Wei X, Liu J, Wang Y. Joint participant selection and learning optimization for federated learning of multiple models in edge cloud. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 38(4): 754-772 July 2023. DOI: 10.1007/s11390-023-3074-4

Joint Participant Selection and Learning Optimization for Federated Learning of Multiple Models in Edge Cloud

Xinliang Wei, *Student Member, IEEE*, Jiyao Liu, and Yu Wang*, *Fellow, IEEE, Distinguished Member, ACM*

Wireless Networking and Sensing (Wang) Laboratory, Department of Computer and Information Sciences Temple University, Philadelphia, PA 19122, U.S.A.

E-mail: xinliang.wei@temple.edu; jiyao.liu@temple.edu; wangyu@temple.edu

Received March 21, 2023; accepted July 31, 2023.

Abstract To overcome the limitations of long latency and privacy concerns from cloud computing, edge computing along with distributed machine learning such as federated learning (FL), has gained much attention and popularity in academia and industry. Most existing work on FL over the edge mainly focuses on optimizing the training of one shared global model in edge systems. However, with the increasing applications of FL in edge systems, there could be multiple FL models from different applications concurrently being trained in the shared edge cloud. Such concurrent training of these FL models can lead to edge resource competition (for both computing and network resources), and further affect the FL training performance of each other. Therefore, in this paper, considering a multi-model FL scenario, we formulate a joint participant selection and learning optimization problem in a shared edge cloud. This joint optimization aims to determine FL participants and the learning schedule for each FL model such that the total training cost of all FL models in the edge cloud is minimized. We propose a multi-stage optimization framework by decoupling the original problem into two or three subproblems that can be solved respectively and iteratively. Extensive evaluation has been conducted with real-world FL datasets and models. The results have shown that our proposed algorithms can reduce the total cost efficiently compared with prior algorithms.

Keywords edge computing, federated learning (FL), participant selection, learning optimization

1 Introduction

Nowadays, a massive amount of data is generated by mobile users, Internet of Things (IoT) devices and artificial intelligence applications, providing potential training datasets for diverse machine learning (ML) tasks. Traditionally, one needs to upload the whole dataset to the remote cloud center for centralized machine learning model training. However, it is non-trivial to upload a large amount of data to the remote data center due to the limited network bandwidth and data privacy concerns. Since training data is generated at the network edge, such as from smart sensing devices and smartphones connected to the edge, edge computing coupled with distributed machine learning is a natural alternative. Nevertheless, training ML models in the edge cloud still faces many challenges. First, due to limited data and computing resources, a single edge device/server may not be able to perform a high-quality ML training task alone. Second, the computing capacity and network resources of edge devices/servers are limited and heterogeneous. When performing ML training tasks, different edge units may lead to various convergence speeds and performances. Third, edge resources generally are shared by many mobile users. Distributed ML training within the edge cloud has to be constrained by the shared resources and the resource competition among vari-

Regular Paper

Recommended by MASS 2022

A preliminary version of the paper was published in the Proceedings of MASS 2022.

This work is partially supported by the US National Science Foundation under Grant Nos. CCF-1908843 and CNS-2006604.

*Corresponding Author

ous users, servers, and applications.

To tackle the aforementioned challenges, a new distributed machine learning paradigm has been proposed, called federated learning (FL)[1–3] that conducts distributed learning at multiple clients without sharing raw local data among themselves. Coupled with edge computing, FL over edge cloud has been recently studied in various settings[4–14]. In such scenarios, several edge servers have been selected as participants (either parameter servers or FL workers), and collaboratively train a shared global ML model without sharing their local datasets and decoupling the ability to do model training from the need to store data in a centralized server. More precisely, as shown in Fig.1, in each global iteration, edge servers, worked as workers, first download the latest global model from the parameter server (PS), and then perform a fixed number of rounds of local training based on their local data. After that, edge servers will upload their local models to the parameter server which is responsible for aggregating parameters from different workers and sending the aggregated global model back to each FL worker. Previously, the efforts of FL over the edge have been focused on the convergence and adaptive control[5, 6], the resource allocation and model aggregation[8, 10, 11], and the communication and energy efficiency[1, 16, 17].

In this work, we focus on a joint participant selection and learning optimization problem in multi-model FL over a shared edge cloud[①]. For each FL model, we aim to find one PS and multiple FL workers and decide the local convergence rate for FL workers. Note that both worker selection and learning rate control have been studied in FL recently. For example, Nishio *et al.*[7] studied a client selection problem in decentralized edge learning where a set of mobile clients are chosen to act as workers for FL and their aim is to maximize the selected clients under time constraints. Jin *et al.*[9] also studied the joint control of local learning rate and edge provisioning in FL to minimize the long-term cumulative cost. However, all of these studies focus on the optimization of training one global FL model instead of multiple FL models, thus they do not consider the parameter server selection for different FL models and ignore the competition of resources among different FL models. In the real scenario, there might be multiple FL models training simultaneously in the edge cloud (Fig.1 shows an example where two FL models are trained with three and four participants, respectively). Especially in the edge computing environment, edge servers can store different types of data and serve different FL models or tasks for diverse applications. With heterogeneous resources and capacities at edge devices, when multiple FL models are trained at the same time, which FL model is preferentially served at

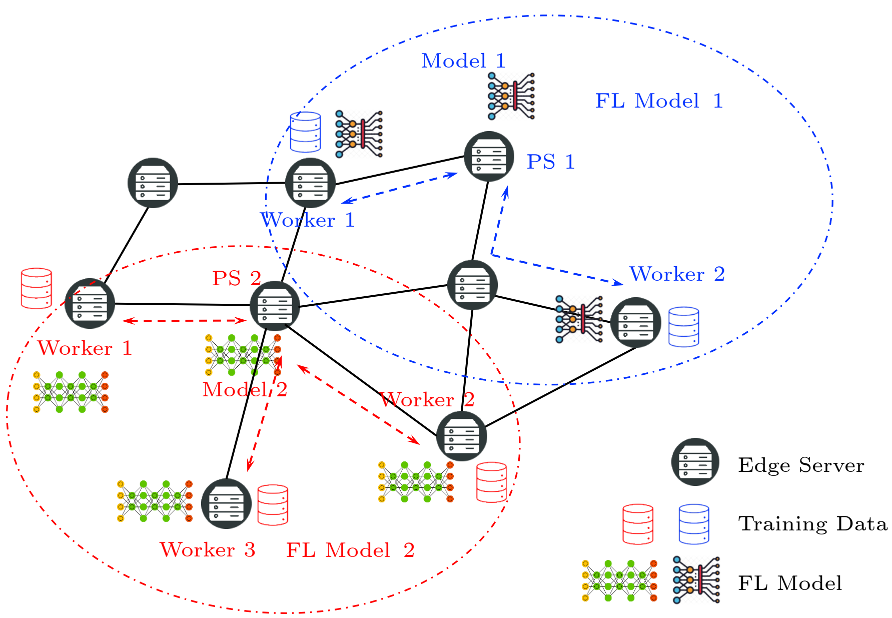

Fig.1. Example of multi-model FL over the edge: two FL models are trained with three and four participants (1 PS + 2 or 3 FL workers), respectively, in a shared edge cloud[15].

[①]As shown in Fig.1, we consider an edge cloud architecture where a set of edge servers are connected to each other without the remote cloud center to form an edge network to serve the users. This kind of edge cloud model has been widely used in [18–22].

which edge server directly affects the total communication cost and computational cost of the FL training. The selection of participants (both the PS and FL workers) for each model will also affect the learning convergence speed. Therefore, in this paper, we formulate a new joint optimization problem of participant selection and learning rate scheduling of multi-model FL, where multiple FL models are trained concurrently in an edge cloud.

Due to the limitation of shared computing and networking resources at edge servers in the edge cloud, we aim to carefully select the FL participants for each FL model and pick the appropriate local learning rate for these selected FL workers, so as to minimize the total cost of FL training of all models while meeting the convergence requirement from each model. The main contributions of this work are summarized as follows.

1) We formulate a new joint participant selection and learning rate scheduling problem of multi-model federated learning in an edge cloud as a mixed-integer programming problem, with a goal to minimize the total FL cost while satisfying various constraints. Note that by allowing different FL models to pick their own PS, it can better handle the competition among models and provide load balancing in the edge cloud compared with the traditional FL solution with a centralized PS for all models.

2) We decouple the original optimization problem into two or three sub-problems and then propose three algorithms to effectively find participants and the learning rate for each FL model, by iteratively solving the sub-problems. We further consider the impact of the processing order of FL models in a resource-limited and heterogeneous edge scenario.

3) We conduct extensive simulations with real FL tasks to evaluate our proposed algorithms, and our results confirm the proposed algorithms can effectively reduce the total FL cost compared with the existing work[9, 23, 24].

The rest of the paper is organized as follows. Section 2 presents the overview of related work. Section 3 introduces the system model and the preliminaries of federated edge learning. Section 4 describes the problem formulation of new joint optimization, and Section 5 provides our proposed multi-stage optimization algorithms. Evaluation of our proposed algorithms is provided in Section 6. Section 7 finally concludes this paper. A preliminary version of this paper appears as [15], and this version includes newly introduced greedy-based variation using a different model processing order, time complexity analysis, additional sets of experiment results, more comprehensive related work, and better overall presentation.

2 Related Work

In this section, we briefly review recent studies on federated learning over edge systems[4]. Federated learning[5–14] has been emerging as a new distributed ML paradigm over different edge systems. Current FL frameworks can be categorized into three types based on the learning topology used for model aggregation: centralized FL (CFL), hierarchical FL (HFL), and decentralized FL (DFL). CFL is the classical FL[9] where the parameter server (PS) and several workers form a star architecture as shown in Fig.1[15]. Wang *et al.*[6] analyzed the convergence of CFL in a constrained edge computing system and proposed a control algorithm that determines the best trade-off between local update and global parameter aggregation to minimize the loss function. This work focused on the convergence and adaptive control of FL and did not consider participant selection.

Nishio and Yonetani[7] studied a client selection problem in CFL in mobile edge computing. Their method uses an edge server in the cellular network as the PS and selects a set of mobile clients as workers. Their client selection aims to maximize the selected workers while meeting the time constraints. Jin *et al.*[9] considered a joint control of FL and the edge provisioning problem in distributed cloud-edge networks where the cloud server is the PS and active edge servers are workers. Their method controls the status of edge servers for training to minimize the long-term cumulative cost of FL and also satisfies the convergence of the trained model. Li *et al.*[25] also considered client scheduling in FL to overcome client uncertainties or stragglers via learning-based task replication. While these studies are similar to ours, they focus on the optimization of one global FL model instead of multiple FL models. More importantly, these studies do not consider PS selection for multiple FL models.

Recently, Nguyen *et al.*[14] studied resource sharing among multiple FL services/models in edge computing where the user equipment is used as an FL worker, and proposed a solution to optimally manage the resource allocation and learning parameter control while ensuring energy consumption requirements.

However, their FL framework is different from ours. First, they used the user equipment as FL workers, while we use edge servers as FL workers. Second, they did not consider the PS selection, since they used a single edge server as the PS. Third, their model allows training multiple FL models at the same user equipment (while we do not allow the edge server to act as workers for multiple models in the same time unit), and thus their method has to manage the CPU and bandwidth allocation on the user equipment.

