$ (Line 18), and calculates the new completion time $(\mathcal{T}^{t,R-1})'$ (Line 19).

If the new completion time $(\mathcal{T}^{t,R-1})'$ is shorter than the best completion time $(\mathcal{T}^{t,R-1})^*$ (Line 20), the coordinator updates the optimized $(\mathbf{b}^{t,R-1})^*, (\mathbf{f}^{t,R-1})^*, (\mathcal{G}^{t,R-1})^*$ to $(\mathbf{b}^{t,R-1})', (\mathbf{f}^{t,R-1})', (\mathcal{G}^{t,R-1})'$ (Lines 21-22). The best completion time $(\mathcal{T}^{t,R-1})^*$ is updated to $(\mathcal{T}^{t,R-1})'$ as well (Line 22). Conversely, if the new completion time is larger than the best completion time, the original topology is retained (Line 25), and e will be halved in the next search iteration (Line 8). This iterative search process will be terminated until there is no link that can be removed from the current graph (Line 27). Upon termination, the optimized bandwidth allocation $(\mathbf{b}^{t,R-1})^*$, frequency assignment $(\mathbf{f}^{t,R-1})^*$, and graph topology $(\mathcal{G}^t)^*$ are returned as the final solution (Line 31). Note that although this heuristic does not guarantee global optimality, its careful and adaptive selection ensures consistent improvement at each iteration.

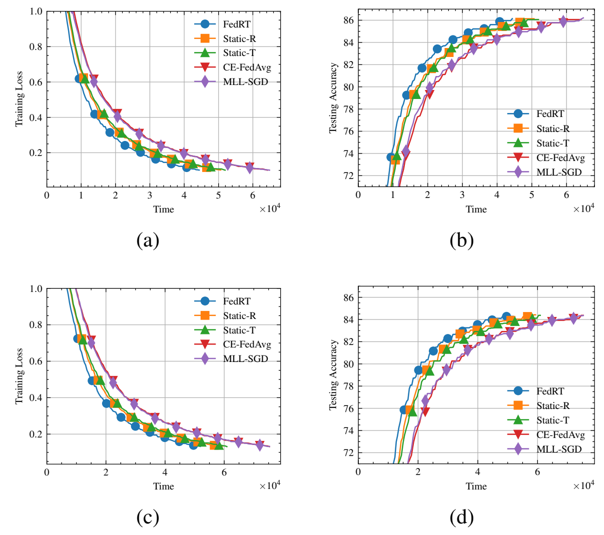


Fig. 4. Training loss (a,c) and testing accuracy (b,d) comparison of FedRT and baselines under IID (a,b) and non-IID (c,d) distribution on CIFAR-10.

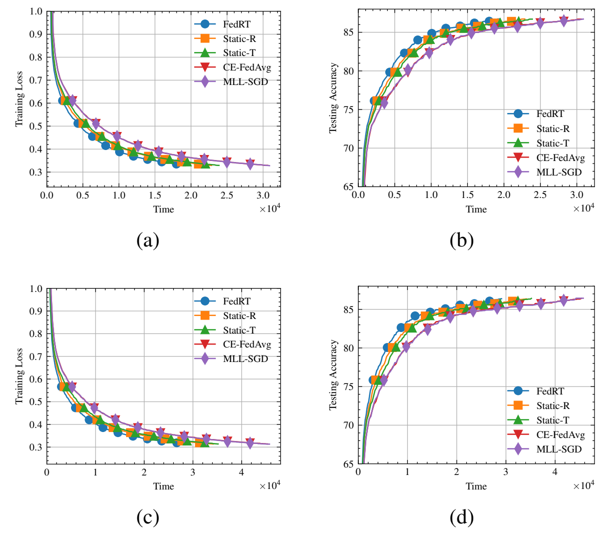


Fig. 5. Training loss (a,c) and testing accuracy (b,d) comparison of FedRT and baselines under IID (a,b) and non-IID (c,d) distribution on FMNIST.

