Joint Participant and Learning Topology Selection for Federated Learning in Edge Clouds

Xinliang Wei, *Member, IEEE*, Kejiang Ye, *Senior Member, IEEE*, Xinghua Shi, *Member, IEEE*, Cheng-Zhong Xu, *Fellow, IEEE*, and Yu Wang, *Fellow, IEEE*

***Abstract*—Deploying federated learning (FL) in edge clouds poses challenges, especially when multiple models are concurrently trained in resource-constrained edge environments. Existing research on federated edge learning has predominantly focused on client selection for training a single FL model, typically with a fixed learning topology. Preliminary experiments indicate that FL models with adaptable topologies exhibit lower learning costs compared to those with fixed topologies. This paper delves into the intricacies of jointly selecting participants and learning topologies for multiple FL models simultaneously trained in the edge cloud. The problem is formulated as an integer non-linear programming problem, aiming to minimize total learning costs associated with all FL models while adhering to edge resource constraints. To tackle this challenging optimization problem, we introduce a two-stage algorithm that decouples the original problem into two sub-problems and iteratively addresses them separately with efficient heuristics. Our method enhances resource competition and load balancing in edge clouds by allowing FL models to choose participants and learning topologies independently. Extensive experiments conducted with real-world networks and FL datasets affirm the better performance of our algorithm, demonstrating lower average total costs with up to 33.5% and 39.6% compared to previous methods designed for multi-model FL.**

***Index Terms*—Edge computing, federated learning, learning topology, participant selection.**

I. Introduction

FEDERATED Learning (FL) [1], [2], [3], [4], [5], [6] stands out as an efficient method for enhancing machine learning (ML) performance and addressing privacy concerns for data owners. It facilitates collaboration among multiple devices to train a shared global ML model by aggregating local models. This approach ensures that training data remains on individual devices to safeguard user privacy, transmitting only essential model data, such as gradients. As smart sensing, mobile computing, and wireless networking grow, there is a shift towards deploying FL frameworks on edge clouds, catering to the agile services required by mobile devices and users. This is particularly crucial for applications like mobile AI, Artificial Intelligence of Things (AIoT), or Augmented Reality (AR) / Extended Reality (XR) [7], [8], [9], [10], [11], [12].

Three types of FL frameworks have emerged based on the learning topology used for model aggregation: *centralized FL* (CFL), *hierarchical FL* (HFL), and *decentralized FL* (DFL). CFL, the classical FL [9], employs a star architecture with a parameter server (PS) dispatching the global model to workers for collaborative training, as shown in Fig. 1(a). However, CFL faces potential communication congestion and a single point of failure at the PS. DFL [10], [13], on the other hand, eliminates the centralized PS, with workers communicating only with their trusted neighbors, as seen in Fig. 1(b). While such distributed P2P learning topology increases the robustness of FL, it might suffer from high communication overhead and lack of management [14]. HFL [15], [16], [17], [18], [19] introduces middle-layer PSs in a hierarchical topology, effectively hiding local updates within groups and enhancing privacy protection, as shown in Fig. 1(c). Each topology has its advantages, making the choice crucial for FL at the edge [10], [11], [20], [21], yet research in this area is limited, often focusing on specific learning topologies.

In participant selection for FL, various strategies have been explored, particularly for CFL [21], [22], [23], [24], [25], [26], [27], [28], [29]. This includes considering client sampling based on training data distribution, diversity, or local update importance. Edge-based FL has addressed client selection, aiming to complete FL training under time constraints. For example, Chen et al. [24] considered client sampling in their FL solution where the policy of selecting a client is based on its training data distribution/diversity or the importance of its local updates. Nishio and Yonetani [21] studied the client selection problem in an edge-based FL where the edge server in a cellular network acts as the PS and a set of mobile clients are selected as workers, to select as many mobile clients to complete the FL training under the time constraints. Deng et al. [17] have formulated an optimization problem in a cluster-based HFL to minimize communication costs incurred by edge or cloud aggregation. Wei et al. [27], [28], [29], [30], [31] studied a participant selection problem in either HFL or CFL aiming to minimize the total learning cost in an edge cloud. While existing works have

Manuscript received 11 December 2023; revised 26 May 2024; accepted 4 June 2024. Date of publication 13 June 2024; date of current version 24 June 2024. The work of Yu Wang was supported by the US National Science Foundation under Grant CNS-2006604. Recommended for acceptance by S. Wang. *(Corresponding authors: Xinliang Wei; Kejiang Ye; Yu Wang.)*

Xinliang Wei and Kejiang Ye are with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China (e-mail: xl.wei@siat.ac.cn; kj.ye@siat.ac.cn).

Xinghua Shi and Yu Wang are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19112 USA (e-mail: mindyshi@temple.edu; wangyu@temple.edu).

Cheng-Zhong Xu is with the State Key Laboratory of IoTSC, Faculty of Science and Technology, University of Macau, Macau 999078, China (e-mail: czxu@um.edu.mo).

Digital Object Identifier 10.1109/TPDS.2024.3413751

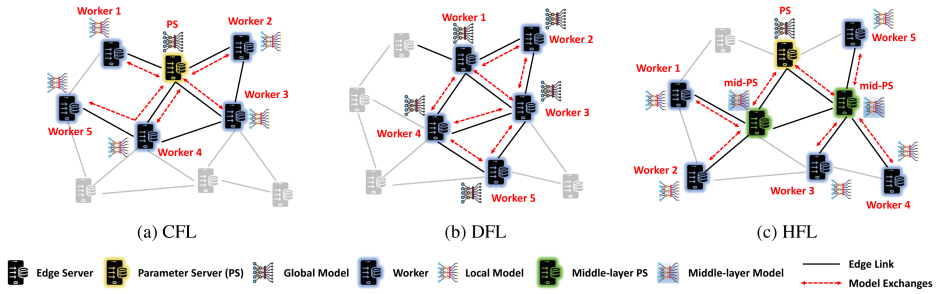

Fig. 1. Examples of different learning topologies with 5 workers of an FL model in an edge network.

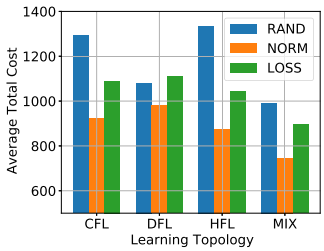

Fig. 2. Performace with different learning topologies and participants.

primarily focused on a single ML model with a specific learning topology, this paper delves into a joint participant and learning topology selection problem in multi-model federated learning over edge clouds.

The challenges in multi-model FL over edge clouds include concurrent training of multiple models, diverse edge device resources, and shared networking/computing resources. Specifically, **first**, in an edge cloud, concurrent training of multiple FL models occurs, utilizing edge servers storing diverse data types for FL model training. The heterogeneity of edge device resources leads to varying learning costs for different participant selections, comprising communication and computation costs. Shared networking/computing resources create competition among models, with the participant selected for one model influencing others and impacting the total learning cost. For instance, Fig. 2 illustrates costs for participant selection strategies (i.e., random (RAND), higher gradient norm (NORM) [22], or loss value (LOSS) [23]) during multi-model FL. Notably, NORM and LOSS consistently outperform random selection. Existing FL works on edge clouds often focus on optimizing a single global FL model, neglecting resource competition among different models. **Second**, different learning topologies (CFL, DFL, and HFL) result in diverse learning costs and performances. Fig. 2 depicts multi-model FL performance over various topologies and a mixed strategy (MIX), confirming MIX's superiority over CFL, DFL, and HFL. Previous works typically consider a specific learning topology, limiting FL training in terms of computation and communication cost, especially in resource-limited edge clouds. Both participant selection and learning topology decisions impact the total learning cost in a shared edge cloud. Motivated by this, we propose a joint optimization of participants and learning topology selection for multi-model FL over edge clouds. This involves addressing the complex optimization problem with edge resource constraints, formulated as an integer non-linear programming problem, to minimize the overall communication and computation cost.

The major contributions of this work include:

- Formulating a joint participant and learning topology selection problem as an integer non-linear programming problem, aiming to minimize the total cost of all models. This approach allows different FL models to choose their participants and learning topology, addressing resource competition and load balancing in edge clouds.
- Decoupling the problem into two sub-problems: participant selection and learning topology determination, instead of directly solving the challenging joint optimization. An effective two-stage method is proposed to iteratively solve these sub-problems, selecting appropriate participants and determining the learning topology for each model.
- Conducting extensive evaluations with real-world datasets, including a real-world network topology (EUA [32]), three classical datasets [33] for the linear regression (LR) model (convex), and three classical image datasets (CIFAR10 [34], Fashion-MNIST (FMNIST) [35], MNIST [36]) for convolutional neural network (CNN) model (non-convex), comparing the proposed method with existing ones. Results show better performance in various settings, confirming the effectiveness of jointly considering both participant selection and learning topology selection in a multi-model FL environment.

The remainder of this paper is organized as follows. In Section II, we introduce our edge cloud model and FL processes at the edge. Section III formulates the joint optimization problem, followed by our proposed two-stage method in Section IV. We provide evaluations of the proposed method in Section V. Section VI reviews related work, and Section VII concludes the paper with possible future directions.