Both [5] and [8] consider a client-edge-cloud hierarchical federated learning (HFL) where cloud and edge servers work as two-tier parameter servers to aggregate the partial models from mobile clients (i.e., FL workers). Liu *et al.*[5] proved the convergence of such an HFL, while Luo *et al.*[8] also studied a joint resource allocation and edge association problem for device users under such an HFL framework to achieve global cost minimization. Wang *et al.*[11] considered the cluster structure formation in HFL where edge servers are clustered for model aggregation. Recently, Wei *et al.*[12] also studied the participant selection for HFL in edge clouds to minimize the learning cost. Liu *et al.*[13] studied the group formation and sampling in a group-based HFL. However, our FL framework does not use HFL.

Meng *et al.*[10] focused on model training of DFL using decentralized P2P methods in edge computing. While their method also selects FL workers from an edge network, the model aggregation is performed at edge devices based on a dynamically formatted P2P topology (without PS). Therefore, it is different from our studied problem, which mainly focuses on CFL.

There are also other studies[16, 17, 26] where energy efficiency and/or wireless communication have/has been taken into consideration in FL in edge systems.

3 Federated Learning over Edge Cloud

In this section, we introduce our model of edge cloud, and then describe the procedures and associated costs of federated learning over the edge cloud.

3.1 Edge Cloud Model

We model the edge cloud as a graph $G(V, E)$, consisting of N edge servers and L direct links among them, as shown in Fig.2. Here $V=\{v_1,\dots,v_N\}$ and $E=\{e_1,\dots,e_L\}$ are the set of edge servers and the set of links, respectively. For each edge server $v_i \in V$, it

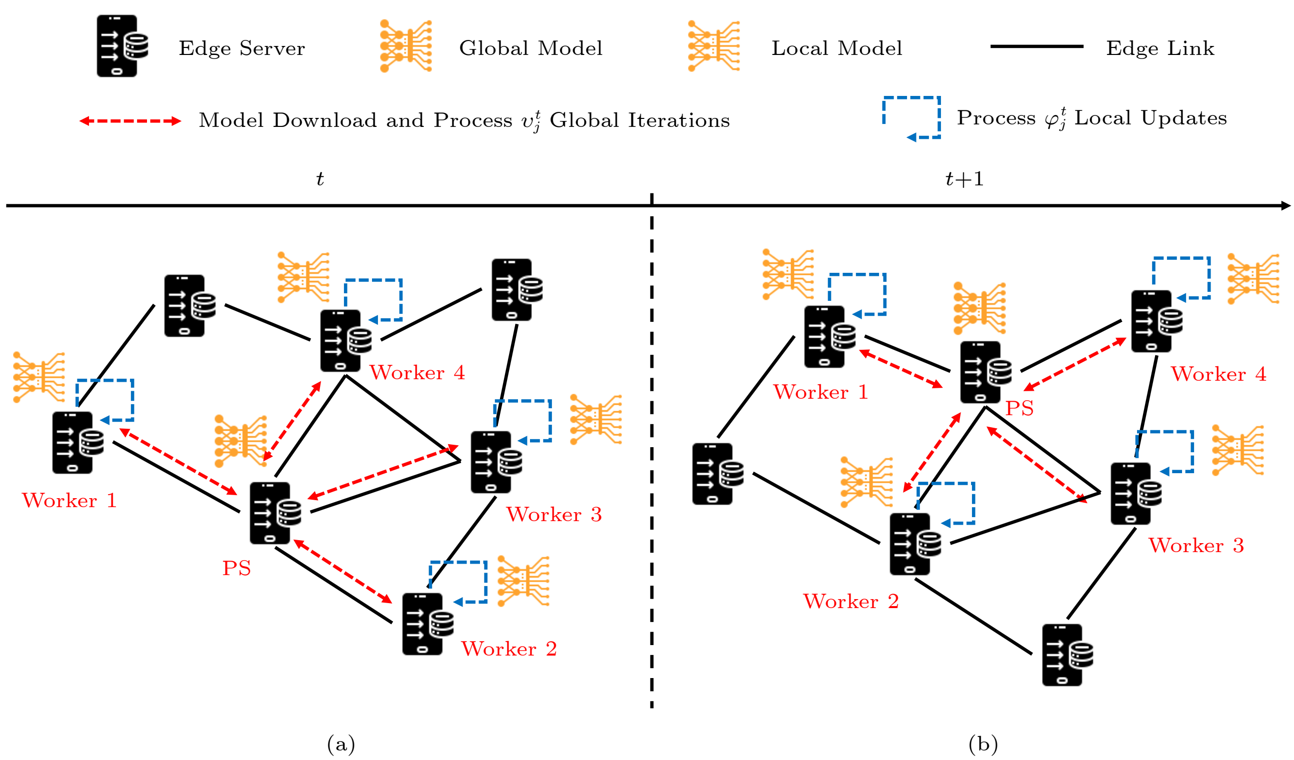

Fig.2. Training process of an FL model within the edge network at different time periods. (a) At each time period, this FL model needs to select one PS and four workers, and they perform the FL via multiple iterations of local and global updates. (b) Due to the dynamic edge cloud environment, the model can change its participants (both PS and workers) at the next time period $t+1$ to minimize the total training cost of all models.

has a storage capacity c_i^t and a CPU frequency f_i^t at time t. For each edge link $e_j \in E$, it has a network bandwidth b_j^t at time t. We omit t from the above notations when it is clear from the context.

Each edge server holds a certain distinct dataset collected from mobile devices/users and can be used for local model training. We consider O types of datasets in the edge cloud, and use $z_{i,k} \in \{0,1\}$ to indicate whether server v_i stores the k-th type of datasets and $S_{i,k}$ to represent the raw sample data of the k-th type stored at server v_i. Note that one edge server can hold multiple types of datasets.

3.2 Federated Learning over Edge

We consider parallel federated learning where multiple machine learning models are trained in parallel within the edge cloud. Compared with the classical FL scenario where the remote cloud works as the parameter server (PS), we select a group of edge servers with enough capacity as the participants (one PS and multiple workers) of FL for each model. We assume that W FL models ($M = \{m_1, \ldots, m_W\}$) need to be trained at the same time. For the training task of each FL model m_j, it requests 1) $\kappa_j + 1$ edge servers as participants, one server as its PS and κ_j servers as its workers, whose CPU frequency should be larger than its required minimal CPU frequency χ_j; 2) the selected workers must have the requested types of a dataset for m_j, where $w_{j,k} \in \{0,1\}$ indicates whether m_j needs the k-th type of datasets; and 3) the achieved global convergence rate needs to be larger than ς_j. Here, we assume that each model uses a fixed number of workers, and one worker can only perform the FL training of one model at a time.

We consider a series of consecutive time periods $t = 1, \ldots, T$, and each time period has an equal duration τ. As shown in Fig.2, at each time t, we select the FL participants for each model and then train W models in parallel through FL, which consists of a number of global iterations (let ϑ_j^t be the number of global iterations of m_j at t). For each model m_j, each global iteration includes four parts: 1) the selected parameter server initializes the global model of m_j; 2) the selected workers download the global model from the parameter server; 3) each worker runs the local updates using its holding raw dataset for φ_j^t local iterations to achieve the desired local convergence rate ϱ_j^t; 4) workers upload the updated model and related gradient to the parameter server for the aggregation to upload the global model. The training process of federated learning at different time periods is shown in Fig.2. Table 1 summarizes all notations we used in this paper.

Next, we define our local training and global aggregation process as well as the loss function during the federated edge learning at each time period.

Loss Function. Let all types of sample data used

Table 1. Summary of Notations

Symbol	Notation
N, L, O, W	Number of edge servers, edge links, datasets, and FL models respectively
v_i, e_j	The i-th edge server, and the j-th edge link respectively
c_i^t, f_i^t, b_j^t	Storage capacity of v_i, CPU frequency of v_i , and the link bandwidth of link e_j at t respectively
$S_{i,k}$, $z_{i,k}$	The k-th type data at v_i, and its indicator respectively
t, τ, T	Index, duration, and number of time periods respectively
m_j, κ_j, μ_j, χ_j	The j-th FL model, its required number of workers, model size, and CPU requirement respectively
ς_j	Global convergence requirement of m_j
$w_{j,k}$, η_j	Indicator of the k-th type data, and the downloading cost of m_j respectively
$x_{i,j}^t$, $y_{i,j}^t$	PS and FL worker selection of e_i for m_j respectively
ϱ_j^t	Local convergence rates of m_j at the t-th time period
ϑ_j^t, φ_j^t	Number of global iterations, and local update of m_j at the t-th time period respectively
α, β, δ	Index of global iteration, local update, and step size of local update respectively
λ, γ	Parameters of loss function
$D_{j,i}^t$, $\psi(D_{j,i}^t)$	Sample data for m_j at sever v_i at t, and CPU cycles to process sample data of $D_{j,i}^t$ respectively
$\rho_j(v_i, v_k)$, $P_{i,k}^t$	Communication cost of m_j from v_i to v_k, and the shortest path from v_i to v_k at the t-th time period respectively
$v_{ps}^t(m_j)$	Selected PS of m_j at the t-th time period
$C_j^{\mathrm{comm},t}$, $C_j^{\mathrm{init},t}$, $C_j^{\mathrm{local},t}$, $C_j^{\mathrm{global},t}$	Communication cost, initialization cost, local update cost, and global aggregation cost of m_j respectively
ϖ_j^t	Total FL cost of m_j

by the j-th model and stored in edge server v_i be defined by $D^t_{j,i} = \bigcup_{w_{j,k}o_{i,k}=1} S_{i,k}$. For each sample data $d = <q_d, r_d> \in D^t_{j,i}$, where q_d is the input data and r_d is the output data/label, we define the average loss of data for the j-th FL model on server v_i in time period t as $\mathcal{A}^t_{j,i}(p)$:

$$\mathcal{A}^t_{j,i}(p) = \frac{1}{|D^t_{j,i}|} \sum_{d \in D^t_{j,i}} \mathcal{H}(\mathcal{I}(q_d; p), r_d),$$

where $\mathcal{H}(\cdot)$ is the loss function to measure the performance of the training model $\mathcal{I}(\cdot)$, and p is the model parameter.

Then the average loss of data for the j-th FL model on all related edge servers in the t-th time period is defined as follows:

$$\mathcal{A}^t_j(p) = \sum_i \frac{|D^t_{j,i}|}{|D^t_j|} \mathcal{A}^t_{j,i}(p),$$

where D^t_j is the union of all involved training samples of model j at time t.

Local Training on FL Workers. For each global iteration of the j-th FL model $\alpha \in [1, \vartheta^t_j]$, the related edge server v_i (FL worker) will perform the following local update process:

$$p^{t,\alpha}_{j,i} = p^{t,\alpha-1}_j + \omega^{t,\alpha}_{j,i},$$

where $p^{t,\alpha}_{j,i}$ is the local model parameter on edge server v_i in the current iteration and $p^{t,\alpha-1}_j$ is the aggregated model downloaded from the parameter server in the last iteration. And $p^{t,0}_j = p^{t-1,\vartheta^t_j}_j$. $\omega^{t,\alpha}_{j,i}$ is the local update from a gradient-based method, and it can be calculated as follows.

$$\omega^{t,\alpha}_{j,i} = \sum_{\beta=1}^{\varphi^t_j} \omega^{t,\alpha,\beta}_{j,i} = \sum_{\beta=1}^{\varphi^t_j} \{\omega^{t,\alpha,\beta-1}_{j,i} - \delta \nabla \mathcal{L}^{t,\alpha}_{j,i}(\omega^{t,\alpha,\beta-1}_{j,i})\},$$

where $\omega^{t,\alpha,\beta}_{j,i}$ is the model parameter for the j-th FL model in the β-th local update and δ is the step size of the local update. Lastly, $\mathcal{L}^{t,\alpha}_{j,i}(\cdot)$ is the predefined local update function. Based on [27], $\mathcal{L}^{t,\alpha}_{j,i}(\cdot)$ is defined as below.

$$\mathcal{L}^{t,\alpha}_{j,i}(\omega) = \mathcal{A}^t_{j,i}(p^{t,\alpha-1}_j + \omega) - \{\nabla \mathcal{A}^t_{j,i}(p^{t,\alpha-1}_j) - \xi_1 \mathcal{J}^t_j(p^{t,\alpha-1}_j)\}^{\mathrm{T}} \omega + \frac{\xi_2}{2} ||\omega||^2,$$

$$\mathcal{J}^t_j(p^{t,\alpha}_j) = \sum_i \nabla \mathcal{A}^t_{j,i}(p^{t,\alpha}_j) / \sum_i y^t_{i,j},$$

where ξ_1 and ξ_2 are two constant variables. $\mathcal{J}^t_j(\cdot)$ is the sum of gradients among all related edge servers and this process will be performed in the global aggregation step.