VII. Experiment

We consider a HFEL system with 72 devices and 8 servers (clusters). Each cluster has 9 devices and 1 server. In the experiments, we employ three image classification datasets: FEMNIST [22], CIFAR-10 [21], and FMNIST [42]. The FEMNIST dataset is the federated splitting version of the EMNIST dataset, which includes 3,550 writers. We randomly sample 72 writers to simulate the practical HFEL application. we divide each writer's local data into 90% for training and 10% for testing. For evaluation purposes, a common testing dataset is constructed by aggregating the testing data from all devices. Note that FEMNIST is a non-IID distributed dataset due to the variations in writing styles among different writers.

TABLE I
TEST ACCURACY AND RESOURCE USAGE COMPARISON OF FEDRT AND BASELINES FOR DIFFERENT DATASETS. NON-IID IS DIRICHLET DISTRIBUTION. TIME DENOTES THE TOTAL TRAINING LATENCY IN HOURS. SAVE DENOTES THE PERCENTAGE OF TIME SAVED BY FEDRT COMPARED TO THE BASELINES. ENG REPRESENTS THE TOTAL ENERGY CONSUMPTION (J). ACC IS THE BEST TESTING ACCURACY.

Method	**CIFAR-10**								**FMNIST**								**FEMNIST**			
	IID				non-IID				IID				non-IID				/			
	Time ↓	Save↑	Eng ↓	Acc ↑	Time ↓	Save↑	Eng ↓	Acc ↑	Time ↓	Save↑	Eng ↓	Acc ↑	Time ↓	Save↑	Eng ↓	Acc ↑	Time ↓	Save↑	Eng ↓	Acc ↑
FedRT	**12.41**	/	864.12	**86.14**	**14.46**	/	1004.05	84.28	**5.41**	/	**119.81**	**86.73**	**8.08**	/	177.93	86.38	**7.86**	/	450.55	**76.76**
Static-R	14.16	12.35	899.06	86.13	16.52	12.47	1047.93	**84.42**	6.35	14.80	122.52	86.72	9.49	14.86	181.57	86.35	9.04	13.05	470.23	**76.76**
Static-T	14.52	14.53	**861.47**	86.06	16.98	14.84	**1001.58**	84.37	6.63	18.40	119.83	86.70	9.77	17.30	**177.92**	86.42	9.40	16.38	**449.51**	76.41
CE-FedAvg	17.95	30.86	933.47	86.06	20.99	31.11	1087.29	84.37	8.51	36.43	127.57	86.70	12.59	35.82	187.10	86.42	11.73	32.99	489.42	76.41
MLL-SGD	17.96	30.90	925.99	86.10	21.00	31.14	1078.44	84.33	8.51	36.43	127.93	86.68	12.59	35.82	187.65	**86.44**	11.62	32.36	450.71	76.57

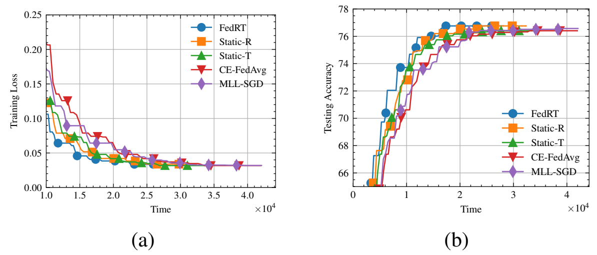


Fig. 6. Training loss (a,c) and testing accuracy (b,d) comparison of FedRT and baselines under IID (a,b) and non-IID (c,d) distribution on FEMNIST.

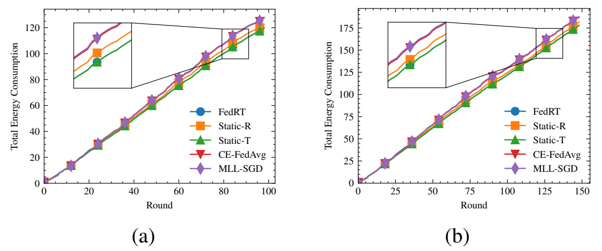


Fig. 8. Cumulative energy consumption comparison of FedRT and baselines under IID (a) and non-IID (b) distribution on FMNIST.