II. Preliminaries

A. Edge Cloud Model

We consider a typical edge cloud as a graph $G(V, E)$ consisting of N edge servers and L edge links, as shown in Fig. 1. Each edge server $v_i \in V$ has a specific storage capacity c_i, CPU frequency f_i and each edge link $e_l \in E$ has a limited bandwidth lb_l. Here, we only consider CPU computation for simplicity,

TABLE I
SUMMARY OF NOTATIONS

Symbol	Notation
V, E	set of edge servers and edge links
N, L, H	number of edge servers, links, & FL models
v_i, e_l	ith edge server, lth edge link
c_i, f_i	storage capacity and CPU frequency of v_i
lb_l	link bandwidth of link e_l
$D_{i,j}$	the local dataset of m_j at server v_i
m_j, κ_j	jth FL model, its required worker number
ψ_j	required number of middle-layer PSs in HFL
μ_j, χ_j	model size, CPU requirement of m_j
$\omega_{i,j}$	local model parameter of m_j at worker v_i
$\hat{\alpha}, \hat{\beta}$	number of global aggregation, local updates
α, β	index of global aggregation, local updates
$\hat{\gamma}$	number of middle-layer aggregation in HFL
γ	index of middle-layer aggregation in HFL
θ, ϵ	local, global convergence accuracy
η	learning rate of the loss function
$a_{i,j}$, $b_{j,k}$	participant & learning topology decision
$x_{i,j}$, $y_{i,j}$ $z_{i,j}$	participant indicator decision
$P_{i,k}$	shortest path from edge server v_i to v_k
C^{local}, C^{comm}	local-update/communication costs of models

but GPU or hybrid computation can be considered in a similar fashion. We assume that mobile users trust the edge cloud and upload their dataset to the edge server [9]. Moreover, various security techniques (e.g., encryption algorithms and differential privacy mechanisms) can be applied to prevent privacy leakage during data transmission[1]. Each edge server can hold multiple types and numbers of datasets from mobile users, such as images, texts, and voices. We use $D_{i,j}$ to denote the local dataset at server i for training model j. Table I summarizes the notations we used in this paper.

B. Federated Learning at Edge

We assume that H FL models (denoted by $M = \{m_j\}$) are being trained at the edge cloud simultaneously. Each model can adapt a distinct learning topology (one of CFL, DFL, and HFL) to perform the FL process. To train each model m_j, it requests a number of participants (κ_j workers and one or a few PSs if needed) who have sufficient resources (i.e., storage and CPU frequency should be larger than the minimal storage μ_j and CPU frequency χ_j of m_j). In addition, we assume each edge server can only perform FL training of one model at one time. The training process of FL for each model m_j, includes three parts: (a) initializing and broadcasting the global model of m_j to each participant; (b) each worker performs the local model computation using its own dataset; (c) aggregating the local models from workers (and middle-layer PSs in HFL).

1) *Initializing the Global Model:* We initialize the global model parameter for each FL model as ω_j and send the global model parameter to each selected participant.

2) *Local Training on Workers:* Let the local model parameters of model m_j on the server v_i be $\omega_{i,j}$ and the loss function on a training data sample s be $f_i(\omega_{i,j}, \mathbf{x}_s, \mathbf{y}_s)$, where $\mathbf{x}_s$ is the input data and $\mathbf{y}_s$ is the required output value. Then the loss function on the whole local dataset of v_i is defined as

$$F_i(\omega_{i,j}) = \frac{1}{|D_{i,j}|} \sum_{s \in D_{i,j}} f_i(\omega_{i,j}, \mathbf{x}_s, \mathbf{y}_s). \tag{1}$$

Generally, FL will perform round by round and we denote the total number of global aggregation, local updates, and middle-layer aggregation as $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$, respectively. In the α-th round, each worker has to run a number of local updates to achieve a *local convergence accuracy* $\theta \in (0, 1)$. At β-th local iteration, each worker follows the same local update rule as

$$\omega_{i,j}^{\alpha,\beta} = \omega_{i,j}^{\alpha,\beta-1} - \eta \nabla F_i(\omega_{i,j}^{\alpha,\beta-1}), \tag{2}$$

where η is the learning rate of the loss function. This process will run until

$$||\nabla F_i(\omega_{i,j}^{\alpha,\beta})|| \leq \theta ||\nabla F_i(\omega_{i,j}^{\alpha,\beta-1})||. \tag{3}$$

Here, we set $\omega_{i,j}^{\alpha,0} = \omega_j$.

3) *Aggregating the Local Model:* For different learning topologies, e.g. CFL, DFL, and HFL, the aggregation process is distinct even though the aggregation method can be the same, such as FedAvg [1].

If a model adopts CFL, one participant has to be chosen as the PS. After $\hat{\beta}$ local training, all workers send their local model parameter $\omega_{i,j}^{\alpha,\hat{\beta}}$ to the PS. The PS performs FedAvg to aggregate the global model parameters as

$$\omega_j^{\alpha} = \sum_{i \in S_j} \frac{D_{i,j}}{D_j^{\alpha}} \omega_{i,j}^{\alpha-1,\hat{\beta}}, \tag{4}$$

where $D_j^{\alpha} = \bigcup_{i \in S_j} D_{i,j}$ is the total number of data sample from κ_j workers and S_j is the set of selected workers.

If DFL is adopted, every selected participant are worker who exchanges and aggregates their local model parameter $\omega_{i,j}^{\alpha,\hat{\beta}}$ with their neighbors in the learning topology after $\hat{\beta}$ local training. The aggregation process at participant v_i is

$$\omega_{i,j}^{\alpha} = \frac{D_{i,j}}{D_j^{\alpha}} \omega_{i,j}^{\alpha-1,\hat{\beta}} + \sum_{k \in S_{i,j}} \frac{D_{k,j}}{D_j^{\alpha}} \omega_{k,j}^{\alpha-1,\hat{\beta}}, \tag{5}$$

where $D_j^{\alpha} = \bigcup_{i \in S_j, k \in S_{i,j}} D_{k,j}$ is the total number of data samples from the worker and its neighbors and $\omega_{k,j}^{\alpha,\hat{\beta}}$ is the model parameters from the set of neighbors $S_{i,j}$ at α-th global round after $\hat{\beta}$ local updates.

HFL is similar to CFL but adds several middle-layer PSs (group leaders) between the PS (at the top layer) and workers. After $\hat{\beta}$ local training, workers send their local model parameter $\omega_{i,j}^{\alpha,\hat{\beta},\gamma}$ to the middle-layer PSs and then the middle-layer PSs will aggregate the local model using averaging algorithms. After $\hat{\gamma}$ middle-layer aggregations, the middle-layer PSs will send their group model parameters $\omega_{i,j}^{\alpha,\hat{\gamma}}$ to the top-layer PS for the final aggregation. Assuming that S_j is the set of all selected middle-layer PSs and $S_{i,j}$ is the set of all selected workers connected to the middle-layer parameter server v_i. Then the

[1] In addition to data privacy, there are also significant developments on the protection of model privacy in FL recently [37], [38], these works are orthogonal to our work but can be integrated with our proposed framework. We leave such study as one of our future works.

middle-layer aggregation at v_i is as follows.

$$\omega_{i,j}^{\alpha,\gamma} = \sum_{k \in S_{i,j}} \frac{D_{k,j}}{D_{i,j}^{\gamma}} \omega_{k,j}^{\alpha,\hat{\beta},\gamma-1}, \tag{6}$$

where $D_{i,j}^{\gamma} = \bigcup_{k \in S_{i,j}} D_{k,j}$ is the total number of data samples from the workers connected to the middle-layer PS v_i. The averaging aggregation at the top-layer PS is similar as follows.

$$\omega_j^{\alpha} = \sum_{i \in S_j} \frac{D_{i,j}^{\gamma}}{D_j^{\alpha}} \omega_{i,j}^{\alpha-1,\hat{\gamma}}, \tag{7}$$

where $D_j^{\alpha} = \bigcup_{i \in S_j} D_{i,j}^{\gamma}$ is the total number of data samples from the set of all middle-layer PSs.

C. Cost Model

Our cost model mainly consists of two parts: *computation cost* and *communication cost*.

1) *Computation Cost:* Since it has been shown that model aggregation is a much easier task compared with model training, we mainly consider the cost of local training. Let Ψ_i be the number of CPU cycles to process one sample data at edge server v_i and $|\cdot|$ is the size of dataset $D_{i,j}$ for jth model at server v_i. Then, the total $\hat{\beta}$ local training at v_i is defined as

$$C_{i,j}^{local} = \hat{\beta} \cdot \frac{\Psi_i |D_{i,j}|}{f_i}. \tag{8}$$

2) *Communication Cost:* The communication cost mainly consists of the transmission cost between two edge servers. Let μ_j be the size of jth model m_j and $P_{i,k}$ be the shortest path between server v_i and v_k. Then, the communication cost [5] between two servers for model m_j is defined by

$$C_{i,k}^{comm} = \sum_{e_l \in P_{i,k}} \frac{\mu_j}{lb_l}. \tag{9}$$

III. Problem Formulation

We now introduce the formulation of the joint optimization, and the theoretical bounds on FL convergence with different learning topologies.

A. Problem Formulation

Before formulating our joint participant and learning topology selection problem, we first introduce the decision variables and optimization goal.