Assume that $\mathcal{A}^t_{j,i}(\cdot)$ is λ-Lipschitz continuous and γ-strongly convex[16, 28], then the local convergence of the local model is represented as

$$\mathcal{L}^{t,\alpha}_{j,i}(\omega^{t,\varphi^t_j}_{j,i}) - \mathcal{L}^{t,*}_{j,i} \leqslant \varrho^t_j [\mathcal{L}^{t,\alpha}_{j,i}(\omega^{t,0}_{j,i}) - \mathcal{L}^{t,*}_{j,i}], \tag{1}$$

where $\mathcal{L}^{t,*}_{j,i}$ is the local optimum of the training model. Furthermore, we can set $\omega^{t,0}_{j,i} = 0$ since the initial value can start from 0 for the training model.

Global Aggregation on Parameter Server. After the local updates for all related FL workers, they have to upload the related local model parameter $\omega^{t,\alpha}_{j,i}$ and the related gradients $\nabla \mathcal{A}^t_{j,i}(p^{t,\alpha}_j)$ to the parameter server for aggregation.

$$p^{t,\alpha}_j = p^{t,\alpha-1}_j + \sum_i \{y^t_{i,j} \omega^{t,\alpha}_{j,i}\} / \sum_i y^t_{i,j}.$$

Then, the global average loss of data for the j-th model is

$$\mathcal{G}^t_j(p^{t,\alpha}_j) = \sum_i \frac{y^t_{i,j} |D^t_{j,i}|}{|D^t_j|} \mathcal{A}^t_{j,i}(p^{t,\alpha}_j).$$

Similarly, the global convergence of the global model is defined as

$$\mathcal{G}^t_j(p^{t,\vartheta^t_j}_j) - \mathcal{G}^{t,*}_j \leqslant \varsigma_j [\mathcal{G}^t_j(p^{t,0}_j) - \mathcal{G}^{t,*}_j], \tag{2}$$

where $\mathcal{G}^{t,*}_j$ is the global optimum of the training model.

Finally, from (1) and (2), in order to achieve the desired local convergence rate ϱ^t_j and global convergence rate ς_j, we need to calculate the number of local updates φ^t_j and the number of global iterations ϑ^t_j. From the above observation, we can find that the global convergence rate ς_j for each FL model can be predefined and we have to conduct the local update and global iteration to achieve required global convergence rate. Then we have the following relationship between the convergence rate and the local update as well as global iterations[9, 27].

$$\vartheta^t_j \geqslant \frac{2\lambda^2}{\gamma^2 \xi_1} \ln\left(\frac{1}{\varsigma_j}\right) \frac{1}{1 - \varrho^t_j} \triangleq \vartheta_0 \ln\left(\frac{1}{\varsigma_j}\right) \frac{1}{1 - \varrho^t_j},$$

$$\varphi^t_j \geqslant \frac{2}{(2 - \lambda\delta)\delta\gamma} \log_2\left(\frac{1}{\varrho^t_j}\right) \triangleq \varphi_0 \log_2\left(\frac{1}{\varrho^t_j}\right),$$

where ξ_1 is the constant variable defined in function $\mathcal{L}^{t,\alpha}_{j,i}(\cdot)$, λ is the λ-Lipschitz parameter, and γ is the γ-strongly convex parameter. Both the values of λ and γ are determined by the loss function. ϑ_0 and φ_0 are two constants where $\vartheta_0 = 2\lambda^2/(\gamma^2\xi_1)$ and $\varphi_0 = 2/((2 - \lambda\delta)\delta\gamma)$.

4 Problem: Joint Participant Selection and Learning Optimization

In this section, we first formulate the studied joint optimization problem, and then introduce the cost models used there.

4.1 Problem Formulation

Under the previously introduced multi-model federated learning scenario, we consider how to choose participants for each of the models and how to schedule their local/global updates. Particularly, at each time period t, we need to make the following participant selection and learning scheduling decisions for each model m_j. We denote $x^t_{i,j}$ or $y^t_{i,j}$ as the decision whether to select edge server v_i as a parameter server or an FL worker for the j-th FL model m_j at time t, respectively. Again, we assume that only one PS and κ_j workers are selected for each model, i.e., $\sum_{i=1}^{M} x^t_{i,j} = 1$ and $\sum_{i=1}^{M} y^t_{i,j} = \kappa_j$. We use $\varrho^t_j \in [0,1)$ to represent the maximal local convergence rate of m_j at time t. We will use ϱ^t_j and ς_j to control the number of global iterations and local updates for m_j at time t. Recall that ς_j is given by m_j as a requirement, thus only ϱ^t_j is used for optimization. Overall, $x^t_{i,j}$, $y^t_{i,j}$, and ϱ^t_j are the decision variables of our optimization in each time period t.

We now formulate our participant selection problem in multi-model FL where we need to select the parameter server and workers for each model and achieve the desired local convergence rate. The objective of our problem is to minimize the total cost of all FL models at time t under specific constraints.

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{W} \varpi^t_j & (3)\\
\text{s.t.} \quad & x^t_{i,j}\mu_j\kappa_j \leqslant c^t_i, \quad \forall i,j, & (4)\\
& x^t_{i,j}\chi_j \leqslant f^t_i, \quad \forall i,j, & (5)\\
& y^t_{i,j}\mu_j \leqslant c^t_i, \quad \forall i,j, & (6)\\
& y^t_{i,j}\chi_j \leqslant f^t_i, \quad \forall i,j, & (7)\\
& w_{j,k} y^t_{i,j} z_{i,k} = 1, \quad \forall i,j,k, & (8)\\
& \sum_i x^t_{i,j} = 1, \quad \sum_i y^t_{i,j} = \kappa_j, \quad \forall j, & (9)\\
& \sum_j (x^t_{i,j} + y^t_{i,j}) \leqslant 1, \quad \forall i, & (10)\\
& x^t_{i,j} \in \{0,1\}, y^t_{i,j} \in \{0,1\}, \varrho^t_j \in [0,1). & (11)
\end{aligned}
$$

Here, ϖ^t_j is the total FL cost of the j-th FL model in the t-th time period, which will be defined in Subsection 4.2. Constraints (4) to (7) make sure that the storage and CPU satisfy the FL model requirements. Constraint (8) ensures that the edge server stores the dataset that matches the FL model. Constraint (9) guarantees the number of the parameter server and FL workers of each model is 1 and κ_j, respectively. Constraint (10) ensures that each edge server only trains one FL model and can only play one role at a time. The decision variables and their ranges are given in (11). With nonlinear learning cost, this formulated optimization is a mixed-integer nonlinear program (MINLP) problem challenging to solve directly.

4.2 Cost Models

Our cost models consider four types of cost: edge communication cost, local update cost, global aggregation cost, and PS initialization cost, as defined in the followings, respectively.

Edge Communication Cost. The edge communication cost mainly consists of the FL model downloading and uploading costs. We denote by μ_j the uploaded and downloaded model size for the j-th FL model m_j. When uploading the FL model to the parameter server or downloading the FL model from the parameter server, we use the shortest path in the edge cloud to calculate the communication cost. Let $\rho_j(v_i, v_k)$ be the communication cost of model m_j from edge server v_i to v_k at time t, and it can be calculated by

$$\rho_j(v_i, v_k) = \sum_{e_l \in P^t_{i,k}} \frac{\mu_j}{b^t_l},$$

where $P^t_{i,k}$ is the shortest path connecting v_i to v_k at time t.

For m_j, the total edge communication cost is

$$C^{\text{comm},t}_j = 2 \times \vartheta^t_j \sum_{k=1}^{N} \sum_{i=1}^{N} x^t_{k,j} \times y^t_{i,j} \times \rho_j(v_i, v_k).$$

Here, v_i and v_j are a worker and the PS of m_j, respectively.

Local Update Cost. Let $\psi(\cdot)$ be the function to define CPU cycles to process the sample data $D^t_{j,i}$ used by the j-th FL model and stored in edge server v_i. Therefore all the local update cost for the j-th FL model in the t-th time period is defined as

$$C^{\text{local},t}_j = \vartheta^t_j \times \varphi^t_j \times \sum_{i=1}^{N} y^t_{i,j} \times \frac{\psi(D^t_{j,i})}{f^t_i}.$$

Global Aggregation Cost. Similarly, we use $\psi(\cdot)$ function to define CPU cycles to process the aggregation step for the uploaded FL model.

$$C_j^{\text{global},t} = \vartheta_j^t \times \sum_{i=1}^{N} x_{i,j}^t \times \frac{\psi(\mu_j)}{f_i^t}.$$

Initialization of Parameter Server. The parameter server needs to download the FL model assigned to it at time t unless it has been the parameter server of the same FL model in the last time period. Let $v_{ps}^t(m_j)$ be the PS selected for model m_j at time t. Then the initialization or switching cost of the parameter can be calculated as,

$$C_j^{\text{init},t} = \begin{cases} \eta_j, & \text{if } t = 1 \text{ or } v_{ps}^{t-1}(m_j) = \text{NIL}, \\ 0, & \text{if } v_{ps}^t(m_j) = v_{ps}^{t-1}(m_j), \\ \min\{\eta_j, \rho_j(v_{ps}^{t-1}(m_j), v_{ps}^t(m_j))\}, & \text{otherwise}. \end{cases}$$

If the FL model is the first time to be trained or has not been updated at the last time period, the selected parameter server has to download model m_j with cost η_j. If the parameter server stays the same from the last time period, there is no cost. Otherwise, the new parameter server needs to either download the model or transfer the model from the previous server.

Now, the total cost of the j-th FL model in time t is given by

$$\varpi_j^t = C_j^{\text{comm},t} + C_j^{\text{local},t} + C_j^{\text{global},t} + C_j^{\text{init},t}.$$

5 Our Proposed Methods

In this section, we propose several multi-stage algorithms to attack the challenging optimization problem.

5.1 Three-Stage Method

Recall that the formulated problem in Subsection 4.1 is a mixed-integer nonlinear program (MINLP), which is challenging to solve directly. Now, we decompose our original problem into three sub-problems and attack it via multiple iterations of solving the decomposed sub-problems, as shown in Fig.3.

5.1.1 Three-Stage Decomposition

The main idea is based on a three-stage decomposition. In each stage, we focus on solving only one of the decision variables of $x_{i,j}^t$, $y_{i,j}^t$, and ϱ_j^t when the other two are fixed. We iteratively repeat these three stages until a certain specific condition is satisfied.

