



TEMPLE
UNIVERSITY

Learning Scheduling and Optimization in Federated Edge Learning

Yu Wang
Temple University
November 14, 2023



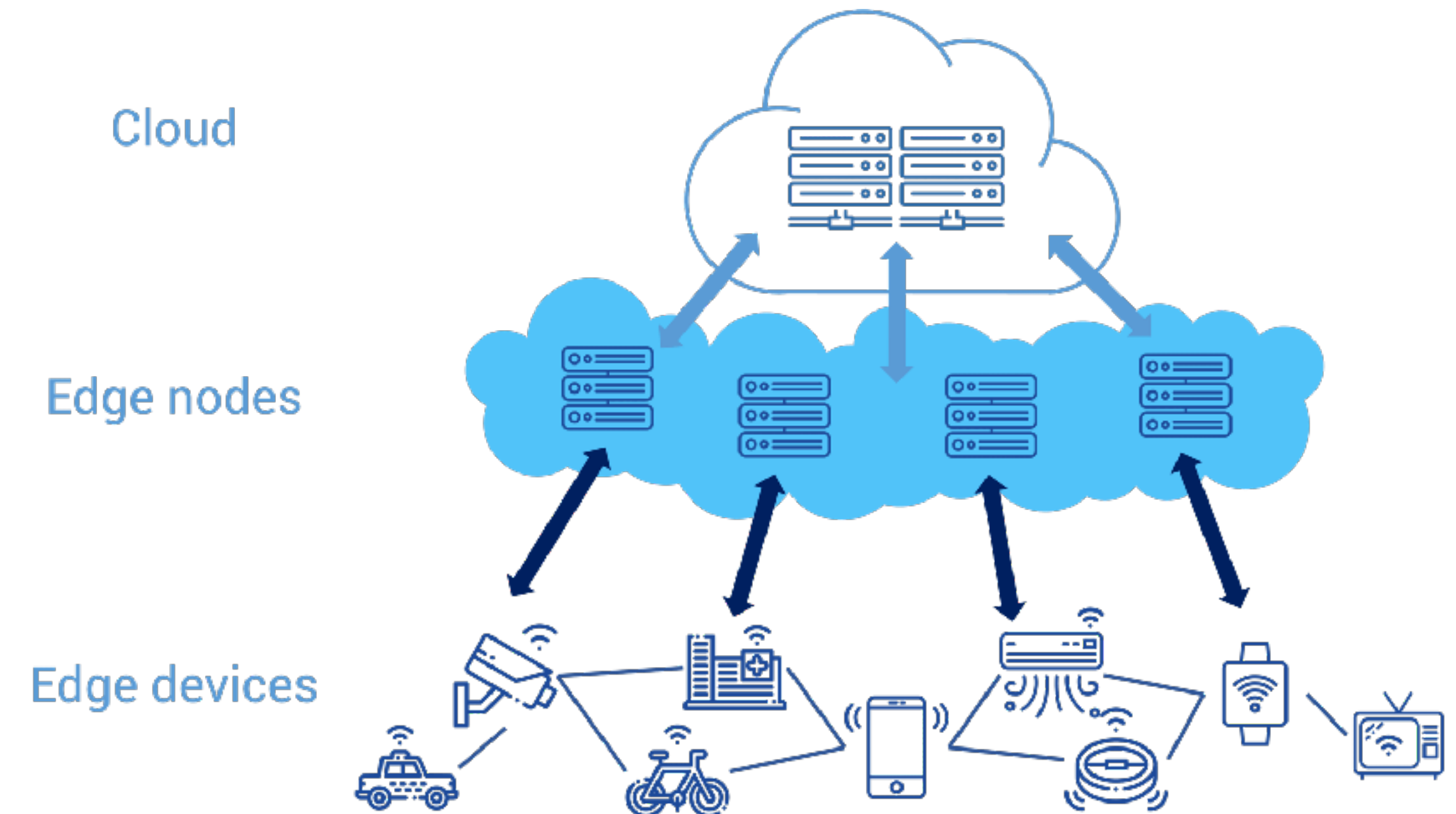
TEMPLE
UNIVERSITY

Outline

- Introduction
 - ▶ Mobile edge computing and federated learning
 - ▶ Federated edge learning (FEL)
- Learning Scheduling and Optimization in FEL
 - ▶ Problem and multi-stage solution
 - ▶ Consider learning topology
 - ▶ Quantum-assisted solution
- Conclusion