1) Decision Variables: We use $a_{i,j} \in \{0,1\}$ to denote whether edge server v_i is chosen as a participant (including both workers and PS) of model m_j. We use $b_{j,k} \in \{0,1\}$ to denote whether selecting the kth learning topology for model m_j, here $k \in \{1,2,3\}$ represents CFL, DFL, and HFL topology. If CFL is selected (i.e. $b_{j,1} = 1$), we use $x_{i,j} \in \{0,1\}$ to indicate whether v_i is selected as the PS for model m_j. Here, the selected PS v_i must be in the selected participants of m_j, i.e., both $a_{i,j}$ and $x_{i,j} = 1$. If DFL is selected (i.e., $b_{j,2} = 1$), we do not need any PS. Here we assume that DFL uses a simple nearest neighbor method to determine the connection among κ_j workers. e.g., we let each worker connect to a fixed number of closest neighbors in the edge cloud. Then the link between workers v_i and v_l in the learning topology can be defined by a binary indicator $\iota_{i,l}$. This information will be used in our optimization. Note that the study of learning topology formation is orthogonal to our work, but other topology formation algorithms can be used in this proposed framework. If HFL is selected (i.e. $b_{j,3} = 1$), we need to pick one top-layer PS and ψ_j middle-layer PSs. We denote $y_{i,j} \in \{0,1\}$ and $z_{i,j} \in \{0,1\}$ as the indicators of top-layer and middle-layer PSs. Similarly, these selected PSs must be among selected participants. The connection between workers and their middle-layer PS is decided by network distance. i.e., each worker selects the nearest middle-layer PS as its aggregator. We use $\xi_{i,l}$ to denote the connection between worker v_i and its middle-layer PS v_l. In summary, $a_{i,j}$ and $b_{j,k}$ are the decisions on participant selection and learning topology determination, while $x_{i,j}$, $y_{i,j}$ and $z_{i,j}$ are the roles of participants in their selected learning topology. The relationship among $a_{i,j}$, $b_{j,k}$, $x_{i,j}$, $y_{i,j}$ and $z_{i,j}$ are enforced via constraints.

2) Optimization Goal: Our problem aims to minimize the total learning cost of all models, enabling the training of more FL models and enhancing the sustainability and cost-effectiveness of FL training. We assume that each edge server will perform $\hat{\beta}$ round of local training and each model will perform $\hat{\alpha}$ global aggregations. If HFL is chosen, it will perform $\hat{\alpha}$ aggregations at top-layer PS and $\hat{\gamma}$ middle-layer aggregations at each middle-layer PS. Please note that the choice of $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ will be discussed in Section III-B. Therefore, the total computation cost of all models is defined as follows,

$$C^{comp} = \sum_{j=1}^{H} \sum_{i=1}^{N} (\hat{\alpha} \cdot C_{i,j}^{local} \cdot ((a_{i,j} - x_{i,j}) \cdot b_{j,1} + a_{i,j} \cdot b_{j,2} + \hat{\gamma} \cdot (a_{i,j} - y_{i,j} - z_{i,j}) \cdot b_{j,3})),$$

and the communication cost of all models can be written as

$$C^{comm} = \sum_{j=1}^{H} \sum_{i=1}^{N} \sum_{l=1}^{N} (\hat{\alpha} \cdot C_{i,l}^{comm} \cdot (a_{i,j} \cdot \iota_{i,l} \cdot b_{j,2} + (a_{i,j} - x_{i,j}) \cdot x_{l,j} \cdot b_{j,1} + (\hat{\gamma} \cdot (a_{i,j} - y_{i,j} - z_{i,j}) \cdot \xi_{i,l} + y_{i,j} \cdot z_{l,j}) \cdot b_{j,3})).$$

Then the total learning cost of all models is simply the summation of computation and communication costs.

3) Joint Optimization: Our joint problem aims to select the optimal participants and learning topology so that the total learning cost of all models is minimized while satisfying the lowest FL performance for each model, formulated as

$$\min \quad C^{comp} + C^{comm} \tag{10}$$

$$\text{s.t.} \quad a_{i,j}\mu_j \le c_i, \qquad \forall i,j, \tag{11}$$

$$a_{i,j}\chi_j \le f_i, \qquad \forall i,j, \tag{12}$$

$$\sum_i a_{i,j} = \kappa_j + b_{j,1} + (1+\psi_j) b_{j,3}, \qquad \forall j, \tag{13}$$

$$\sum_j a_{i,j} \leq 1, \quad \forall i, \quad (14)$$

$$\sum_i a_{i,j} x_{i,j} \cdot \sum_i x_{i,j} \cdot b_{j,1} - b_{j,1} = 0, \quad \forall j, \quad (15)$$

$$\sum_i a_{i,j} y_{i,j} \cdot \sum_i y_{i,j} \cdot b_{j,3} - b_{j,3} = 0, \quad \forall j, \quad (16)$$

$$\sum_i a_{i,j} z_{i,j} \cdot \sum_i z_{i,j} \cdot b_{j,3} - \psi_j^2 b_{j,3} = 0, \quad \forall j, \quad (17)$$

$$\sum_j (y_{i,j} + z_{i,j}) b_{j,3} \leq 1, \quad \forall i, \quad (18)$$

$$\sum_k b_{j,k} = 1, \quad \forall j, \quad (19)$$

$$a_{i,j} \in \{0,1\}, b_{j,k} \in \{0,1\}, \quad (20)$$

$$x_{i,j} \in \{0,1\}, y_{i,j} \in \{0,1\}, z_{i,j}, \in \{0,1\}, \quad (21)$$

$$i \in [1,..,N], j \in [1,..,H], k \in [1,..,3]. \quad (22)$$

Here, Constraint (11) and (12) ensures that the storage and CPU frequency of edge servers satisfy the FL model requirements. Constraint (13) guarantees the number of participants of each model is κ_j for DFL, $\kappa_j + 1$ for CFL, and $\kappa_j + 1 + \psi_j$ for HFL. Constraint (14) makes sure that each participant only works for one model. Constraints (15)–(17) enforce that (i) for each model the specific number of PS is 1 for both CFL and HFL and the number of middle-layer PSs is ψ_j for HFL, and (ii) the selected PSs for each model are indeed within the selected participants for this model (such as, if $x_{i,j} = 1$ then $a_{i,j}$ must be 1). For example, in Constraint (15) for CFL, the term of $\sum_i x_{i,j}$ makes sure only one PS is selected for model m_j, while the term of $\sum_i a_{i,j} x_{i,j}$ combined with Constraint (14) makes sure that there exists and only exists a server v_i for model j that both $a_{i,j} = 1$ and $x_{i,j} = 1$. Meanwhile, Constraint (18) ensures that the PS and middle-PS of HFL are different. Furthermore, Constraint (19) guarantees that each FL model must select one learning topology. Decision variables and their ranges are given by Constraints (20)–(22).

B. Theoretical Bounds on Convergence

To solve the optimization problem, we must examine how the choice of participants and learning topology impacts the FL training in terms of the number of local training and global updates for a single model. Since the update of the global model depends on the selected learning topology, we can only analyze the expected convergence rate of FL. We first make the following assumptions, as done in previous works [10], [13], [39], [40], [41], [42], [43], [44].

Assumption 1. (Smoothness and Strongly Convex): All loss functions $F_i(\cdot)$ are ζ-Smoothness and ε-strongly convex, where $0 < \varepsilon \leq \zeta$. Thus, for all i, j, we have the following relationships.

$$\varepsilon ||\omega_{i,j}^{\alpha,\beta+1} - \omega_{i,j}^{\alpha,\beta}|| \leq ||\nabla F_i(\omega_{i,j}^{\alpha,\beta+1}) - \nabla F_i(\omega_{i,j}^{\alpha,\beta})||;$$
$$||\nabla F_i(\omega_{i,j}^{\alpha,\beta+1}) - \nabla F_i(\omega_{i,j}^{\alpha,\beta})|| \leq \zeta ||\omega_{i,j}^{\alpha,\beta+1} - \omega_{i,j}^{\alpha,\beta}||.$$

Given the local training process (1) to (3), the convergence rate of local model training satisfies as follows.

$$F_i(\omega_{i,j}^{\alpha,\hat{\beta}}) - F_i(\omega_{i,j}^{\alpha,*}) \leq \theta^{\hat{\beta}} (F_i(\omega_{i,j}^{\alpha,0}) - F_i(\omega_{i,j}^{\alpha,*})), \quad (23)$$

where $\omega_{i,j}^{\alpha,*}$ is the optimal local model at worker v_i for model m_j at α-th global iteration.

Lemma 1: Analogous to previous works [9], [20], [43], with Assumption 1 and the local convergence rate (23), the number of local training to achieve a *θ-local convergence accuracy* is

$$\hat{\beta} \geq \frac{2}{(2 - \zeta\eta)\eta\varepsilon} log \frac{1}{\theta}, \quad (24)$$

where η is the learning rate of the local loss function.

Assumption 2. (Bounded Gradient): For any worker v_i, the stochastic gradients are uniformly bounded by

$$\mathbb{E}_{s \sim D_{i,j}} ||\nabla f_i(\omega_j^\alpha, \mathbf{x}_s, \mathbf{y}_s)||^2 \leq \varsigma^2, \forall i, j.$$

Assumption 3. (Bounded Variance): The stochastic gradient variance is bounded for any worker v_i, i.e. there exists a constant σ_i, such that

$$\mathbb{E}_{s \sim D_{i,j}} ||\nabla f_i(\omega_j^\alpha, \mathbf{x}_s, \mathbf{y}_s) - \nabla F_i(\omega_j^\alpha)||^2 \leq \sigma_i^2, \forall i, j.$$

Here s is the data sample from dataset $D_{i,j}$ at v_i for m_j.

Assumption 4. (Data Heterogeneity): For any worker v_i, the local gradient $\nabla F_i(\omega_j^\alpha)$ and the global gradient $\nabla F(\omega_j^\alpha)$ are uniformly bounded by

$$\mathbb{E}||\nabla F_i(\omega_j^\alpha) - \nabla F(\omega_j^\alpha)||^2 \leq \nu^2, \forall i, \forall j.$$

Note that if data samples among workers are *i.i.d* setting, then $\nu = 0$; otherwise, ν is a positive constant.

Definition 1. (Weight Matrix): We can represent the learning topology as an undirected graph, then we denote $\mathbf{W} \in \mathbb{R}^{N \times N}$ as a symmetric doubly stochastic matrix, which can be computed via $\mathbf{W} = \mathbf{I} - \frac{\mathbf{L}}{\lambda_{\max}(\mathbf{L})}$, where $\mathbf{L}$=$\mathbf{D}$-$\mathbf{A}$ is Laplacian matrix. Here $\mathbf{D}$ is the degree matrix of the adjacent matrix $\mathbf{A}$ of the learning topology. $\mathbf{W}_{ij}$ encodes whether workers v_i connects v_j in the topology.