Stage 1: Parameter Server Selection. Given a worker selection and a local convergence rate, we aim

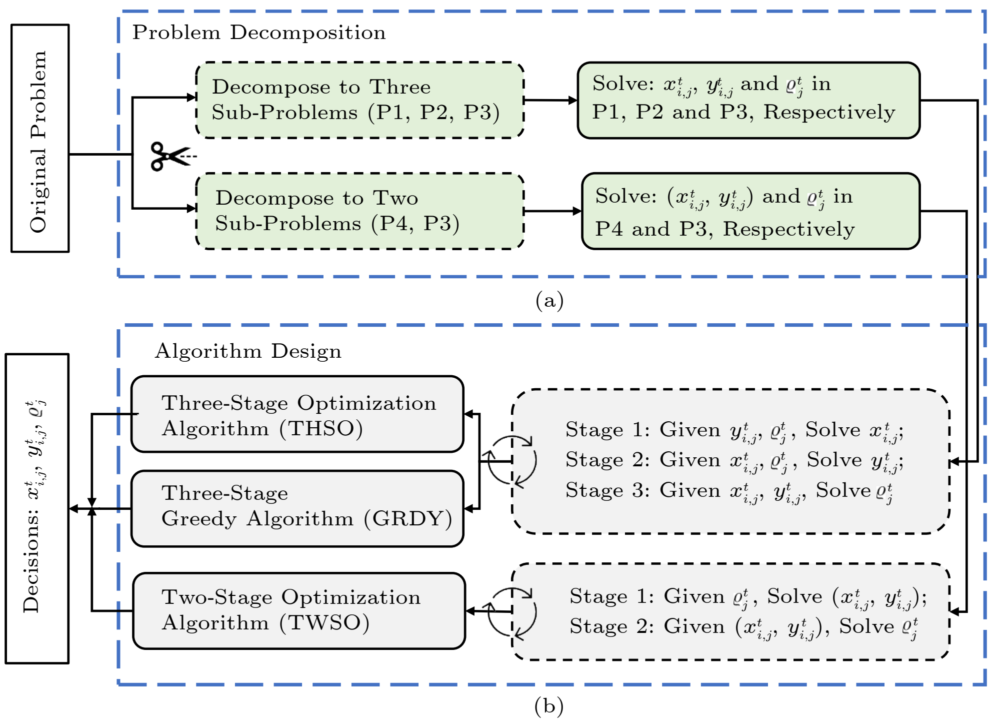

Fig.3. Problem decomposition and design of our proposed multi-stage algorithms. (a) The original problem is decomposed into either two or three subproblems. (b) Three different algorithms (two three-stage ones and a two-stage one) are then proposed to solve these sub-problems iteratively[15].

to find a parameter server for each model to minimize the total cost, i.e.,

$$\begin{aligned} \text{P1}: \quad \min \quad & \sum_{j=1}^{W} \varpi_j^t \\ \text{s.t.} \quad & (4), (5), (9), (11). \end{aligned}$$

Stage 2: *FL Worker Selection*. We take the latest parameter server selection and fixed local convergence rate to select FL workers of each model to minimize the total cost, i.e.,

$$\begin{aligned} \text{P2}: \quad \min \quad & \sum_{j=1}^{W} \varpi_j^t \\ \text{s.t.} \quad & (6)-(11). \end{aligned}$$

Stage 3: *Local Convergence Rate Decision*. With the latest PS and FL worker selections, we can determine the optimal local convergence rate in order to minimize the total cost, i.e.,

$$\begin{aligned} \text{P3}: \quad \min \quad & \sum_{j=1}^{W} \varpi_j^t \\ \text{s.t.} \quad & (11). \end{aligned}$$

5.1.2 *Three-Stage Algorithms*

After we decompose the original problem into three sub-problems, we can solve each sub-problem by using either the linear programming technique or greedy heuristics. The basic idea shared by these methods is as follows. First, we randomly generate FL worker selection decision $y_{i,j}^{t,0}$ and the local convergence rate $\varrho_j^{t,0}$, and then solve the optimization problem P1 to get parameter server selection decision $x_{i,j}^{t,1}$. Second, given the local convergence rate $\varrho_j^{t,0}$ and the latest parameter server selection decision $x_{i,j}^{t,1}$, we solve P2 to get FL worker selection decision $y_{i,j}^{t,1}$. Last, based on the latest $x_{i,j}^{t,1}$ and $y_{i,j}^{t,1}$, we solve P3 to achieve the desired local convergence rate $\varrho_j^{t,1}$. This process will be repeated until it satisfies a specific condition (either no further improvement of the objective value of the optimization or reaching the maximal iteration number).

Algorithm 1 in Fig.4(a) shows the three-stage optimization method using the linear programming technique. Here, we leverage an optimization solver (PuLP)[29] to solve each sub-problem for its convenience.

Algorithm 2 in Fig.4(b) shows a three-stage greedy algorithm where greedy heuristic methods are used to solve the three sub-problems. 1) For stage 1, given a fixed worker selection and local convergence rate, we select a parameter server for each model with minimal cost. 2) For stage 2, we calculate the total cost of each potential edge server for each FL model and then sort the edge server list in ascending order of the cost. We greedily select the top κ_j edge servers as FL workers for each model. 3) In the last stage, we greedily decrease the local convergence rate in a specific threshold to get the minimal total cost until it reaches the global convergence rate. We repeat the above steps until the ending condition is met.

Note that during the first two stages of Algorithm 2, we need to select the PS or workers for all models in a certain order. Obviously, the processing order of each model may affect the final performance. By default, we simply process them in a first come first serve mode, i.e., we first find the solution for the model that arrives earlier. Due to the heterogeneity of edge servers in the real edge cloud, some edge servers may have more sufficient resources (storage and computing capacity) while the others do not. In such a resource-limited scenario, serving the more complex FL model first may reduce the total completion cost of FL of all models. Therefore, we also introduce a variation greedy method in which the FL models are sorted based on their model sizes and we process models based on a larger model first in both the first and second stages of Algorithm 2. In this variation, the more complex FL model will first have more chance to select more high-performance workers leading to a lower total cost. In our experiments (Section 6), we have evaluated the impact of these two different processing orders. In addition, other ordering methods can also be applied to our proposed algorithm (Algorithm 2), such as choosing the model that requests more resources first.

5.2 Two-Stage Method

We can also combine the first two stages since both are with integer variables. Then the optimization can be solved via a two-stage decomposition. Here, we separate the integer variables $(x_{i,j}^t, y_{i,j}^t)$ and the continuous variable ϱ_j^t into two sub-problems, as shown in Fig.3.

Stage 1: *Parameter Server and Worker Selection*. Given the last local convergence rate, we want to find an optimal decision for selecting the parameter server and workers, i.e.,

Algorithm 1. Three-Stage Optimization Algorithm

Initialize max_itr, max_occur, $bound\_val$
Generate a random initial FL worker selection decision $y_{i,j}^{t,0}$ and local convergence rate $\varrho_j^{t,0}$
$\iota = 1$ and $count\_num = 0$;
repeat
  Stage 1: Calculate $x_{i,j}^{t,\iota}$ by solving P1 with fixed $y_{i,j}^{t,\iota-1}$ and $\varrho_j^{t,\iota-1}$
  Stage 2: Calculate $y_{i,j}^{t,\iota}$ by solving P2 with fixed $x_{i,j}^{t,\iota}$ and $\varrho_j^{t,\iota-1}$
  Stage 3: Calculate $\varrho_j^{t,\iota}$ by solving P3 with fixed $x_{i,j}^{t,\iota}$ and $y_{i,j}^{t,\iota}$. Let obj_val be the achieved objective value (total learning cost of all FL models)
  if $obj\_val > bound\_val$ **then**
    $bound\_val$ = obj_val; $count\_num$ = 1
    $x_{j,i}^t = x_{j,i}^{t,\iota}$; $y_{k,i}^t = y_{k,i}^{t,\iota}$; $\varrho_j^t = \varrho_j^{t,\iota}$
  else if obj_val = $bound\_val$ **then**
    $count\_num = count\_num + 1$
  end if
  $\iota = \iota + 1$
until $count\_num = max\_occur$ or $\iota = max\_itr$
return $x_{i,j}^t$, $y_{i,j}^t$ and ϱ_j^t

(a)

Algorithm 2. Three-Stage Greedy Algorithm

Initialize max_itr, max_occur, $bound\_val$
Generate a random initial FL worker selection decision $y_{i,j}^{t,0}$ and local convergence rate $\varrho_j^{t,0}$
$\iota = 1$ and $count\_num = 0$;
repeat
  Stage 1: Pick the PS $x_{i,j}^{t,\iota}$ for each FL model with minimal total cost with fixed $y_{i,j}^{t,\iota-1}$ and $\varrho_j^{t,\iota-1}$
  Stage 2: Calculate the total cost of each potential edge server for each FL model, and sort the list in ascending order and greedily select the first κ_j edge servers to get $y_{i,j}^{t,\iota}$ with the latest $x_{i,j}^{t,\iota}$ and fixed $\varrho_j^{t,\iota-1}$
  Stage 3: Calculate $\varrho_j^{t,\iota}$ by greedily decreasing the local convergence rate to get a minimal total cost with latest $x_{i,j}^{t,\iota}$ and $y_{i,j}^{t,\iota}$. Let obj_val be the achieved objective value (total learning cost of all FL models)
  if $obj\_val > bound\_val$ **then**
    $bound\_val$ = obj_val; $count\_num$ = 1
    $x_{j,i}^t = x_{j,i}^{t,\iota}$; $y_{k,i}^t = y_{k,i}^{t,\iota}$; $\varrho_j^t = \varrho_j^{t,\iota}$
  else if obj_val = $bound\_val$ **then**
    $count\_num = count\_num + 1$
  end if
  $\iota = \iota + 1$
until $count\_num = max\_occur$ or $\iota = max\_itr$
return $x_{i,j}^t$, $y_{i,j}^t$ and ϱ_j^t

(b)

Algorithm 3. Two-Stage Optimization Algorithm

Initialize max_itr, max_occur, $bound\_val$
Generate a random initial local convergence rate $\varrho_j^{t,0}$
$\iota = 1$ and $count\_num = 0$;
repeat
  Stage 1: Calculate $x_{i,j}^{t,\iota}$ and $y_{i,j}^{t,\iota}$ by solving P4 with fixed $\varrho_j^{t,\iota-1}$
  Stage 2: Calculate $\varrho_j^{t,\iota}$ by solving P3 with latest $x_{i,j}^{t,\iota}$ and $y_{i,j}^{t,\iota}$. Let obj_val be the achieved objective value (total learning cost of all FL models)
  if $obj\_val > bound\_val$ **then**
    $bound\_val$ = obj_val; $count\_num$ = 1
    $x_{j,i}^t = x_{j,i}^{t,\iota}$; $y_{k,i}^t = y_{k,i}^{t,\iota}$; $\varrho_j^t = \varrho_j^{t,\iota}$
  else if obj_val = $bound\_val$ **then**
    $count\_num = count\_num + 1$
  end if
  $\iota = \iota + 1$
until $count\_num = max\_occur$ or $\iota = max\_itr$
return $x_{i,j}^t$, $y_{i,j}^t$ and ϱ_j^t

(c)

Fig.4. Proposed algorithms[15]. (a) Three-stage optimization (THSO). (b) Three-stage greedy (GRDY). (c) Two-stage optimization (TWSO).

$$\begin{aligned} \text{P4}: \quad & \min \quad \sum_{j=1}^{W} \varpi_j^t \\ & \text{s.t.} \quad (4)-(11). \end{aligned}$$

Stage 2: *Local Convergence Rate Decision*. This is the same with the third sub-problem P3 in three-stage methods.