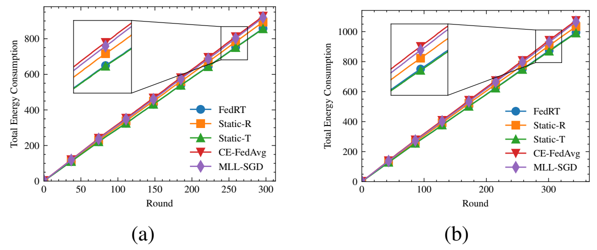


Fig. 7. Cumulative energy consumption comparison of FedRT and baselines under IID (a) and non-IID (b) distribution on CIFAR-10.

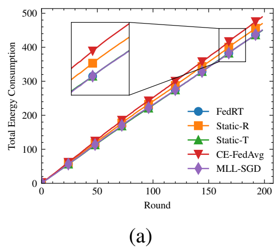


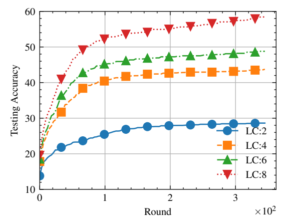
Fig. 9. Cumulative energy consumption comparison of FedRT and baselines on FEMNIST.

We train a modified ResNet-20 for FEMNIST, comprising a total of 272,814 parameters.

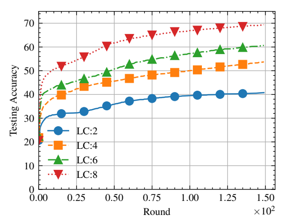
The CIFAR-10 dataset is composed of 50,000 training images and 10,000 testing images. To simulate the practical data distribution scenarios, we adopt three data partition strategies: IID distribution, Dirichlet distribution, and cluster-pathological distribution. In the case of IID distribution, the 50,000 training images are evenly and randomly distributed among all the devices, ensuring each device receives an equal share of the dataset. For Dirichlet distribution, the partitioning of the 50,000 training images among devices follows the Dirichlet distribution [43] with a concentration parameter of 1. For cluster-pathological distribution, we first partition the 50,000 training images into 8 clusters based on the given number of labels per cluster (LC). Within each cluster, the data are distributed among all the devices in an IID fashion, similar to the IID distribution strategy. This method aims to capture the characteristics of clustered data in real-world scenarios. We train a ResNet-20 (269,722 parameters) on CIFAR-10. The original 10,000 testing images are used as the common testing dataset.

The FMNIST dataset comprises 60,000 training images and 10,000 testing images. Similar to the CIFAR-10 dataset, we utilize three data partitioning strategies in the experiments. We train a LeNet-5 (431,080 parameters) on FMNIST. The original 10,000 testing images are used as the common testing dataset. To demonstrate the effectiveness of FedRT, we compare it with three baselines: *Static-R*, *Static-T*, and *CE-FedAvg*. The details of baseline algorithms are as follows:

- *Static-R* assumes that each edge server evenly assigns its communication bandwidth among all edge devices within its cluster. To fulfill the energy requirements of the HFEL system, we solve the optimization problem as in FedRT to determine the feasible CPU frequency and the edge backhaul topology, except the allocated communication bandwidth remains constant.
- *Static-T* adopts a static edge backhaul topology and only considers the resource allocation for edge devices. To satisfy the energy constraint, the communication bandwidth and CPU frequency are optimized following the same methodology as FedRT, except the edge backhaul topology is fixed.

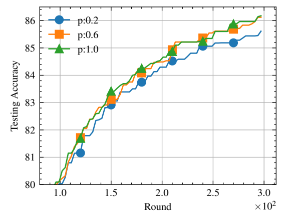


(a) CIFAR-10

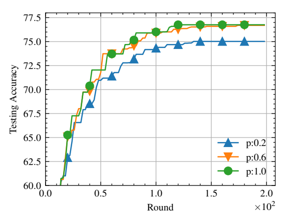


(b) FMNIST

Fig. 10. FedRT under different levels of inter-cluster data heterogeneity.



(a) CIFAR-10



(b) FEMNIST

Fig. 11. The performance of FedRT under different base graph topologies.