Weight matrix $\mathbf{W}$ has the following properties: (1) $\mathbf{W1} = \mathbf{1}, \mathbf{1}^T\mathbf{W} = \mathbf{1}^T, \mathbf{W} = \mathbf{W}^T$; (2) The n-th eigenvalue $\lambda_n(\mathbf{W})$ satisfies $-1 < \lambda_n(\mathbf{W}) \leq \cdots \leq \lambda_1(\mathbf{W}) = 1$.

With all these, the following theorem for the convergence rate of different learning topologies under a single model can be proved using similar proofs from [13], [39], [44].

Theorem 1: Given the participant selection matrix $\mathbf{a}$, learning topology selection matrix $\mathbf{b}$, a set of parameter server selection matrix $\mathbf{x}, \mathbf{y}, \mathbf{z}$, optimal global model $\mathbf{w}_j^*$, and the following variables: $\Pi = \max(\frac{8\zeta}{\varepsilon}, \hat{\beta}), \rho = (\max\{|\lambda_2(\mathbf{W})|, |\lambda_n(\mathbf{W})|\})^2, \sigma = \frac{1}{\kappa_j} \sum_{i=1}^N (a_{i,j} - x_{i,j} - y_{i,j} - z_{i,j}) \sigma_i$, then the convergence rate of $\mathbb{E}[F(w_j^{\hat{\alpha}}) - F(w_j^*)]$ can be given by

$$\mathbb{E}[F(w_j^{\hat{\alpha}}) - F(w_j^*)] \leq b_{j,1}\mathcal{A} + b_{j,2}\mathcal{B} + b_{j,3}\mathcal{C}, \forall j,$$

where

$$\mathcal{A} = \mathcal{O}\left(\frac{\sigma^2}{\varepsilon\hat{\alpha}} + \frac{\zeta\nu}{\varepsilon\hat{\alpha}} + \frac{\hat{\beta}^2\varsigma^2(\kappa_j + 1)}{\kappa_j\varepsilon\hat{\alpha}} + \frac{\Pi\varsigma^2}{\varepsilon\hat{\alpha}}\right),$$

$$\mathcal{B} = \mathcal{O}\left(\frac{\zeta}{\hat{\alpha}} + \frac{\sigma^2}{\kappa_j} + \frac{\kappa_j}{(1-\sqrt{\rho})^2 - 18\kappa_j}\left(\frac{\sigma^2(1-\sqrt{\rho})^2}{1-\rho} + \nu^2\right)\right),$$

$$\mathcal{C} = \mathcal{O}\left(\frac{1+\sigma^2}{\sqrt{\kappa_j\hat{\alpha}\hat{\beta}\hat{\gamma}}} + \frac{(\psi_j - 1)}{\hat{\alpha}(\sigma^2 + \hat{\beta}\hat{\gamma}\nu^2)^{-1}} + \frac{(\kappa_j - \psi_j)}{\hat{\alpha}\hat{\gamma}(\sigma^2 + \hat{\beta}\nu^2)^{-1}}\right).$$

Recall that the details of the proof can be found in [13], [39], [44]. For each convergence rate analysis, they are all related to the number of workers κ_j and the data heterogeneity ν. As the number of workers increases, the convergence rate of each method decreases. Also, the data heterogeneity impacts the convergence performance where *non-i.i.d* setting ($\nu > 0$) needs more time to converge to the predefined accuracy.

Corollary 1: Let ϵ be the *global convergence accuracy* and given the learning topology selection matrix $\mathbf{b}$, to achieve $F(w_j^{\hat{\alpha}}) - F(w_j^*) \leq \epsilon$, the number of global iteration $\hat{\alpha}$ can be approximated by

$$\hat{\alpha} \propto \begin{cases} \left(\left(1 + \frac{1}{\kappa_j}\right)\hat{\beta}^2\varsigma^2 + \sigma^2 + \zeta\nu + \varsigma^2\right)\log\frac{1}{\epsilon}, & \text{if } b_{j,1} = 1, \\ \frac{4\zeta^2\kappa_j^5}{\sigma^6}\left(\frac{\sigma^2}{1-\rho} + \frac{9\zeta^2}{(1-\sqrt{\rho})^2}\right)^2, & \text{if } b_{j,2} = 1, \\ \left(\frac{1+\sigma^2}{\sqrt{\kappa_j\hat{\beta}\hat{\gamma}}} + \left(\psi_j - 1 + \frac{\kappa_j - \psi_j}{\hat{\gamma}}\right)\sigma^2\right. \\ \left. + \left(\psi_j\hat{\gamma} - \hat{\gamma} + \frac{\kappa_j - \psi_j}{\hat{\gamma}}\right)\hat{\beta}\nu^2\right)\log\frac{1}{\epsilon}, & \text{if } b_{j,3} = 1. \end{cases}$$

Note that, for DFL, some works [45] leverage the condition number of communication graph to represent the topology of selected workers, so the number of global iterations can also be approximated by $\hat{\alpha} = (\varrho + (1+\nu)\frac{\zeta}{\varepsilon})\log\frac{1}{\epsilon}$, where $\varrho = \frac{\lambda_{\max}(\mathbf{I}-\mathbf{W})}{\lambda_{min}^{+}(\mathbf{I}-\mathbf{W})}$ is the condition number of the topology. For HFL, we loose the effect of the number of middle-layer aggregations and the number of middle-layer PS for simplicity. We set a fixed number for $\hat{\gamma}$ and ψ_j based on the selected number of workers, and use a fixed grouping strategy where each group has a fixed number of workers.

IV. Algorithm Design

In this section, we introduce the details of our algorithm design. We start with a brief overview of the proposed two-stage algorithm, then present the detailed methods used in two stages. Finally, we will provide the complexity analysis of the proposed algorithm.

A. Overall Algorithm

To solve the original joint optimization problem, we decompose it into two separated stages and iteratively solve these two sub-problems, as shown in Fig. 3. In Stage 1, we solve the participant selection sub-problem for each model with an initial learning topology or the learning topology determined by Stage 2 in the previous iteration. In Stage 2, we determine the learning topology for each model with the selected participants from Stage 1. We iteratively process these two stages until it reaches the maximum number of iterations or the same selection appears at the maximal occurring times. Note that the processing order of all models will affect the results. In our simulations, we simply adopt the first come first serve mode, i.e. first handling the model that arrives earlier. The overall iterative algorithm is given by Algorithm 1.

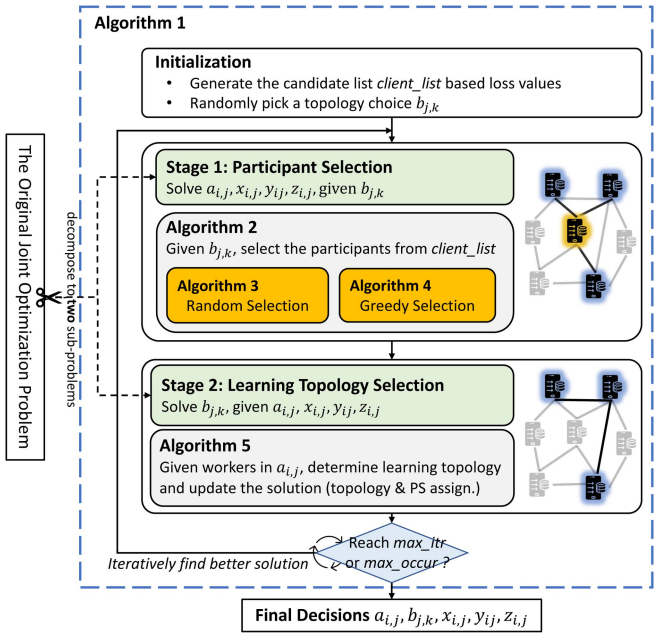

Fig. 3. The problem decomposition and the overall design of our proposed algorithms.

The selection of participants directly from all possible edge servers in the edge cloud could lead to high computation overhead. Therefore, we first generate a worker candidate list (Line 3 of Algorithm 1), denoted by $client\_list_j$ for model m_j. Here, we assume we choose ϖ_j servers into this list, and $N > \varpi_j > \kappa_j$. Later, in Stage 1, we will pick κ_j workers from this $client\_list_j$ using different heuristics by running Algorithm 2, which can be performed much more efficiently than an exhausting search over all edge servers. As we mentioned before, the client selection problem in FL has been well-studied. Some solutions prefer to select workers based on the highest gradient norm [22] or loss [23] value while others focus on the proportion of data size in each client or the minimal completion time of each client. Here, we adopt the idea of selecting edge servers with higher loss value as FL worker candidates. For each model m_j, we generate ϖ_j FL worker candidates with the top ϖ_j high loss values to form the $client\_list_j$ (Line 3 of Algorithm 1). In Line 5, a random learning topology is initialized for each model.

B. Stage 1: Participant Selection

In the first stage, given the learning topology selection $b_{j,k}$ of model m_j, we would like to select the participants for m_j (including κ_j workers and possible PSs if CFL or HFL is used). We use Algorithm 2 to select participants for each model m_j. (1) It first selects κ_j workers from the candidates set $client\_list_j$ and decides PSs if needed from the remaining edge servers (Line 2/5/9). (2) Then it determines the specific topology for DFL/HFL (Line 6/10). For DFL, we construct a topology in which each participant connects to a fixed number of closest neighbors. For

Algorithm 1: Overall Iterative Algorithm.