Here, we use an optimization solver (GEKKO)[30] to solve the sub-problem P4 since it is a non-linear problem with two integer variables. For P3, we still use the PuLP solver. The detail of the two-stage method is given by Algorithm 3 in Fig.4(c).

5.3 Time Complexity

We now analyze the time complexity of each of the proposed algorithms. Here, we assume that the time taken to solve P1, P2, P3, and P4 with N

servers and W models is $T_1(N,W)$, $T_2(N,W)$, $T_3(N,W)$, and $T_4(N,W)$, respectively. Given a decision of $x^t_{i,j}$, $y^t_{i,j}$, and ϱ^t_j, we can calculate the total learning cost with $T_{\text{cost}}(N,W)$. Let ϵ be the step length of reducing the convergence rate in the stage 3 of Algorithm 2. Then it is easy to prove the following theorem regarding the time complexity of all proposed algorithms.

Theorem 1. *The time complexity of Algorithms 1–3 is bounded by* $O((T_1+T_2+T_3)\times max\_itr)$, $O((N+(1/\epsilon))\times T_{\text{cost}}\times W\times max\_itr)$, *and* $O((T_4+T_3)\times max\_itr)$, *respectively.*

Note that in Algorithms 2, the time complexity of stage 1 and stage 2 is bounded by $O(N\times T_{\text{cost}}\times W)$ and $O((1/\epsilon)\times T_{\text{cost}}\times W)$, respectively.

6 Performance Evaluation

In this section, we present our experimental setup and evaluate the performance of our proposed methods via simulations.

6.1 Environmental Setup

Edge Cloud. In our edge computing environment, we adopt different random topologies consisting of $20-40$ edge servers where the distribution of servers is based on the real-world EUA-Dataset[31]. This dataset is widely used in edge computing and contains the geographical locations of 125 cellular base stations in the Melbourne central business district area. Fig.5 illustrates one example of topology used in our simulations. In each simulation, a certain number of edge servers are randomly selected from the dataset. Each edge server has a maximal storage capacity c_i, CPU frequency f_i, and link bandwidth b_j in the range of 512 GB–1 024 GB, 2 GHz–5 GHz, and 512 Mbps–1 024 Mbps, respectively. We consider $O=5$ different data types (e.g., image, audio, and text) where the size $S_{i,k}$ is in the range of 1 GB–3 GB. Each type of data has been distributed in different edge servers and one edge server may store more than one type of data. Furthermore, the total number of time periods T is set to 30.

Federated Learning Models. To verify the performance of the federated learning process, we conduct a set of federated learning experiments. We assume that

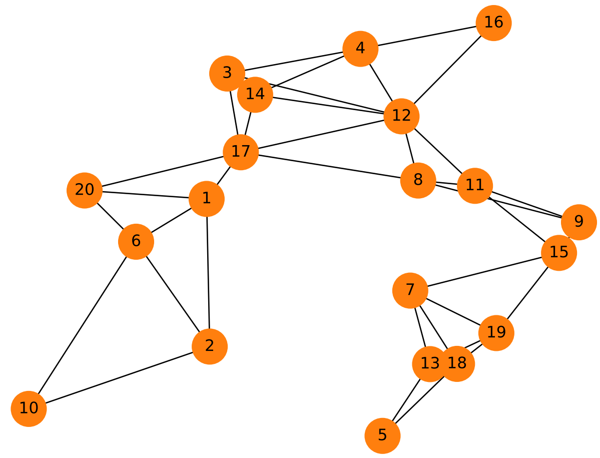

Fig.5. Example of edge cloud topology with 20 edge servers generated based on the real-world EUA-Dataset[31].

there are W different FL tasks (vision, audio, text, or data) running in our environment simultaneously. The number of FL workers κ_j required by each model is in the range of 3–7. Each FL task has a specific model size μ_j, CPU requirement χ_j, and download cost η_j in the range of 10 MB–100 MB, 1 GHz–3 GHz, and 1–5, respectively. The global convergence requirement and the two constant variables are set based on [9]: $\varsigma_j=0.001$, $\vartheta_0=15$, and $\varphi_0=4$. Three classical datasets in scikit-learn 1.0.2[32] are used to train linear regression (LR) models: the California Housing dataset, the 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 datasets are used for these FL tasks: Fashion-MNIST (FMNIST)[33], Speech_Commands[34], and AG_NEWS[35]. Each of them is trained with a CNN model. We assign random data samples of these three datasets to clients in such a way that each client has a different amount of training and testing data. The Python library PyTorch (v1.10) is used to build the model. All experiments are tested on a Linux workstation including 16 CPU cores and 512 GB of RAM, and 4x NVIDIA Tesla V100 GPUs interconnected with NVlink2. Detailed parameters of both edge cloud and FL models are listed in Table 2.

Baselines and Metrics. We compare our proposed algorithms (three-stage optimization THSO, three-stage greedy GRDY, and two-stage optimization TWSO) with four competitive methods[②].

[②]Since our studied joint optimization is an MINLP problem, it is challenging to obtain the optimal solution even when the problem is in a small scale. Therefore, we focus on comparison with existing simple heuristics and participant selection methods.

Table 2. Detailed Parameter Setting in Our Experiments

	Parameter	Value or Range
Edge cloud parameter	Number of edge servers N	20−40
	v_i's storage capacity c_i (GB)	512−1 024
	v_i's CPU frequency f_i (GHz)	2−5
	e_i's link bandwidth b_i (Mbps)	512−1 024
	Number of different datasets O	5
	Each dataset size $\lvert S_{i,k}\rvert$ (GB)	1−3
Federated learning parameter	Number of time periods T	30
	Number of FL models W	1−5
	Number of m_j's FL workers κ_j	1−7
	m_j's model size μ_j (MB)	10−100
	m_j's CPU requirement χ_j (GHz)	1−3
	m_j's downloading cost η_j	1−5
	m_j's global convergence requirement ς_j	0.001−0.1
	Constant FL variables ϑ_0 and φ_0	15, 4

• *ROUND*[9]. It selects the FL workers and the local convergence rate for each model based on a randomized rounding method[9]. Since it does not consider the PS selection, we use a random choice for PS at the beginning.

• *RAND*. It randomly generates the parameter server selection, the FL worker selection decision, and the local convergence rate under certain constraints.

• *DATA*[23]. It selects the FL workers based on the fraction of data at the servers and prefers the server with more data. Since it ignores the PS and local convergence rate selection, we randomly determine them.

• *LOCAL*[24]. It selects its top workers that will complete the local training first (based on estimation). Again random decisions are used for the PS and local rate.

The main metrics used for evaluation are the average total FL learning cost and the average accuracy (or the average training loss and the average R2 score for LR) of FL models.

6.2 Evaluation Results

Via extensive simulations, we evaluate the performance of our methods mainly focusing on their cost performances.

6.2.1 Performance Comparison—Total Learning Cost

We first investigate different algorithms with a different number of edge servers and global convergence rates. We consider three FL models for three different types of tasks (i.e., image classification, speech recognition, and text classification) to be trained simultaneously, where each FL model has requested five FL workers. Figs.6(a) and 6(b) show the results of two groups of simulations. In the first group, we set the number of edge servers to be from 20 to 40, while fixing the global convergence rate at 0.001. In the second group, we change the global convergence rate from 0.001 to 1.1 with 30 edge servers. We have the following observations.

First, clearly for both sets of simulations, our proposed three algorithms (TWSO, THSO, and GRDY) have better performance than the other four benchmarks in terms of average total learning cost. Having better performances than ROUND (which focuses on worker selection and learning rate optimization) confirms the advance of our methods by considering the PS selection in the joint optimization. The better performances of our methods and ROUND compared with DATA and LOCAL (which only focus on the worker selection) show the advantage of joint optimization. In all simulations, RAND has the worst performance since it does not take any optimization.

Second, as shown in Fig.6(a), the average total cost of every algorithm decreases first and increases again as the number of edge servers increases. Initially, with more edge servers, all algorithms have better chances to find a good solution to minimize the total cost of all FL models. On the other hand, the further larger topology with more servers may also begin to increase the average total cost due to larger transmission costs from workers to PS.

Third, as shown in Fig.6(b), as the global convergence rate increases, the average total cost decreases. This is reasonable since the larger global convergence rate requests less local training or global update, which leads to lower total learning costs.

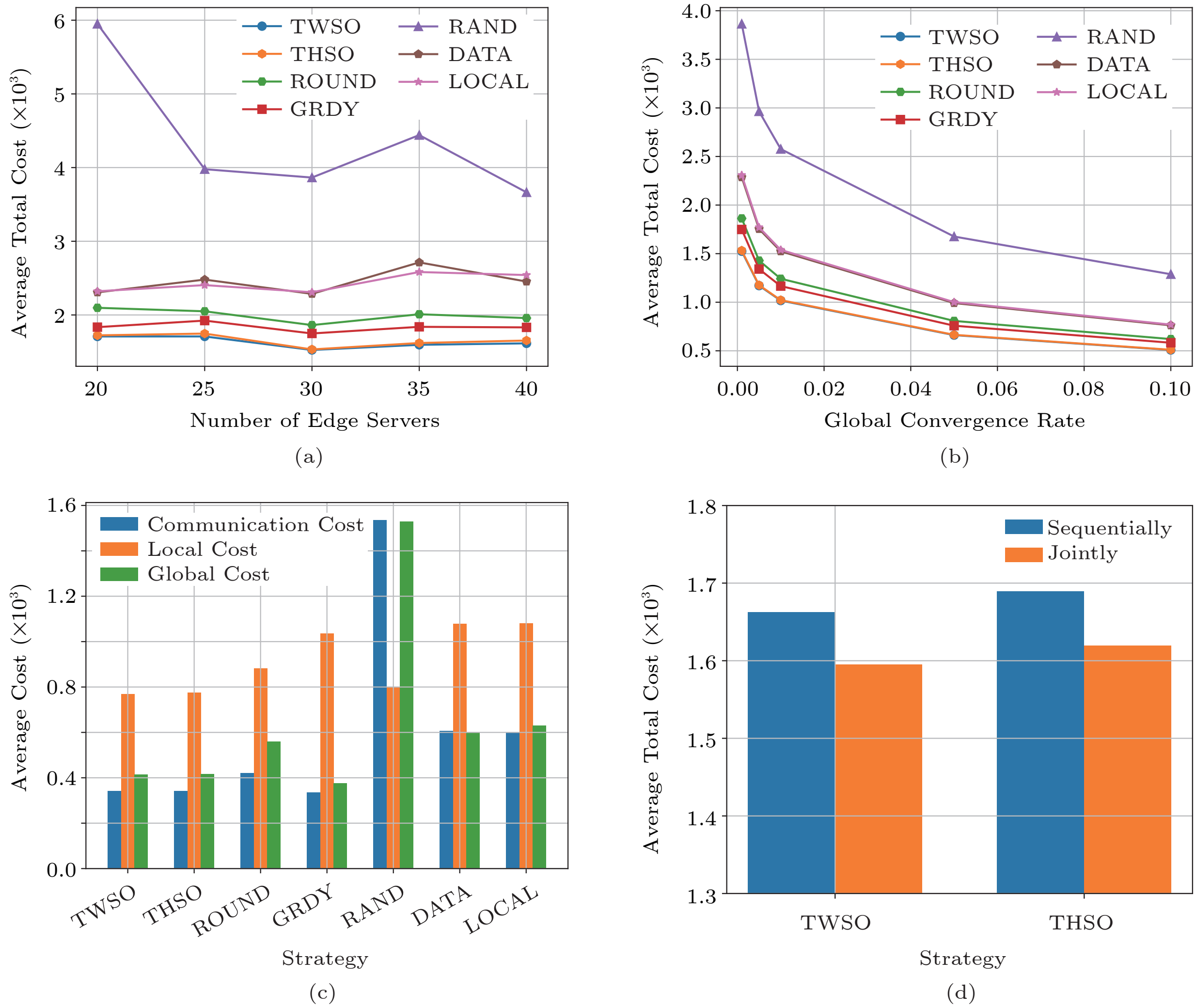

Fig.6. Performance comparison with (a) different number of edge servers, (b) different global convergence rates, (c) detailed costs of different methods, and (d) single model vs multiple models[15].