- *Ce-FedAvg [8]* assumes that all edge devices are homogeneous and the edge backhaul topology is given a prior. We adapt it to our problem setting by employing the fixed edge backhaul topology and static resource allocation. Specifically, the server communicates following the base edge backhaul topology. The communication bandwidth is evenly assigned among edge devices. To satisfy the energy requirement, similar to *Static-R* and *Static-T*, we resolve the optimization problem as in FedRT to obtain a feasible CPU frequency for each edge device.
- *MLL-SGD* [44] adaptively assigns different local training iterations to each device based on their resource capabilities. For instance, slower devices are assigned fewer training steps to save computation time, while faster devices receive more steps to maintain model performance. In adapting this approach to our problem setting, we set different numbers of local training iterations according to each edge device's computation resource. Similar to Ce-FedAvg, we solve the optimization as FedRT with fixed edge backhaul topology and homogeneous communication bandwidth to derive the feasible CPU frequency for edge devices.

A. Datasets and Models

For all experiments, we use mini-batch SGD with 0.9 momentum [8] to train the local model with a batch size of 32 [24]. For the learning rate, we performed grid search with values of {0.01, 0.05, 0.1} for CIAFR-10 and {0.1, 0.06, 0.03, 0.01} for FEMNIST. The number of local iterations is 10 for all datasets. The number of edge rounds is 2 for all experiments [8]. The number of global rounds is determined through preliminary experiments for each model. Specifically, we train each model on its respective dataset with surfficient global rounds across all algorithms, recording both training loss and testing accuracy at each global round. The number of global rounds is selected based on the point at which the accuracy curves for all algorithms plateau. For CIAFR-10, we run 300 global rounds under IID data distribution and 350 rounds under non-IID data distribution. For FEMNIST, the total number of global rounds is 200. For FMNIST, the total number of global rounds is 100 under the IID data distribution and 150 under the non-IID distribution. We run each experiment with 3 random seeds and report the average metric. The number of server communication times in each global round is $\psi = 10$ [8].

We record the total completion time and testing accuracy for performance evaluation. We use thop[2] to estimate the computation workload in terms of the number of floating point operations (FLOPs). The number of FLOPs needed for each training sample per iteration is 123.9 MFLOPs for ResNet-20 on CIFAR-10, 94.2 MFLOPs for ResNet-20 on FEMNIST, and 3.9 MFLOPs for LeNet-5 on FMNIST, respectively. To simulate a practical heterogeneous HFL system, we adopted parameter settings and device characteristics from previous works [20], [23], [32], [36], [37], [45], as well as realistic IoT device parameters. Specifically, the device's communication power is set to 0.01 W [33]. The maximum available CPU frequency $f_n^{\max}$ is 3.0GHz and the minimum CPU frequency $f_n^{\min}$ is 2.0GHz [45]. In practice, edge devices usually have heterogeneous computation capabilities. To mimic the CPU capability of real-world edge devices, we generate the efficient capacity coefficient for each device by generating an efficient capacity coefficient for each device. This coefficient is the product of a constant factor and a random variable. We adopt the value of efficient capacity coefficient $\alpha_n = 2 \times 10^{-28}$ as per [45], and the random variable is uniformly distributed within range $[0.01, 0.1]$. The total available communication bandwidth for all servers is set to $B = 1\text{MHz}$ for all servers [23]. To simulate the realistic wireless communication, we assume the SNR are uniformly distributed within the range $0 - 15$ dB for all devices and all time slots, resulting in heterogeneous communication channels. The available energy supply for all devices $\bar{\mathcal{E}}_n = 1\text{J}$ [33]. By default, we assume a fully connected base graph topology unless otherwise specified. To model heterogeneous edge backhaul communication bandwidth, we assume $\mathbf{B}^t$ fluctuates randomly (uniformly) within the range $0.1 - 10$ Mbps [23].