Input: Edge cloud with N servers V, H FL models M
Output: Participant selection $a_{i,j}$, learning topology selection $b_{j,k}$, and PS assignments $x_{i,j}$, $y_{i,j}$, $z_{i,j}$
1: Initialize remaining edge server list $V^r \leftarrow V$
2: **for** $m_j \in M$ **do**
3: Generate a candidate set $client\_list_j$ with the top ϖ_j highest loss values from V^r
4: Initialize $\max\_itr$, $\max\_occur$, $bound\_val$, and set $itr = 1$ and $count\_num = 0$
5: Randomly pick initial learning topology by setting $b_{j,k}$
6: **repeat**
7: Stage 1: run Algorithm 2 to generate the new participant selection $a_{i,j}$ and PS assignments ($x_{i,j}$, $y_{i,j}$, $z_{i,j}$) based on current topology choice $b_{j,k}$
8: Stage 2: run Algorithm 5 to determine the learning topology $b_{j,k}$ and update PS selection based on current participant selection from Stage 1. Let obj_val be the achieved learning cost
9: **if** $obj\_val \leq bound\_val$ **then**
10: $bound\_val = obj\_val$; $count\_num++$
11: Update decision variable $a^*_{i,j}$, $b^*_{j,k}$, $x^*_{i,j}$, $y^*_{i,j}$, $z^*_{i,j}$
12: $itr = itr + 1$
13: **until** $count\_num = max\_occur$ or $itr = \max\_itr$
14: $V^r \leftarrow V^r / \{v_i | a^*_{i,j} = 1\}$
15: **return** $a^*_{i,j}$, $b^*_{j,k}$, $x^*_{i,j}$, $y^*_{i,j}$, $z^*_{i,j}$

Algorithm 2: Participant Selection for Model m_j.

Input: Candidates set $client\_list_j$, learning topology $b_{j,k}$, and required number of workers κ_j of model m_j
Output: Participant selection $a_{i,j}$ and role assignments $x_{i,j}$, $y_{i,j}$, $z_{i,j}$ in learning topology
1: **if** $b_{j,1} = 1$ **then**
2: $participant\_list_j =$ RandomSel() or GreedySel()
3: Update $a_{i,j}$, $x_{i,j}$ with $participant\_list_j$
4: **else if** $b_{j,2} = 1$ **then**
5: $participant_list_j =$RandomSel() or GreedySel()
6: Determine DFL topology for $participant\_list_j$
7: Update $a_{i,j}$, $\iota_{i,l}$ with $participant\_list_j$ and the topology
8: **else if** $b_{j,3} = 1$ **then**
9: $participant\_list_j =$ RandomSel() or GreadySel()
10: Determine HFL topology for $participant\_list_j$
11: Update $a_{i,j}, y_{i,j}, z_{i,j}, \xi_{i,l}$ with $participant\_list_j$ and the topology
12: **return** $a_{i,j}$, $x_{i,j}$, $y_{i,j}$, $z_{i,j}$

HFL, after the PS decision, each worker will connect to the closest middle-layer PS. (3) Last, it generates the participant selection decision (both $a_{i,j}$ and $x_{i,j}, y_{i,j}, z_{i,j}$) and updates the related learning topology variables ($\iota_{i,l}, \xi_{i,l}$) for DFL/HFL (Line 3/7/11).

When we select a participant list $participant\_list_j$ (Line 2/3/9 in Algorithm 2), we provide two options: *random selection*

Algorithm 3: RandomSel() - Random Selection.

Input: Candidates set $client\_list_j$, learning topology $b_{j,k}$, and required number of workers κ_j of model m_j
Output: Participant list $participant\_list_j$ (including $worker\_list$ and PS lists ps, top_ps, mid_ps if needed)
1: $worker\_list =$ random($client\_list_j$, κ_j)
2: **if** $b_{j,1} = 1$ **then**
3: $ps =$ the closest one to the selected workers among all remaining edge servers outside $worker\_list$
4: **else if** $b_{j,3} = 1$ **then**
5: $mid\_ps =$ the closest ψ_j servers to the selected workers among servers outside $worker\_list$
6: $top\_ps =$ the $(\psi_j + 1)$-th closest server to the selected workers among servers outside $worker\_list$
7: **return** $participant\_list_j$ (i.e., $worker\_list$ and PS lists)

or *greedy selection*. In the random method, we randomly select κ_j workers from the candidates set $client\_list_j$ (Line 1 of Algorithm 3). After that, for CFL we select a PS outside $worker\_list$ with the lowest communication cost to all workers; for HFL, we select the top ψ_j servers outside $worker\_list$ with the least communication cost to all workers as middle-layer PS and take the $(\psi_j + 1)$th server as the top-layer PS. In the greedy method, we first pick PSs from outside $client\_list_j$ if needed (Line 2-3 Algorithm 4). Then we iteratively select an edge server that has the minimal total cost from the candidates set $client\_list_j$ until we get κ_j workers (Lines 4–7).

C. Stage 2: Learning Topology Selection

In the second stage, given the selected participants (mainly the workers $worker\_list$) from Stage 1, we aim to determine the better learning topology for them. Thus, we choose the PSs for CFL/HFL (from remaining servers outside $worker\_list$) and determine the topology for DFL/HFL, then compare their learning costs. Finally, the learning topology with the minimum cost is selected and returned. Algorithm 5 shows the detail of learning topology selection.

Recall that the study of the optimal learning topology construction for DFL and HFL is orthogonal to our work, but other topology formation algorithms can be used in this proposed framework. For instance, we can leverage graph embedding to construct an optimal topology for participants in DFL, while considering the data heterogeneity.

D. Complexity Analysis

We now provide the complexity analysis of our algorithm.

Theorem 2: The overall time complexity of our proposed algorithm is $\mathcal{O}(\sum_{j=1}^{H} Z(N \log N + \kappa_j \varpi_j))$, where N and H are the numbers of edge servers and models, κ_j and ϖ_j are the numbers of required workers and candidate workers for jth model, and Z is the maximal iteration $\max\_itr$.

Proof: In Algorithm 2, the participant selection is done in one of two ways, either random selection or greedy selection

Algorithm 4: GreedySel() - Greedy Selection.

Input: Candidates set $client\_list_j$, learning topology $b_{j,k}$, and required number of workers κ_j of model m_j
Output: Participant list $participant\_list_j$
1: $worker\_list = \varnothing$
2: $ps =$ the closest server outside $client\_list_j$ to $client\_list_j$
3: $mid\_ps, top\_ps =$ the closest $(\psi_j + 1)$ edge servers outside $client\_list_j$ to $client\_list_j$
4: **for** $i = 1$ *to* κ_j **do**
5: **for** c_i *in* $client\_list_j$ *but not in* $worker\_list$ **do**
6: Calculate $total\_cost(c_i)$ if adding c_i to $worker\_list$ based on the topology choice $b_{j,k}$
7: Let c_i^* be the one leading to minimal $total\_cost(c_i)$, then add c_i^* to $worker\_list$
8: **return** $participant\_list_j$ (i.e., $worker\_list$ and PS lists)

Algorithm 5: Learning Topology Selection for Model m_j.

Input: Worker list $worker\_list$ from $a_{i,j}, x_{i,j}, y_{i,j}, z_{i,j}$
Output: Learning topology selection $b_{j,k}$, participant selection $a_{i,j}, x_{i,j}, y_{i,j}, z_{i,j}$ and learning topology
1: $ps =$ the closest one to workers in $worker\_list$ among remaining servers; calculate cfl_cost with this topology
2: Determine the DFL topology for $worker\_list$ and calculate dfl_cost for this DFL topology
3: $mid\_ps, top\_ps =$ the closest $(\psi_j + 1)$ servers to $worker\_list$ outside $worker\_list$; determine the HFL topology for $worker\_list$ and calculate hfl_cost
4: Choose the learning topology with the minimum learning cost among cfl_cost, dfl_cost and hfl_cost; update $a_{i,j}, b_{j,k}, x_{i,j}, y_{i,j}, z_{i,j}, \xi_{i,l}, \iota_{i,l}$ with the selected topology
5: **return** $a_{i,j}, b_{j,k}, x_{i,j}, y_{i,j}, z_{i,j}$ and learning topology

(Line 2/5/7). Random selection needs $\mathcal{O}(N \log N)$ to find the closest servers as PSs (Line 3 or Lines 5–6 in Algorithm 3). Greedy selection (Algorithm 4) also needs $\mathcal{O}(N \log N)$ for PS selection (Lines 2–3), plus $\mathcal{O}(\kappa_j \varpi_j)$ for iteratively selecting κ_j workers from $client\_list$ (Lines 4-7). Thus, the worst case of time complexity for selection is $\mathcal{O}(N \log N + \kappa_j \varpi_j)$. Then Algorithm 2 also needs to determine the topology if DFL/HFL with a computational complexity of $\mathcal{O}(\kappa_j^2)$ or $\mathcal{O}(\kappa_j \varpi_j)$. Since $\varpi_j > \kappa_j$, this step takes up to $\mathcal{O}(\kappa_j \varpi_j)$. In summary, the time complexity of Algorithm 2 is bounded by $\mathcal{O}(\kappa_j \varpi_j + N \log N)$.

Algorithm 5 needs $\mathcal{O}(N \log N)$ to find PSs and $\mathcal{O}(\kappa_j^2)$ or $\mathcal{O}(\kappa_j \varpi_j)$ to determine the DFL/HFL topology (Line 1/3). Hence, its time complexity is $\mathcal{O}(N \log N + \kappa_j \varpi_j)$ too.