Fig.6(c) also plots the detailed costs of different methods when 30 edge servers are considered, and the global convergence rate is 0.001. It shows that the local cost dominates the total cost, and consequently, GRDY has a higher total cost than TWSO and THSO as seen in Fig.6(a).

We also evaluate the effects of joint optimization over multi-models compared with the separative optimization with only a single model. In the latter case, we still use TWSO and THSO but force them only on a single FL model at a time, and thus sequentially choose the decision for each model. Again we train three FL models when 30 edge servers are considered and the global convergence rate is 0.001. Fig.6(d) shows the comparison of determining the choices for three FL models jointly or sequentially with TWSO and THSO. We can clearly see the lower total cost when we jointly optimize the decisions. This confirms the effectiveness of jointly determining the selection decision for multiple FL models rather than sequentially determining the decision for every single model.

6.2.2 *Impact of FL Model Number*

Next, we look into the impact of different numbers of FL models. We simultaneously run one to five FL models. The number of edge servers and the number of FL workers are set to 30 and 5, respectively. The global convergence rate is also set to 0.001. As shown in Fig.7(a), the more FL models, the more the average total learning cost. Our proposed algorithms still perform better than the other four methods. TWSO and THSO still enjoy a little performance improvement against GRDY. We also plot the detail of three types of costs (i.e., communication cost, local cost, and global cost) in Fig.7(b), Fig.7(c), and

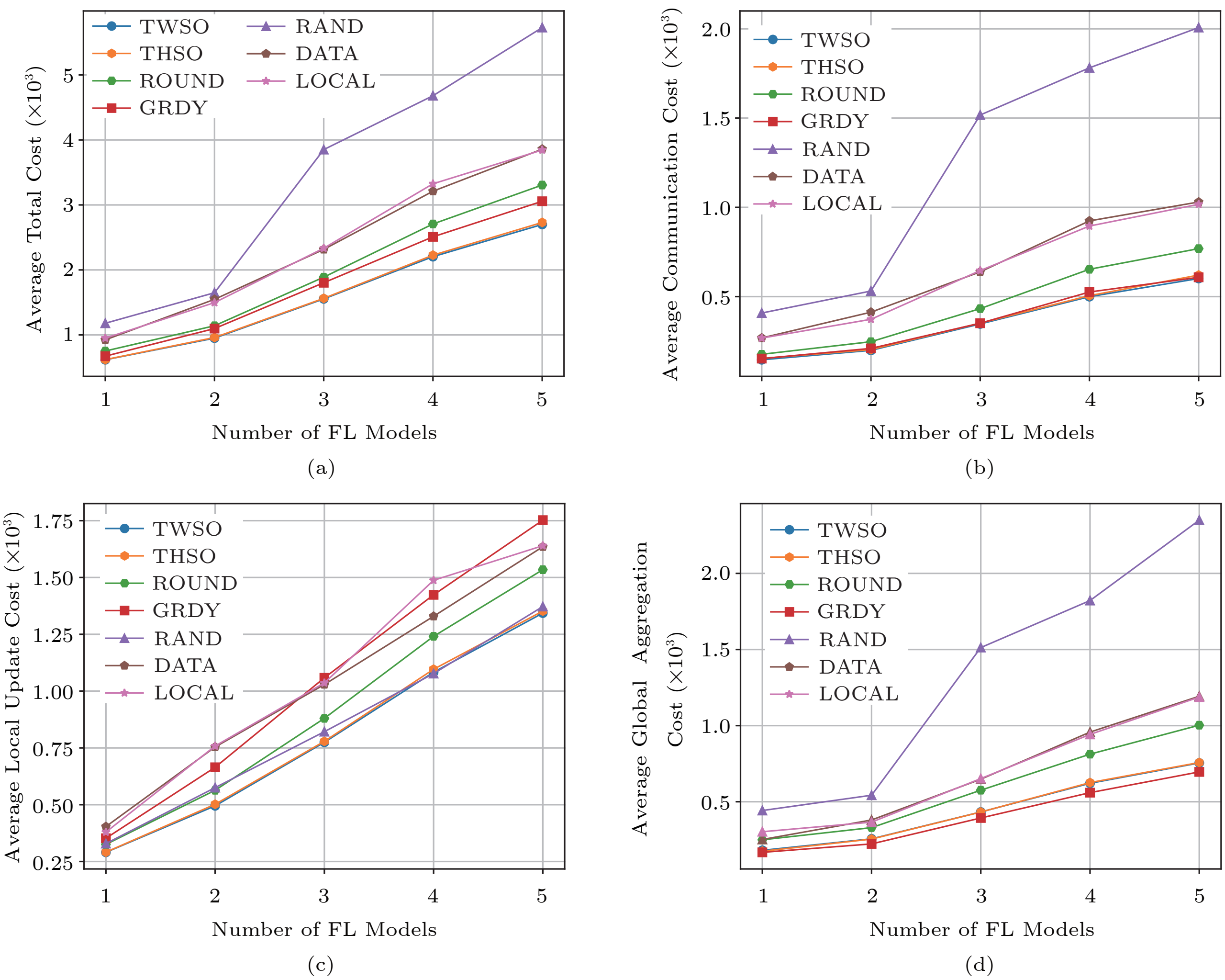

Fig.7. Impact of the number of FL models on costs[15]. (a) Average total cost. (b) Average communication cost. (c) Average local update cost. (d) Average global aggregation cost.

Fig.7(d) respectively. We can observe that the communication cost of GRDY is similar to those of TWSO and THSO. However, GRDY has the highest local cost and the lowest global cost compared with the other algorithms. That is because GRDY greedily runs more local training so that the number of global updates can be reduced while satisfying the expected global convergence rate.

6.2.3 Impact of FL Worker Number

We further investigate the impact of a different number of FL workers. We consider 30 edge servers and train three FL models while the global convergence rate is 0.001 as well. Results are reported in Fig.8, which are similar to those with a different number of FL models. First, the average total cost of all algorithms increases as the number of FL workers increases since the more FL workers, the more resource-consuming. Second, the proposed algorithms have better performance than ROUND, RAND, DATA, and LOCAL as shown in Fig.8(a). Last, GRDY has the highest local cost while having the lower communication cost and global cost compared with other strategies as shown in Figs.8(b)–8(d).

6.2.4 Impact of Model Processing Order in GRDY

Remember that in GRDY (Algorithms 2) we need to select the PS and workers for each model following certain processing order among FL models. We now study the impact of different processing orders in GRDY. We test on two specific processing orders: the default one with First-in-First-Serve (GRDY) and the variation in which priority is given to the model with a larger size (GRDY-Max). The experiments run un-

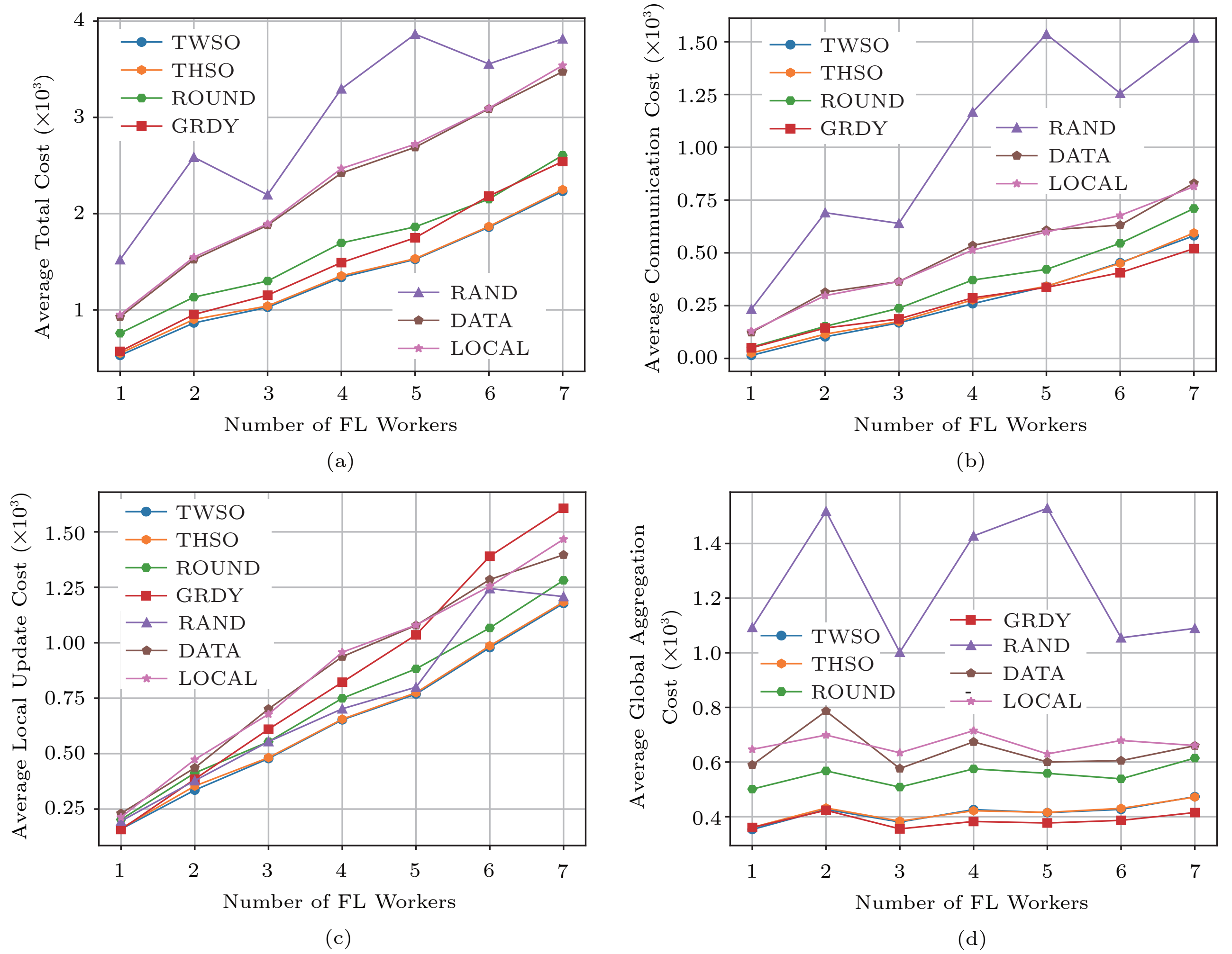

Fig.8. Impact of the number of FL workers on costs[15]. (a) Average total cost. (b) Average communication cost. (c) Average local update cost. (d) Average global aggregation cost.

der the edge cloud with 30 edge servers that have limited resources and significant differences. We run 20 different cases in each different number of maximum iterations, and Fig.9 shows the experimental result. First, as the number of maximum iterations increases, the total cost of both two greedy algorithms decreases since they have more chances to find a better solution with a lower cost. However, the improvement becomes smaller when the maximum iteration further increases. Second, under the resource-limited scenario, GRDY-Max performs better than GRDY in almost all cases. This result confirms the necessity and superiority of selecting an optimal processing order in different edge scenarios. In addition, we need to select an appropriate maximum iteration to control the convergence speed of our greedy algorithms.