B. Experimental Results

Performance Comparison with Baselines. We first evaluate the convergence speed of FedRT in comparison to the baseline methods. Fig. 4 shows the testing accuracy and training loss w.r.t training time (in seconds) on CIFAR-10 under both IID and Dirichlet data distribution. The results indicate that FedRT exhibits a faster convergence speed than baselines. Throughout the training process, FedRT consistently achieves higher testing accuracy and lower training loss under both IID and non-IID data distributions under the same training time budget. Moreover, FedRT reduces the total training latency

[2]https://pypi.org/project/thop/

on CIFAR-10 by 30.90% and 31.14% for IID and non-IID distributions, respectively. Fig. 5 depicts the comparison results on FMNIST under IID and non-IID data distribution. Fig. 6 presents the comparison results on FEMNIST. Similarly, FedRT has a faster convergence speed than baselines. On the FMNIST dataset, the total training latency of FedRT is reduced by 36.43% and 35.82% under IID and non-IID data distribution, respectively. Moreover, the total training latency is reduced by 32.35% compared to baselines on the FEMNIST dataset. The results of Fig. 4, Fig. 5, and Fig. 6 demonstrate that FedRT consistently outperforms all baseline methods across both IID and non-IID data distributions, thereby demonstrating the effectiveness of the proposed method.

Note that Static-R and Static-T are simplified variants of FedRT, excluding resource optimization and topology design, respectively. As shown in Figs. 4, 5, and 6, Static-R consistently outperforms CE-FedAvg across all datasets, demonstrating the effectiveness of the consensus distance estimation approach presented in Section IV-B. Furthermore, the performance gains of FedRT over all baselines further validate the effectiveness of **Algorithm 3**. If our proposed methods were ineffective, achieving simultaneous and consistent improvements over baselines that optimize only individual aspects would not be feasible.

Our experimental results demonstrate FedRT's consistent latency advantages across both IID and non-IID settings (Table I). For CIFAR-10 under IID conditions, FedRT completes training in 12.41 hours, achieving time savings of 12.35% (vs Static-R's 14.16 hours), 14.53% (vs Static-T's 14.52 hours), and 30.86%-30.90% (vs CE-FedAvg's 17.95 hours and MLL-SGD's 17.96 hours). Under non-IID conditions, FedRT maintains superior performance at 14.46 hours compared to Static-R (16.52 hours), Static-T (16.98 hours), CE-FedAvg (20.99 hours), and MLL-SGD (21.00 hours), preserving time savings of $12.47\% - 31.14\%$. While all methods experience comparable relative slowdowns (16.5%-16.9%) from IID to non-IID conditions, FedRT's absolute latency remains substantially lower - maintaining at least a 12.5% advantage and up to 31.1% faster than baselines even in heterogeneous data environments.

A similar trend is observed in the FMNIST dataset. Under IID conditions, FedRT completes training in 5.41 hours, saving 36.43% over CE-FedAvg and MLL-SGD, and around 14.80% and 18.40% over Static-R and Static-T. In the non-IID case, the time required by FedRT increases to 8.08 hours, while the baselines again exhibit an increase: Static-R reaches 9.49 hours, Static-T 9.77 hours, CE-FedAvg and MLL-SGD 12.59 hours. As a result, the time-saving ratio of FedRT reaching 35.82% compared to CE-FedAvg and 17.30% compared to Static-R. For the FEMNIST dataset, which is inherently non-IID in nature. FedRT requires only 7.86 hours of training time. This achieves a time saving of 13.05% compared to Static-R, 16.38% compared to Static-T, and approximately 32% over CE-FedAvg and MLL-SGD.

Energy Consumption Comparsion. We then compare the energy consumption of FedRT against several baselines. Fig. 7 shows the curve of accumulated energy consumption for edge devices w.r.t the training round on CIFAR-10 under both IID and Dirichlet data distribution. The results indicate that FedRT consistently consumes less energy than the Static-R, CE-FedAvg, and MLL-SGD throughout the training process. As the number of training rounds increases, the discrepancy between the curves becomes more pronounced, demonstrating that our algorithm consistently reduces energy costs, and its advantage becomes increasingly evident. Furthermore, the energy consumption pattern of FedRT closely aligns with that of Static-T, which also implements resource allocation for devices. This observation underscores the effectiveness of FedRT in optimizing resources for energy conservation. Moreover, we can find that FedRT consumes more energy under non-IID distribution than IID, highlighting the critical need for resource optimization under data heterogeneity. Fig. 8 and Fig. 9 show the accumulated energy consumption w.r.t the training round on FMNIST and FEMNIST datasets. We have similar conversations about energy consumption. In conclusion, FedRT achieves better energy usage compared with Static-R, CE-FedAvg, and MLL-SGD baselines. Although the energy consumption of FedRT is nearly identical to that of Static-T, FedRT achieves a faster convergence rate, showing an advantage in training performance.