Overall, our proposed iterative algorithm (i.e., Algorithm 1) iteratively determines the participant selection and learning topology (using Algorithm 2 and Algorithm 5, respectively) for H models within the maximal iteration Z. Therefore, the total time complexity is bounded by $\mathcal{O}(\sum_{j=0}^{H} Z(N \log N + \kappa_j \varpi_j))$. □

V. Performance Evaluation

In this section, we provide a detailed performance evaluation of our proposed algorithms based on simulations over real-world network topology and machine learning tasks.

A. Simulation Setup

Edge Network: To generate edge cloud networks, we leverage a real-world EUA-Dataset [32] to form a more realistic distribution of edge servers. This dataset is widely used in edge computing and contains the geographical locations of 125 cellular stations in central Melbourne. We first randomly select a certain number of stations in this dataset as the edge servers and then determine all edge links to form the network topology based on the distances among edge servers. Each edge server v_i then randomly assigned with a maximal storage capacity c_i, CPU frequency f_i and link bandwidth b_j in ranges of $512 - 1,024$ GB, $2 - 6$ GHz, and $512 - 1,024$ Mbps, respectively.

FL Models: To verify the FL performance over the edge, we conduct a set of FL experiments, where multiple FL models ($1 - 5$ models) need to be concurrently trained in our environment. For each model m_j, it requires κ_j workers ($8 - 16$ workers), and has a specific model size μ_j, CPU requirement χ_j in range $10 - 100$ MB, $1 - 3$ GHz, respectively. We also fix the number of closest neighbors that each FL worker connects in DFL to 4 and the number of middle-layer PSs in HFL to 2 (i.e. $\psi_j = 2$).

Three classical datasets in scikit-learn 1.0.2 [33] are used to train linear regression (LR) models: California Housing dataset, Diabetes dataset, and randomly generated LR datasets. Each LR model is trained with the loss of Mean Square Error (MSE). In addition, we are interested in the performance of the proposed methods in non-convex loss functions. Hence, three different types of image datasets are used to train CNN models: CIFAR10 [34], Fashion-MNIST (FMNIST) [35], MNIST [36]. These are well-known ML datasets for image classification/recognition tasks. For CIFAR10/FMNIST/MNIST datasets, we train CNN models with two $5 \times 5/3 \times 3/2 \times 2$ convolution layers, respectively. For CIFAR10, the first layer has 6 output channels followed by 2×2 max pooling and the second layer has 16 output channels. Then it follows three linear layers and a softmax output layer. For FMNIST, the two layers have 32/64 output channels followed by 2×2 max pooling. For MNIST, the two layers have 16/32 output channels followed by 2×2 max pooling. All datasets are distributed in non-IID settings.

Baselines: We compared our proposed algorithms (**PS-LTS**) with two groups of baselines. In Group 1, all methods iteratively determine the learning topology by Algorithm 5 while performing participant selection using different strategies.

- *RAND-LTS:* Randomly select the participant.
- *NORM-LTS [22]:* Select the participants with top κ_j highest gradient **norm** values for each model.
- *LOSS-LTS [23]:* First sample ϖ_j candidate client with the fraction of data at each client, and then select κ_j workers with the highest **loss** values.

In Group 2, all methods use Algorithm 2 to select participants but determine the learning topology in different ways.

- *PS-CFL:* All models adopt CFL learning topology.

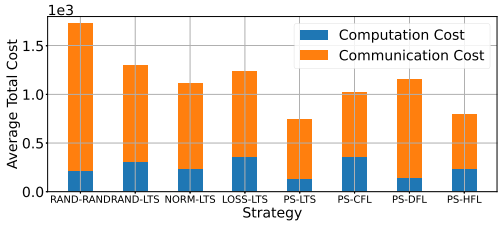

Fig. 4. Performance comparison (i.e., total cost, computation/communication cost) of all methods.

- *PS-DFL:* All models adopt DFL learning topology.
- *PS-HFL:* All models adopt HFL learning topology.

In addition, we include a pure random solution (**RAND-RAND**), which randomly selects the participant and learning topology for each model.

B. Evaluation Results

Overall Performance: Fig. 4 shows the overall average total cost of all methods where 3 models are trained with 10 workers. Clearly, our proposed PS-LTS outperforms other benchmarks by a large margin and thus confirms the advantage of jointly considering participant selection and learning topology selection. For example, PS-LTS achieves lower average total costs with up to 33.5% and 39.6% compared to NORM-LTS and LOSS-LTS, respectively. RAND-RAND has the worst performance since it randomly determines the participants and learning topology without any optimization. Within Group 1, NORM-LTS and LOSS-LTS have smaller total learning costs than RAND-LTS. This shows the advantage of considering gradient norm value or loss value for participant selection. Compared with methods in Group 2 where the same learning topology is used for every model, our proposed method also shows clear advances by adapting appropriate learning topology selection for each model. Fig. 4 also shows that the communication cost dominates the total cost in this experiment due to bandwidth constraints in the edge network.

Learning Cost vs Resources and Max Iterations: We also study the impact of different networking/computing resources (such as network bandwidth and CPU frequency) in edge cloud for our proposed method PS-LTS. As shown in Fig. 5(a) and (b), with more bandwidth (or faster CPU frequency), the communication (or computation) cost of our method can be reduced, which also leads to a lower total cost. We also plot the result of the impact of max iteration threshold max_itr for our method in Fig. 5(c). With more iterations, the learning cost decreases, since it has more chances to find a better solution with a lower cost. However, the improvement becomes smaller when the max iteration further increases.

Learning Cost vs Number of Edge Servers: Next, we investigate the impact of the number of edge servers in the edge cloud for all methods. In this setting, the number of edge servers varies from 50 to 70, and we consider 3 models and 10 workers for each model. Fig. 6 shows the comparison of our method against baselines in both groups. From Fig. 6(a) and (b), we can observe that the total cost of all methods (except RAND-RAND) declines as the number of edge servers rises. This is reasonable since with more servers it has chances to find better participants to join the FL training leading to a lower total cost. It is not surprising that NORM-LTS and LOSS-LTS perform better than RAND-RAND and RAND-LTS while our proposed PS-LTS outperforms other methods in terms of the average total learning cost.

Similar results also appear in Fig. 6(c) and (f) which plots the detail of computation and communication costs. In Fig. 6(c) and (e), LOSS-LTS has the highest computation cost since the server with a higher loss value may own a larger local data. Meanwhile, the communication cost of LOSS-LTS is close to that of NORM-LTS. For the combined total cost, LOSS-LTS still has a higher cost than NORM-LTS as shown in Fig. 6(a). In Fig. 6(d) and (f), the computation cost of PS-DFL is very close to our proposed method but it has the highest communication cost due to the topology complexity of PS-DFL. Recall that DFL forces each worker to connect to a fixed number of closest workers, thus leading to a more complicated topology than CFL and HFL. Furthermore, PS-HFL has the lowest communication cost since each worker connects to the closest middle-layer PS in HFL.

Learning Cost vs Number of Models: We also study the influence of the number of FL models on the costs. Similarly, we set 70 edge servers in edge cloud and aim to train $1-5$ models with 10 FL workers per model. Fig. 7 plots the results. Obviously, with more models, all learning costs increase. Our proposed method still performs better than other methods.

Learning Cost vs Number of Workers: Next, we further inspect the performance impact of the number of workers. Here we train 3 models with the required worker number from 8 to 16 over an edge cloud with 65 servers. As shown in Fig. 8, the total cost of all methods rises with the increase in the number of workers. This is reasonable since more workers cause more computation and communication costs leading to a higher total cost. Additionally, the gap between our proposed method and baseline methods becomes smaller as the number of workers increases due to the domination of the communication cost as shown in Fig. 4. Still, our proposed method is the best among all methods.

Training Loss: We then investigate the training loss of our method in federated learning experiments over LR datasets. We introduce the R2 score metric to evaluate the performance of LR model (convex) training. R2 score is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). In this experiment, we concurrently train 3 LR models with 3 different datasets. Each dataset is split into 65 edge servers unequally (i.e. non-IID setting) and the number of global training rounds is 100. We can see from Fig. 9(b) and (d), that the training loss decreases as the number of workers increases for each model. Fig. 9(a) shows the R2 score of all LR models. Obviously, with more workers, the R2 score of all models increases, which means all models are well-regressed. However, model 2 has a worse R2 score (a negative value) in fewer workers due to the small size of the training dataset. But as the number of workers increases, the performance of model 2 becomes better.

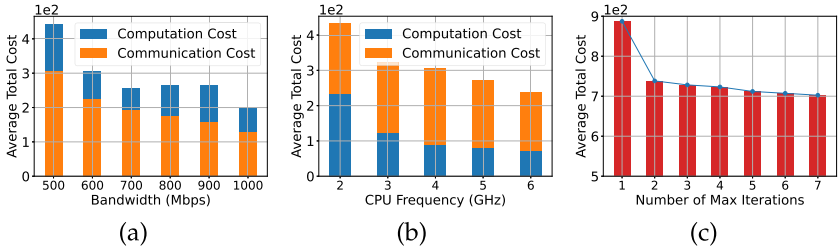

Fig. 5. Impact of (a) bandwidth, (b) CPU frequency, and (c) max_iter on our method.