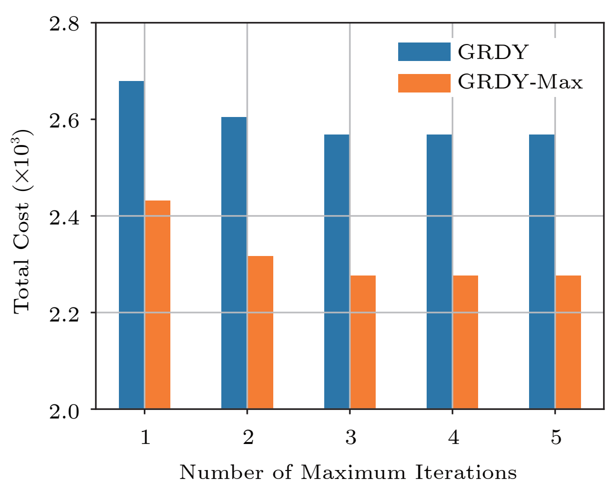

Fig.9. Total cost comparison of two different processing orders of FL models in GRDY and its variation GRDY-Max.

6.2.5 FL Training Loss and Accuracy

Fig.10 shows the training loss of our method in real-world federated learning experiments over LR datasets. We introduce the R2 score metric to evaluate the performance of LR (Linear Regression) model (convex) training. The R2 score is the proportion of

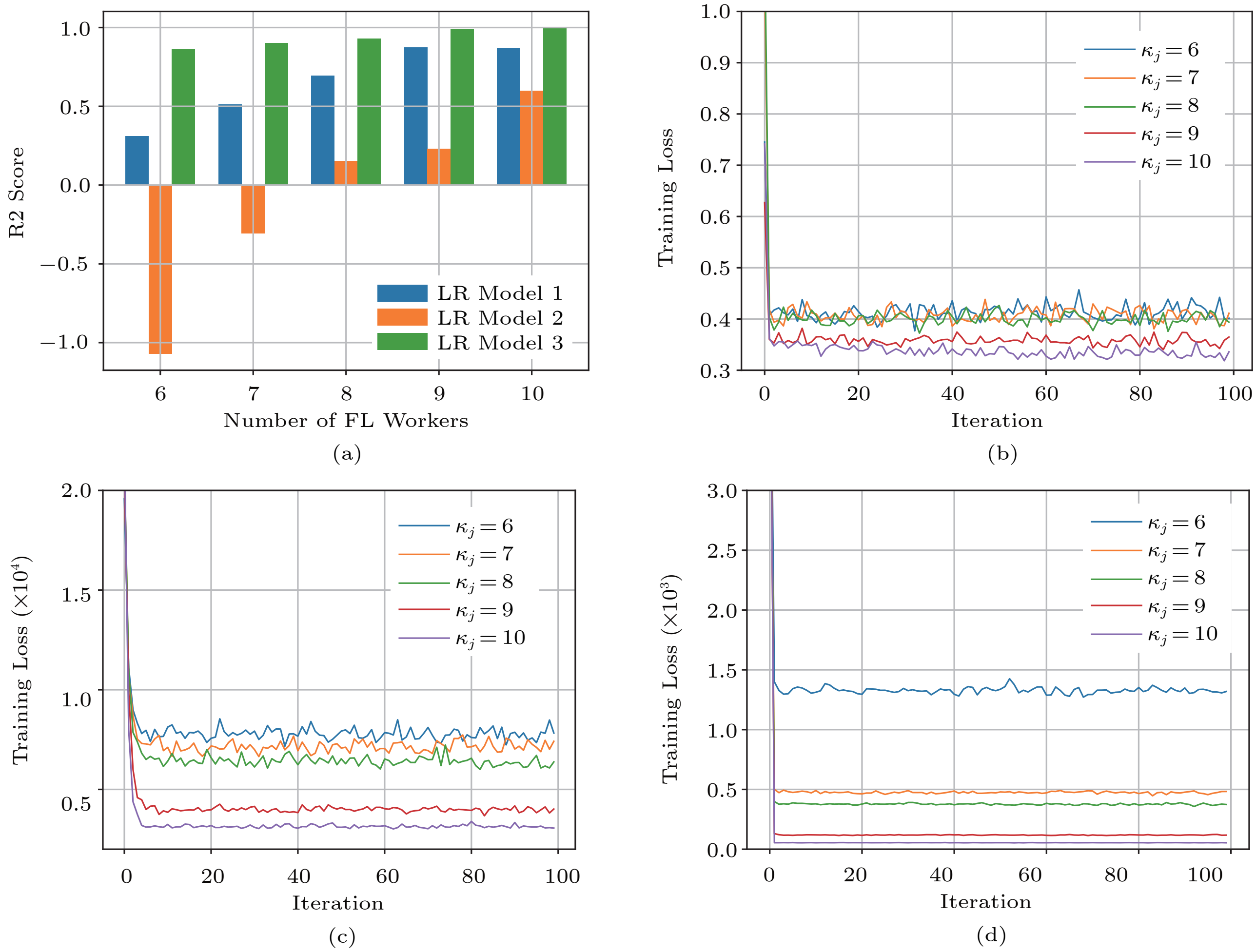

Fig.10. Training loss with LR models/tasks and the impact of the number of FL workers. (a) R2 score of three LR models. (b) Training loss of Linear Regression over the California Housing dataset. (c) Training loss of Linear Regression over the diabetes dataset. (d) Training loss of Linear Regression over a randomly generated dataset.

the variance in the dependent variable that is predictable from the independent variable(s). In this set of experiments, we concurrently train three LR models with three different datasets. Each dataset is split into 10 edge servers unequally (i.e., non-IID setting), and the number of global training rounds is 100. We can see from Figs.10(b)–10(d), the training loss decreases as the number of workers (κ_j) increases for each model. Fig.10(a) shows the R2 score of all LR models. Obviously, with more workers, the R2 scores of all models increase, 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.11 also reports the learning accuracy of our method on more complex FL tasks with different numbers of workers (due to space limitation, we only show the one from THSO). Here, the datasets of three FL models (image classification, speech recognition, and text classification) are split into 30 partitions and the number of the global update is set to 300. Fig.11(a) shows the training accuracy of all three FL models increases with the increasing number of iterations. Figs.11(b)–11(d) show the detailed training accuracy of three different models with different numbers of FL workers. We can observe that with more FL workers, the training accuracy of all models can reach a higher value. However, when comparing the result in Fig.8, the more FL workers, the more total cost consumed. Hence, there is a trade-off between the training accuracy and the total cost. Another interesting observation is that for FMNIST and Speech_Command, the accuracy increases with more FL workers, but for AG_News, the accuracy is similar or the difference is very minimal. This may be due to the simplicity of the AG_News learning task. In

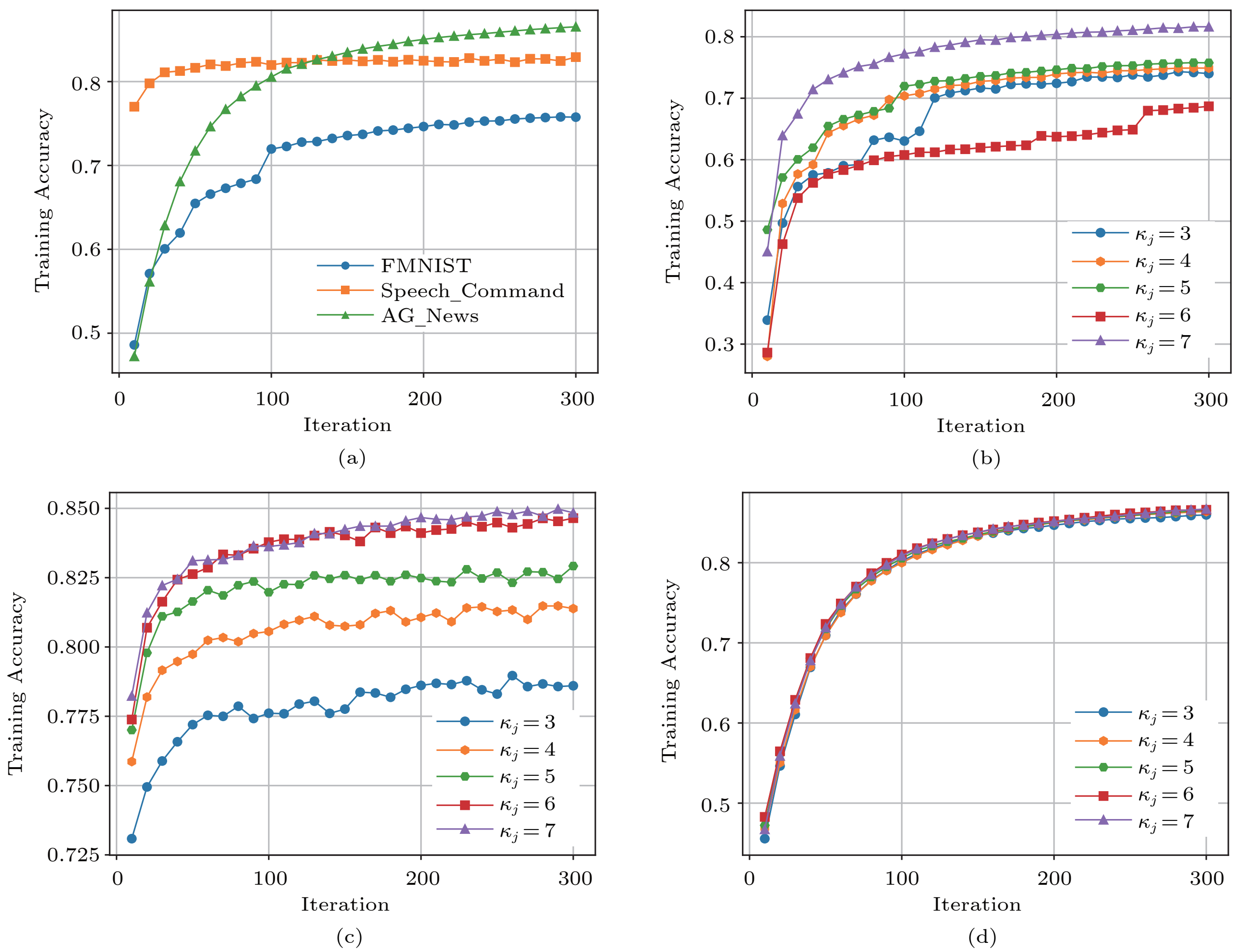

Fig.11. Training accuracy with three FL tasks and the impact of FL workers[15]. (a) Accuracy of three FL models with 30 edge servers and 5 workers. (b) Accuracy of image classification over FMNIST. (c) Accuracy of voice recognition in Speech_Command. (d) Accuracy of text generation in AG_News.

summary, one needs to consider the trade-off between the training accuracy and the total cost. If you need more FL workers, it will incur more total cost but get higher training accuracy, and vice versa.