Energy, and Accuracy. Table. I presents a comprehensive comparison of the total energy consumption and the highest testing accuracy across all datasets and methods. FedRT achieves the highest final testing accuracy on the CIFAR-10 dataset under the IID distribution, the FMNIST dataset under the IID distribution, and the FEMNIST dataset. Although FedRT's final testing accuracy is marginally lower than the best baselines for CIFAR-10 under the non-IID distribution and FMNIST under the non-IID distribution, the differences are minimal—only 0.14% and 0.06% below the highest accuracies, respectively. These findings validate the effectiveness of the consensus distance constraint in **P2**. Additionally, FedRT demonstrates superior performance in terms of energy efficiency, yielding the lowest energy consumption for the FMNIST dataset and nearly the lowest for the CIFAR-10 and FEMNIST datasets. In summary, FedRT achieves the best trade-off between the total training latency and testing accuracy while satisfying the energy constraint.

Effect of Inter-cluster non-IID Data. We evaluate FedRT in combating the inter-cluster non-IID data distribution. The number of labels within each cluster is $2, 4, 6, 8$ both for CIFAR-10 and FMNSIT. With more labels within the cluster, the data heterogeneity tends to be IID distributed. In Fig. 10, we show the convergence rate of FedRT and the baselines without topology optimization (Static-T and CE-FedAvg) w.r.t. to global rounds. We can find the testing accuracy decreased as data heterogeneity increased. Compared with Static-T/CE-FedAvg, FedRT-RT has a best convergence speed.

Effect of Base Graph. We study the impact of base graph topology on the performance of FedRT. We generate the random graph as the base graph through Erdős–Rényi method [46]. The probability for edge creation is set to be $0.2, 0.4, 0.6, 0.8, 1.0$. With a higher probability for edge creation, the graph topology is more densely connected. The probability is 1.0, which means the generated graph is a fully connected graph. We illustrate the convergence rate in terms of global rounds for CIFAR-10 and FEMNIST. The data distribution for CIFAR-10 is IID. From Fig. 11, FedRT achieves a fast convergence rate under the IID data distribution for CIAFR-10. For FEMNIST, a loss connection leads to a slightly slow convergence rate. However, in most cases, FedRT yields a similar curve. This result demonstrates that FedRT works well under sparse base graph topology.

VIII. Conclusion

This paper introduces FedRT, a method designed to minimize training latency within a specified energy budget for a two-tier HFL system. The experimental results show that FedRT effectively reduces training latency while maintaining high accuracy, outperforming conventional methods. However, the scope of our experiments was limited to small-scale models, datasets, and a limited number of edge devices. In future work, we plan to extend these experiments to larger models, such as large language models, and more complex datasets, while also expanding the number of edge devices. Additionally, we aim to explore optimal client selection strategies to further enhance the efficiency of the HFL system.

References

[1] W. Wu, X. Deng, P. Jiang, S. Wan, and Y. Guo, "Crossfuser: Multi-modal feature fusion for end-to-end autonomous driving under unseen weather conditions," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[2] J. Xiong, E.-L. Hsiang, Z. He, T. Zhan, and S.-T. Wu, "Augmented reality and virtual reality displays: emerging technologies and future perspectives," *Light: Science & Applications*, vol. 10, no. 1, pp. 1–30, 2021.

[3] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 359–376.

[4] T. Wang, K. Zhang, J. Cai, Y. Gong, K.-K. R. Choo, and Y. Guo, "Analyzing the impact of personalization on fairness in federated learning for healthcare," *Journal of Healthcare Informatics Research*, vol. 8, no. 2, pp. 181–205, 2024.

[5] J. Cai, Z. Gao, Y. Guo, B. Wibranek, and S. Li, "Fedhip: Federated learning for privacy-preserving human intention prediction in human-robot collaborative assembly tasks," *Advanced Engineering Informatics*, vol. 60, p. 102411, 2024.