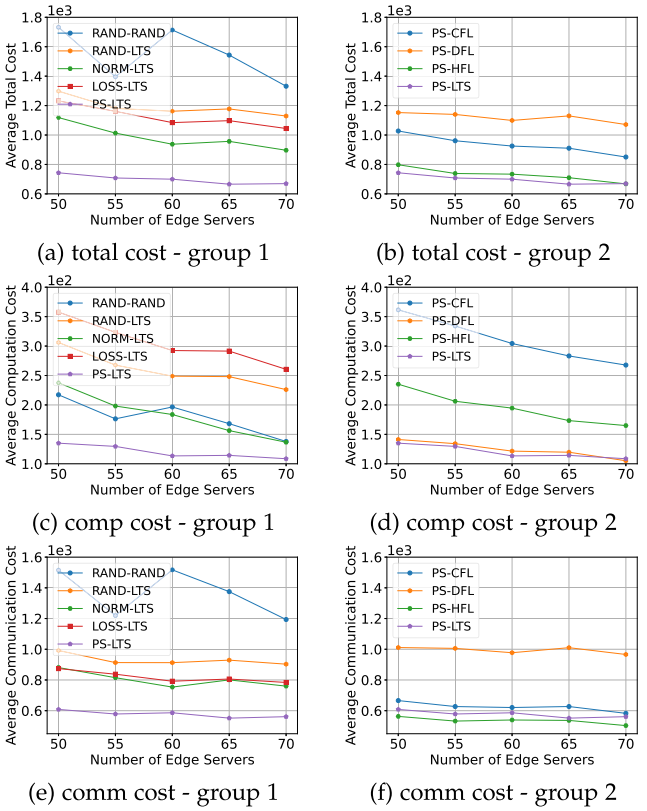

Fig. 6. Impact of number of edge servers on costs: (a-b) total cost, (c-d) computation cost, (e-f) communication cost.

Training Accuracy: Finally, we look into the training accuracy of our method over some benchmarks in CNN model (non-convex) training. Here, we apply to consider the training of 3 FL models with an edge cloud of 65 edge servers, each FL model uses different datasets (i.e., CIFAR10, FMNIST, and MNIST) and adopts the non-IID setting. Fig. 10(a) shows the accuracy of models for baselines (RAND-LTS, NORM-LTS, LOSS-LTS) and our proposed method. We can find that the training accuracy of all strategies is similar but recall that from Fig. 8 our proposed method takes less cost to reach a similar level of accuracy. In addition, Fig. 10(b) and (d) shows the detailed training accuracy of three different models with different numbers of workers for our proposed method. We can see that the training accuracy grows with more workers, but it

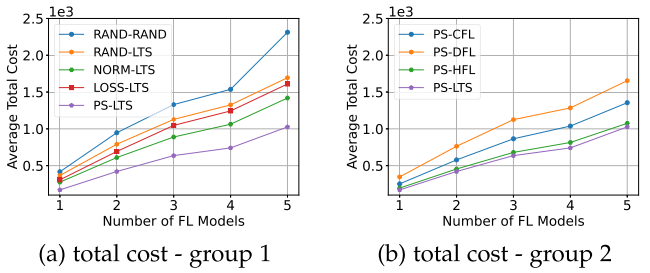

Fig. 7. Impact of number of models on costs.

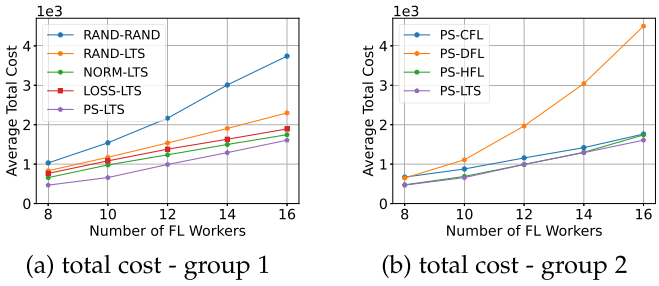

Fig. 8. Impact of number of workers on costs.

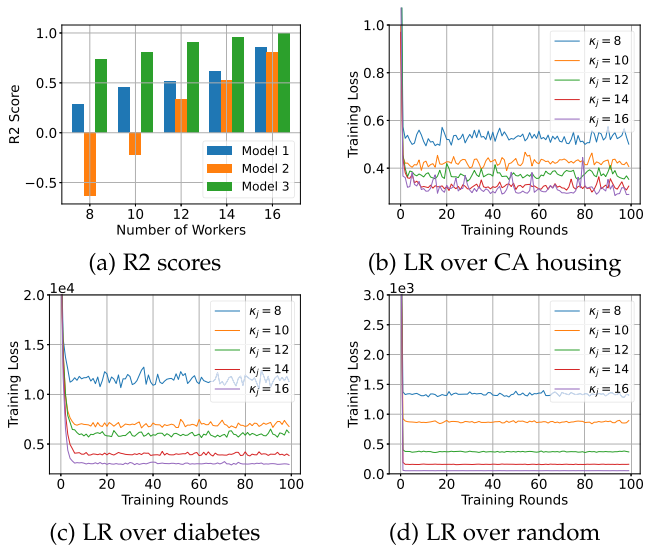

Fig. 9. Training loss of LR models and impact from the number of workers: (a) R2 scores of three LR models; (b) loss of LR over California Housing dataset; (c) loss of LR over diabetes dataset; (d) loss of LR over the randomly generated dataset.

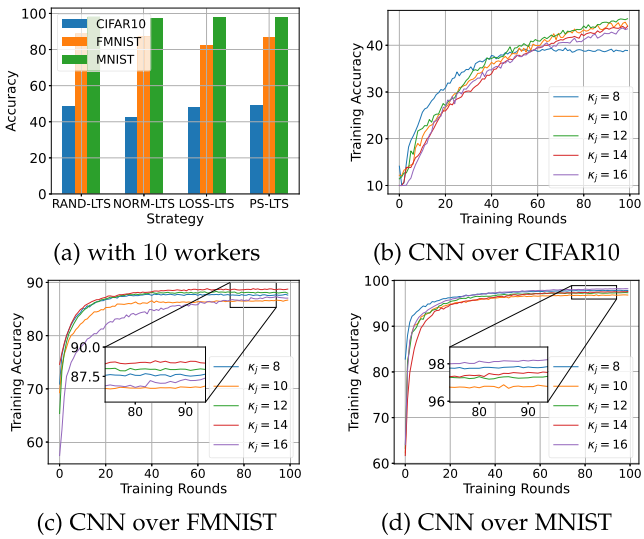

Fig. 10. Training accuracy of CNN models and impact from the number of workers: (a) accuracy of all strategies with 10 workers; (b)-(d) accuracy of three models using our method with the different number of workers.

varies due to the non-IID setting and the unequal data partition in the experiment. In some cases, more workers do not mean more distinct data is being selected so the training accuracy may drop a little as the number of workers increases. But overall all three models (CIFAR10, FMNIST, and MNIST) share similar trends.

VI. Related Work

In this section, we briefly review related work in federated learning at the edge, learning topology of FL, and participant selection in FL.

A. Federated Learning at the Edge

Federated learning [7], [8], [9], [10], [11], [21] has been emerging as a new distributed learning paradigm in edge computing. Most existing works of FL at the edge focus on the convergence and adaptive control of FL or the resource allocation and learning scheduling within the edge network/system. For example, Wang et al. [7] focused on FL's convergence and adaptive control in edge computing without participant selection. They proposed a control algorithm to determine the trade-off between local update and global parameter aggregation to minimize the loss function. Jin et al. [9] considered a joint control of FL and edge provisioning problems in distributed cloud-edge networks and proposed a method to control the status of edge servers to minimize the long-term cumulative cost of FL while satisfying the training models' convergence. Nishio and Yonetani [21] studied the client selection problem in an edge-based FL where the edge server in a cellular network acts as the PS and a set of mobile clients are selected as workers, to select as many mobile clients to complete the FL training under the time constraints.

B. Different Learning Topology

Most prior works focused on the classical CFL learning topology [1], [2], [3], [7], [9]. Sattler et al. [2] proposed a new compression framework called sparse ternary compression (STC) to meet the requirement of the CFL environment from non-I.I.D data. Ji et al. [3] studied the device sampling problem in CFL and proposed a communication-efficient algorithm to reduce the overall communication cost during the FL training by dynamic sampling and top-k selective masking method. On the other hand, some works also concentrated on DFL [10], [13], [45] and HFL [11], [16], [17], [20]. Meng et al. [10] focused on the FL model training using decentralized P2P methods in edge computing. They also select the FL workers from the edge network and dynamically form the decentralized topology (no PS) for FL training by a reinforcement learning method. Luo et al. [20] considered a client-edge-cloud HFL where the cloud acts as the top-tier parameter server to aggregate the FL models from edge servers and edge servers work as middle-layer parameter servers to aggregate the partial models from mobile clients. As stated before, all of these works only considered a specific learning topology in FL training and also did not consider the competition among multiple FL models.

C. Participant Selection in FL

Participant selection [22], [23], [24], [25], [26], [27], [28], [30], [31], [46], [47], also known as client selection or sampling, has been well studied in FL. For example, Qu et al. [46] studied the client selection problem in HFL scenario and proposed a context-aware online algorithm based on the Multi-Armed Bandit (MAB) framework. This framework leverages clients' computational and transmission information to select a subset of clients to maximize the training utilities. Similarly, Li et al. [26] considered client selection as the contextual-combinatorial MAB framework with fairness requirement and proposed a deadline-aware task replication for surplus client scheduling policy. Cho et al. [47] considered the biased client selection problem and proposed a power-of-choice algorithm and its variants to minimize communication and computation overheads. Wei et al. [27], [28], [29], [30], [31] studied a participant selection problem in either HFL or CFL aiming to minimize the total learning cost in an edge cloud. However, all of these works only considered a specific learning topology for all models, either CFL or HFL.

Overall, previous works only consider one FL model or a specific learning topology when conducting FL training or selecting FL participants. In this work, we are the first to consider adaptive learning topology and participant selection for multi-model FL scenarios. By carefully selecting the training participant and learning topology for all FL models, one can effectively manage resource competition and load balancing in edge clouds.