7 Conclusions

In this paper, we mainly focused on multi-model FL over an edge cloud and carefully selected participants (both PS and workers) for each model by considering the resource limitation and heterogeneity of edge servers as well as different data distributions. We formulated a joint participant selection and learning optimization problem to minimize the total FL cost of multiple models while ensuring their convergence performance. We proposed three different algorithms to decompose the original problem into multi-stages so that each stage can be solved by an optimization solver or a greedy algorithm. Extensive simulations with real FL experiments showed that our proposed algorithms outperform similar existing solutions. In the future, we plan to further investigate: 1) reinforcement learning based solutions for similar optimization problems in a more dynamic edge system, 2) the extension of proposed multi-stage methods over multiple time-scales, where learning schedule and participant selection can be optimized at fast and slow timescales separately, similar to [22], 3) new quantum-assisted methods for similar optimization problems[36], 4) new joint optimization problems where different FL models can choose different FL structures (such as DFL[10] or HFL[5, 8, 11]), and 5) similar joint optimization problems but in a more complex edge system with multiple edge operators.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] McMahan B, Moore E, Ramage D, Hampson S, Arcas B A Y. Communication-efficient learning of deep networks from decentralized data. In *Proc. the 20th International Conference on Artificial Intelligence and Statistics*, Apr. 2017, pp.1273–1282. DOI: 10.48550/arXiv.1602.05629.

[2] Ji S X, Jiang W Q, Walid A, Li X. Dynamic sampling and selective masking for communication-efficient federated learning. *IEEE Intelligent Systems*, 2022, 37(2): 27–34. DOI: 10.1109/MIS.2021.3114610.

[3] Sattler F, Wiedemann S, Muller K R, Samek W. Robust and communication-efficient federated learning from non-I. I. D. data. *IEEE Trans. Neural Networks and Learning Systems*, 2019, 31(9): 3400–3413. DOI: 10.1109/TNNLS.2019.2944481.

[4] Lim W Y B, Luong N C, Hoang D T, Jiao Y T, Liang Y C, Yang Q, Niyato D, Miao C Y. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2020, 22(3): 2031–2063. DOI: 10.1109/COMST.2020.2986024.

[5] Liu L M, Zhang J, Song S H, Letaief K B. Client-edge-cloud hierarchical federated learning. In *Proc. the 2020 IEEE International Conference on Communications*, Jun. 2020. DOI: 10.1109/ICC40277.2020.9148862.

[6] Wang S Q, Tuor T, Salonidis T, Leung K K, Makaya C, He T, Chan K. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 2019, 37(6): 1205–1221. DOI: 10.1109/JSAC.2019.2904348.

[7] Nishio T, Yonetani R. Client selection for federated learning with heterogeneous resources in mobile edge. In *Proc. the 2019 IEEE International Conference on Communications*, May 2019. DOI: 10.1109/ICC.2019.8761315.

[8] Luo S Q, Chen X, Wu Q, Zhou Z, Yu S. HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wireless Communications*, 2020, 19(10): 6535–6548. DOI: 10.1109/TWC.2020.3003744.

[9] Jin Y B, Jiao L, Qian Z Z, Zhang S, Lu S L. Learning for learning: Predictive online control of federated learning with edge provisioning. In *Proc. the 2021 IEEE Conference on Computer Communications*, May 2021. DOI: 10.1109/INFOCOM42981.2021.9488733.

[10] Meng Z Y, Xu H L, Chen M, Xu Y, Zhao Y M, Qiao C M. Learning-driven decentralized machine learning in resource-constrained wireless edge computing. In *Proc. the 2021 IEEE Conference on Computer Communications*, May 2021. DOI: 10.1109/INFOCOM42981.2021.9488817.

[11] Wang Z Y, Xu H L, Liu J C, Huang H, Qiao C M, Zhao Y M. Resource-efficient federated learning with hierarchical aggregation in edge computing. In *Proc. the 2021 IEEE Conference on Computer Communications*, May 2021. DOI: 10.1109/INFOCOM42981.2021.9488756.

[12] Wei X L, Liu J Y, Shi X H, Wang Y. Participant selection for hierarchical federated learning in edge clouds. In *Proc. the 2022 IEEE International Conference on Networking, Architecture and Storage*, Oct. 2022. DOI: 10.1109/NAS55553.2022.9925313.

[13] Liu J, Wei X, Liu X, Gao H, Wang Y. Group-based hierarchical federated learning: Convergence, group formation, and sampling. In *Proc. International Conference on Parallel Processing*, Aug. 2023. DOI: 10.1145/3605573.3605584.

[14] Nguyen M N H, Tran N H, Tun Y K, Han Z, Hong C S. Toward multiple federated learning services resource sharing in mobile edge networks. *IEEE Trans. Mobile Computing*, 2023, 22(1): 541–555. DOI: 10.1109/TMC.2021.3085979.

[15] Wei X L, Liu J Y, Wang Y. Joint participant selection and learning scheduling for multi-model federated edge learning. In *Proc. the 19th International Conference on Mobile Ad Hoc and Smart Systems*, Oct. 2022, pp.537–545. DOI: 10.1109/MASS56207.2022.00081.

[16] Yang Z H, Chen M Z, Saad W, Hong C S, Shikh-Bahaei M. Energy efficient federated learning over wireless communication networks. *IEEE Trans. Wireless Communications*, 2021, 20(3): 1935–1949. DOI: 10.1109/TWC.2020.3037554.

[17] Li L, Shi D, Hou R H, Li H, Pan M, Han Z. To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices. In *Proc. the 2021 IEEE Conference on Computer Communications*, May 2021. DOI: 10.1109/INFOCOM42981.2021.9488839.

[18] Wang J Y, Pan J L, Esposito F, Calyam P, Yang Z C, Mohapatra P. Edge cloud offloading algorithms: Issues, methods, and perspectives. *ACM Computing Surveys*, 2020, 52(1): Article No. 2. DOI: 10.1145/3284387.

[19] Li T, Qiu Z J, Cao L J, Cheng D Z, Wang W C, Shi X H, Wang Y. Privacy-preserving participant grouping for mobile social sensing over edge clouds. *IEEE Trans. Network Science and Engineering*, 2021, 8(2): 865–880. DOI: 10.1109/TNSE.2020.3020159.

[20] Tan H S, Han Z H, Li X Y, Lau F C M. Online job dispatching and scheduling in edge-clouds. In *Proc. the 2017 IEEE Conference on Computer Communications*, May 2017. DOI: 10.1109/INFOCOM.2017.8057116.

[21] Yang S, Li F, Trajanovski S, Chen X, Wang Y, Fu X M. Delay-aware virtual network function placement and routing in edge clouds. *IEEE Trans. Mobile Computing*, 2021, 20(2): 445–459. DOI: 10.1109/TMC.2019.2942306.

[22] Wei X L, Rahman A B M M, Cheng D Z, Wang Y. Joint optimization across timescales: Resource placement and task dispatching in edge clouds. *IEEE Trans. Cloud Computing*, 2023, 11(1): 730–744. DOI: 10.1109/TCC.2021.3113605.

[23] Cho Y J, Wang J Y, Joshi G. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. arXiv: 2010.01243, 2020. https://arxiv.org/abs/2010.01243, Jul. 2023.

[24] Li X, Huang K X, Yang W H, Wang S S, Zhang Z H. On the convergence of FedAvg on non-IID data. arXiv: 1907.02189, 2019. https://arxiv.org/abs/1907.02189, Jul.

2023.

[25] Li Y Q, Li F, Chen L X, Zhu L H, Zhou P, Wang Y. Power of redundancy: Surplus client scheduling for federated learning against user uncertainties. *IEEE Trans. Mobile Computing*, 2023, 22(9): 5449–5462. DOI: 10.1109/TMC.2022.3178167.

[26] Tran N H, Bao W, Zomaya A, Nguyen M N H, Hong C S. Federated learning over wireless networks: Optimization model design and analysis. In *Proc. the 2019 IEEE Conference on Computer Communications*, Apr. 29–May 2, 2019, pp.1387–1395. DOI: 10.1109/INFOCOM.2019.8737464.

[27] Jin Y B, Jiao L, Qian Z Z, Zhang S, Lu S L, Wang X L. Resource-efficient and convergence-preserving online participant selection in federated learning. In *Proc. the 40th International Conference on Distributed Computing Systems*, Nov. 29–Dec. 1, 2020, pp.606–616. DOI: 10.1109/ICDCS47774.2020.00049.

[28] Chen M Z, Yang Z H, Saad W, Yin C C, Poor H V, Cui S G. A joint learning and communications framework for federated learning over wireless networks. *IEEE Trans. Wireless Communications*, 2021, 20(1): 269–283. DOI: 10.1109/TWC.2020.3024629.

[29] Mitchell S, Kean A, Mason A, O'Sullivan M, Phillips A, Peschiera F. PuLP 2.6. 0. https://pypi.org/project/PuLP/, July 2023.

[30] Beal L D R, Hill D C, Martin R A, Hedengren J D. GEKKO optimization suite. *Processes*, 2018, 6(8): 106. DOI: 10.3390/pr6080106.

[31] Lai P, He Q, Abdelrazek M, Chen F F, Hosking J, Grundy J, Yang Y. Optimal edge user allocation in edge computing with variable sized vector bin packing. In *Proc. the 16th International Conference on Service-Oriented Computing*, Nov. 2018, pp.230–245. DOI: 10.1007/978-3-030-03596-9_15.

[32] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 2011, 12: 2825–2830.

[33] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv: 1708.07747, 2017. https://arxiv.org/abs/1708.07747, Jul. 2023.

[34] Warden P. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv: 1804.03209, 2018. https://arxiv.org/abs/1804.03209, Jul. 2023.

[35] Zhang X, Zhao J B, LeCun Y. Character-level convolutional networks for text classification. In *Proc. the 28th Advances in Neural Information Processing Systems*, Dec. 2015, pp.649–657. DOI: 10.5555/2969239.2969312.

[36] Wei X, Fan L, Guo Y, Gong Y, Han Z, Wang Y. Quantum assisted scheduling algorithm for federated learning in distributed networks. In *Proc. the 32nd International Conference on Computer Communications and Networks*, Jul. 2023. DOI: 10.1109/ICCCN58024.2023.10230094.

Xinliang Wei is a Ph.D. student in the Department of Computer and Information Sciences at Temple University, Philadelphia. He received his M.S. and B.E. degrees both in software engineering from Sun Yat-sen University, Guangzhou, in 2016 and 2014, respectively. His research interests include edge computing, federated learning, reinforcement learning, and Internet of Things. He is a recipient of Outstanding Research Assistant from College of Science and Technology and Scott Hibbs Future of Computing Award from Department of Computer & Information Sciences at Temple University.

Jiyao Liu is a Ph.D. candidate at the Department of Computer and Information Sciences at Temple University, Philadelphia. He received his B.E. degree in information security from North China University of Technology, Beijing, in 2016. His research interests include AI, security, and privacy in edge computing.

Yu Wang is a professor in the Department of Computer and Information Sciences at Temple University, Philadelphia. He received his Ph.D. degree from Illinois Institute of Technology, Chicago, his M.Eng. degree and B.Eng. degree from Tsinghua University, Beijing, all in computer science. His research interests include wireless networks, smart sensing, and mobile computing. He has published over 200 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 at the University of North Carolina at Charlotte (2008), Fellow of IEEE (2018), and ACM Distinguished Member (2020). He has served as associate editor for IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, among others.