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

[7] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning for fast convergence on non-iid data," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1898–1903.

[8] Z. Zhang, Z. Gao, Y. Guo, and Y. Gong, "Scalable and low-latency federated learning with cooperative mobile edge networking," *IEEE Transactions on Mobile Computing*, 2022.

[9] R. Saha, S. Misra, and P. K. Deb, "Fogfl: Fog-assisted federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8456–8463, 2020.

[10] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[11] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Demystifying why local aggregation helps: Convergence analysis of hierarchical sgd," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8548–8556.

[12] T. Castiglia, A. Das, and S. Patterson, "Multi-level local sgd: Distributed sgd for heterogeneous hierarchical networks," in *International Conference on Learning Representations*, 2020.

[13] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[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*, vol. 82, no. 4, pp. 1558–1586, 2022.

[15] 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.

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

[17] Z. Jiang, Y. Xu, H. Xu, Z. Wang, and C. Qian, "Adaptive control of client selection and gradient compression for efficient federated learning," *arXiv preprint arXiv:2212.09483*, 2022.

[18] S. Wang, J. Perazzone, M. Ji, and K. S. Chan, "Federated learning with flexible control," *arXiv preprint arXiv:2212.08496*, 2022.

[19] P. Li, G. Cheng, X. Huang, J. Kang, R. Yu, Y. Wu, and M. Pan, "Anycostfl: Efficient on-demand federated learning over heterogeneous edge devices," *arXiv preprint arXiv:2301.03062*, 2023.

[20] T. Wu, Y. Qu, C. Liu, Y. Jing, F. Wu, H. Dai, C. Dong, and J. Cao, "Joint edge aggregation and association for cost-efficient multi-cell federated learning," in *IEEE International Conference on Computer Communications*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1–16.

[21] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[22] S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," in *Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[23] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *IEEE INFOCOM 2022-IEEE conference on computer communications*. IEEE, 2022, pp. 1739–1748.

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

[25] S. Wang, J. Perazzone, M. Ji, and K. Chan, "Federated learning with flexible control," in *IEEE Conference on Computer Communications*, 2023.

[26] J. Zhang, L. Chen, Y. Chen, X. Chen, and G. Wei, "Hierarchically federated learning in wireless networks: D2d consensus and inter-cell aggregation," *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[27] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2687–2700, 2021.

[28] W. Wen, Z. Chen, H. H. Yang, W. Xia, and T. Q. Quek, "Joint scheduling and resource allocation for hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 21, no. 8, pp. 5857–5872, 2022.

[29] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "Hfel: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6535–6548, 2020.

[30] X. Zhou, W. Liang, J. She, Z. Yan, I. Kevin, and K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5308–5317, 2021.

[31] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 1387–1395.

[32] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, “Cost-effective federated learning in mobile edge networks,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3606–3621, 2021.

[33] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, “Communication-efficient device scheduling for federated learning using stochastic optimization,” in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1449–1458.

[34] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[35] T. D. Burd and R. W. Brodersen, “Processor design for portable systems,” *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, no. 2-3, pp. 203–221, 1996.

[36] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, “Adaptive configuration for heterogeneous participants in decentralized federated learning,” in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[37] 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.

[38] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. Stich, “Consensus control for decentralized deep learning,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 5686–5696.

[39] Z. Qu, R. Duan, L. Chen, J. Xu, Z. Lu, and Y. Liu, “Context-aware online client selection for hierarchical federated learning,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4353–4367, 2022.

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

[41] 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.

[42] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[43] T.-M. H. Hsu, H. Qi, and M. Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” *arXiv preprint arXiv:1909.06335*, 2019.

[44] T. Castiglia, A. Das, and S. Patterson, “Multi-level local {sgd}: Distributed {sgd} for heterogeneous hierarchical networks,” in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=C70cp4Cn32

[45] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy efficient federated learning over wireless communication networks,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2020.

[46] P. ERDdS and A. R&wi, “On random graphs i,” *Publ. math. debrecen*, vol. 6, no. 290-297, p. 18, 1959.