VII. Conclusion

This paper investigates the problem of jointly selecting participants and adaptive learning topologies for multiple federated

learning models being trained concurrently in edge clouds. We formulate the problem as an integer non-linear programming problem and propose a two-stage iterative algorithm to solve it by breaking it into two sub-problems. In Stage 1, we solve the participant selection problem based on the initial or Stage 2-derived learning topology via different heuristic algorithms. In Stage 2, we establish the learning topology for each model based on Stage 1's selected participants, and then choose the best one. This iterative process continues until certain conditions are satisfied. Through extensive simulations and actual FL experiments, we demonstrate that our proposed algorithm outperforms existing methods with up to 33.5% and 39.6% lower average total costs in a multi-model FL environment.

As a concluding remark, we expect that our proposed algorithm can be further extended to solve larger-scale FL scenarios and can be deployed in a real edge testbed. Moreover, we will investigate potential reinforcement learning approaches and privacy protection mechanisms to enhance and complement our proposed framework to support secure large-scale FL in dynamic edge clouds.

REFERENCES

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

[2] F. Sattler, S. Wiedemann, K.E. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. on Neural Netw. Learn. Sys.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[3] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," *IEEE Intell. Syst.*, vol. 37, no. 2, pp. 27–34, Mar./Apr. 2022.

[4] Z. Jiang, W. Wang, B. Li, and Q. Yang, "Towards efficient synchronous federated training: A survey on system optimization strategies," *IEEE Trans. Big Data*, vol. 9, no. 02, pp. 437–454, Apr. 2023.

[5] Z. Tang, S. Shi, B. Li, and X. Chu, "GossipFL: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 3, pp. 909–922, Mar. 2023.

[6] Y. Li, F. Li, S. Yang, C. Zhang, L. Zhu, and Y. Wang, "A cooperative analysis to incentivize communication-efficient federated learning," *IEEE Trans. Mobile Comput.*, pp. 1–16, Mar. 2024, doi: 10.1109/TMC.2024.3373501.

[7] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[8] M. N. Nguyen, N. H. Tran, Y. K. Tun, Z. Han, and C. S. Hong, "Toward multiple federated learning services resource sharing in mobile edge networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 541–555, Jan. 2023.

[9] Y. Jin, L. Jiao, Z. Qian, S. Zhang, and S. Lu, "Learning for learning: Predictive online control of federated learning with edge provisioning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[10] Z. Meng, H. Xu, M. Chen, Y. Xu, Y. Zhao, and C. Qiao, "Learning-driven decentralized machine learning in resource-constrained wireless edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[11] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[12] D. Xu et al., "Edge Intelligence: Empowering intelligence to the edge of network," *Proc. IEEE*, vol. 109, no. 11, pp. 1778–1837, Nov. 2021.

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

[14] L. Yuan, L. Sun, P. S. Yu, and Z. Wang, "Decentralized Federated Learning: A Survey and Perspective," 2023, *arXiv:2306.01603*.

[15] A. Wainakh, A. S. Guinea, T. Grube, and M. Mühlhäuser, "Enhancing privacy via hierarchical federated learning," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, 2020, pp. 344–347.

[16] H. Yang, "H-FL: A hierarchical communication-efficient and privacy-protected architecture for federated learning," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 479–485.

[17] Y. Deng et al., "SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 24–34.

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

[19] J. Liu, X. Wei, X. Liu, H. Gao, and Y. Wang, "Group-based hierarchical federated learning: Convergence, group formation, and sampling," in *Proc. 52nd Int. Conf. Parallel Process.*, 2023, pp. 264–273.

[20] 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 Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[21] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.

[22] O. Marnissi, H. E. Hammouti, and E. H. Bergou, "Client selection in federated learning based on gradients importance," 2021, *arXiv:2111.11204*.

[23] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," 2020, *arXiv: 2010.01243*.

[24] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," 2020, *arXiv: 2010.13723*.

[25] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[26] Y. Li, F. Li, L. Chen, L. Zhu, P. Zhou, and Y. Wang, "Power of redundancy: Surplus client scheduling for federated learning against user uncertainties," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5449–5462, Sep. 2023.

[27] X. Wei, J. Liu, X. Shi, and Y. Wang, "Participant selection for hierarchical federated learning in edge clouds," in *Proc. IEEE Int. Conf. Netw. Archit. Storage*, 2022, pp. 1–8.

[28] X. Wei, J. Liu, and Y. Wang, "Joint participant selection and learning scheduling for multi-model federated edge learning," in *Proc. IEEE Int. Conf Mobile Ad-Hoc Smart Syst.*, 2022, pp. 537–545.

[29] X. Wei, J. Liu, and Y. Wang, "Joint participant selection and learning optimization for federated learning of multiple models in edge cloud," *J. Comput. Sci. Technol.*, vol. 38, no. 4, pp. 754–772, 2023.

[30] X. Wei, L. Fan, Y. Guo, Y. Gong, Z. Han, and Y. Wang, "Quantum assisted scheduling algorithm for federated learning in distributed networks," in *Proc. 32nd Int. Conf. Comput. Commun. Netw.*, 2023, pp. 1–10.

[31] X. Wei, L. Fan, Y. Guo, Y. Gong, Z. Han, and Y. Wang, "Hybrid quantum-classical Benders' decomposition for federated learning scheduling in distributed networks," *IEEE Trans. Netw. Sci. Eng.*, pp. 1–13, Apr. 2024.

[32] P. Lai et al., "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. 16th Int. Conf. Service-Oriented Comput.*, 2018, pp. 230–245.

[33] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[34] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Tront, 2009.

[35] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.

[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[37] C. Ma et al., "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul./Aug. 2020.

[38] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.

[39] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–26.

[40] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019, *arXiv: 1908.07782*.

[41] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.
[42] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[43] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.
[44] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Demystifying why local aggregation helps: Convergence analysis of hierarchical SGD," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2022, pp. 8548–8556.
[45] X. Liu, Y. Li, R. Wang, J. Tang, and M. Yan, "Linear convergent decentralized optimization with compression," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–30.
[46] Z. Qu, R. Duan, L. Chen, J. Xu, Z. Lu, and Y. Liu, "Context-aware online client selection for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4353–4367, Dec. 2022.
[47] Y.J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 10351–10375.

Xinliang Wei (Member, IEEE) received the BE and MS degrees in software engineering from SUN Yat-sen University, Guangzhou, China in 2014 and 2016, respectively and the PhD degree in computer and information sciences from Temple University, Philadelphia, USA in 2023. He is an assistant professor in Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include edge computing, federated learning, reinforcement learning, and Internet of Things. He is a recipient of *Outstanding Research Assistant Award* from College of Science and Technology (2022) and *Scott Hibbs Future of Computing Award* from Department of Computer & Information Sciences (2023) with Temple University.

Kejiang Ye (Senior Member, IEEE) received the BS and PhD degrees from Zhejiang University in 2008 and 2013, respectively. He is currently a professor and the deputy director of the Research Center for Cloud Computing, Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS). Before joining SIAT, he was a postdoctoral research associate with Carnegie Mellon University (CMU), from 2014 to 2015, and was a research fellow with Wayne State University (WSU), from 2015 to 2016. His research interests include cloud computing, Big Data, and industrial internet.

Xinghua Shi (Member, IEEE) received the BEng and MEng degrees in computer science from the Beijing Institute of Technology, China and the MS and PhD degrees in computer science from the University of Chicago. She is an associate professor with the Department of Computer and Information Sciences and a core faculty member with the Institute for Genomics and Evolutionary Medicine, Temple University. She was trained as a postdoctoral scholar and genetics fellow with Brigham and Women's Hospital, Harvard Medical School, and Broad Institute. Her research interests include machine learning, bioinformatics, data privacy, and Big Data analytics in biomedical research. Her research has been supported by NSF (including an NSF CAREER award), NIH, DoD and Wells Fargo Foundation.

Cheng-Zhong Xu (Fellow, IEEE) received the BSc and MSc degrees in computer science and engineering from Nanjing University in 1986 and 1989, respectively, and the PhD degree in computer science and engineering from The University of Hong Kong in 1993. He was a professor of electrical and computer engineering with Wayne State University and the director of advanced computing and digital engineering with the Shenzhen Institutes of Advanced Technology (SIAT). He is currently the dean of the Faculty of Science and Technology, University of Macau, and a chair professor of computer and information science. He also holds a courtesy position as the director of the Center for Cloud Computing, SIAT, Chinese Academy of Sciences. He published two research monographs and more than 300 papers in journals and conference proceedings, including more than 50 in IEEE/ACM transactions; his publications received more than 12800 citations with an H-index of 60. His main research interests include parallel and distributed computing, with an emphasis on resource management for system performance, reliability, availability, power efficiency, and security, and in Big Data and data-driven intelligence applications. He was the Best Paper Nominee or Awardee of the 2013 IEEE High Performance Computer Architecture (HPCA), the 2013 ACM High Performance Distributed Computing (HPDC), IEEE Cluster 2015, ICPP 2015, GPC 2018, and UIC 2018.

Yu Wang (Fellow, IEEE) received the BEng and MEng degrees in computer science from Tsinghua University and the PhD degree in computer science from the Illinois Institute of Technology. He is a professor and chair of the Department of Computer and Information Sciences, Temple University. His research interest includes wireless networks, smart sensing, and mobile computing. He has published more than 300 papers in peer-reviewed journals and conferences. He is a recipient of *Ralph E. Powe Junior Faculty Enhancement Awards* from Oak Ridge Associated Universities (2006), *Outstanding Faculty Research Award* from College of Computing and Informatics with the University of North Carolina at Charlotte (2008), *fellow of IEEE* (2018), *ACM distinguished member* (2020), and *IEEE Benjamin Franklin Key Award* (2024). He has served as associate editor for *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, among others.