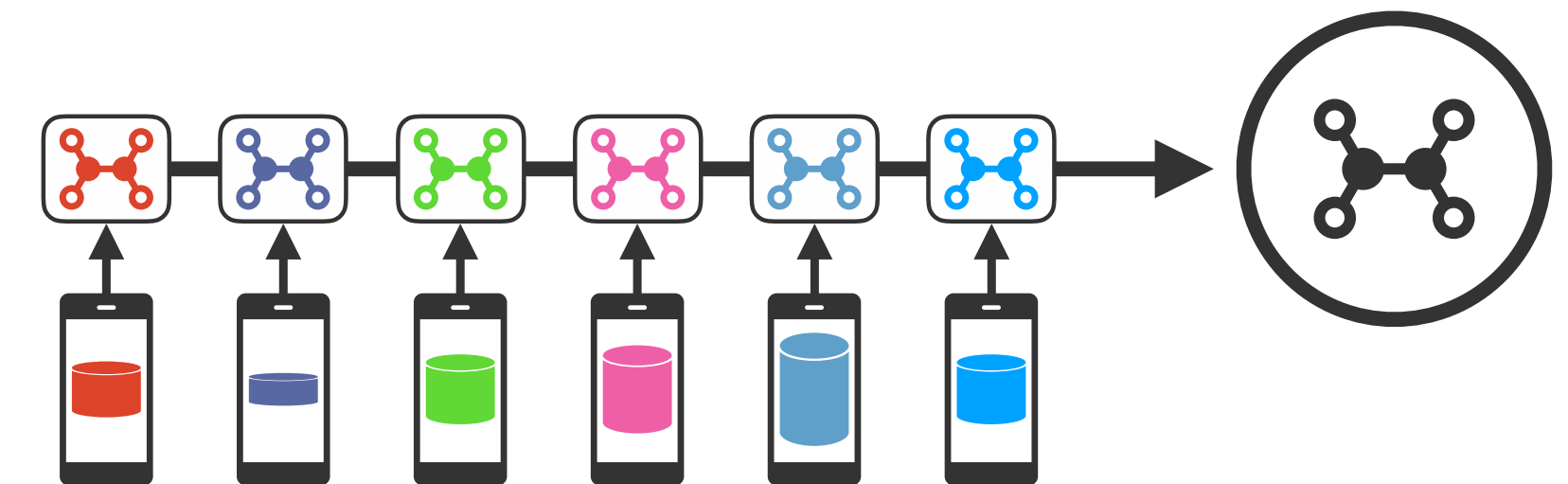
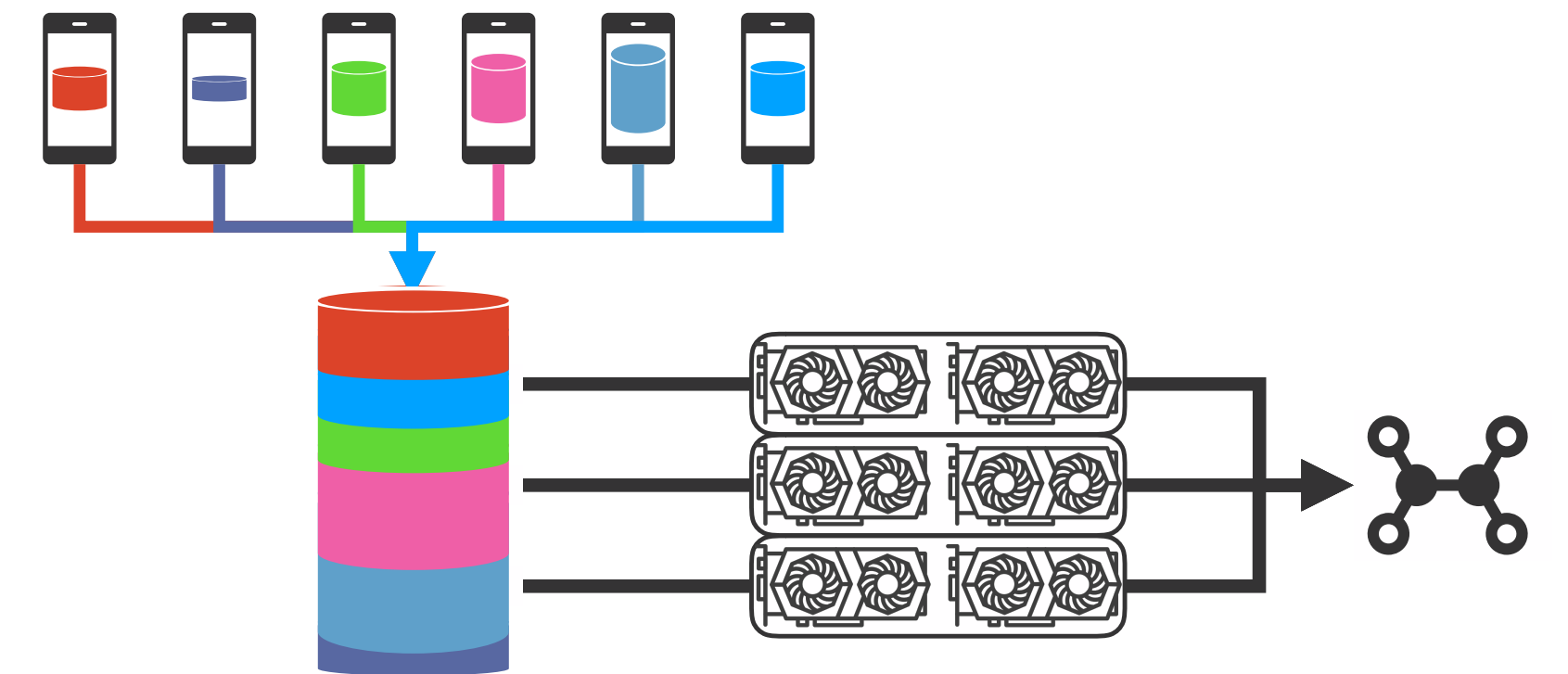
Mobile Edge Computing

- Mobile Edge Computing (MEC) - a “new” computing paradigm
 - offers applications and content providers **cloud-computing capabilities** and a **service environment at the edge** of the mobile network
 - supports diverse services (e.g., data management, mobile computing, **ML/AI services**) for wide range of applications and IoT/smart/mobile devices
- This environment is characterized by
 - proximity
 - location awareness
 - ultra-low latency
 - high bandwidth
 - real-time access to radio network and network resources



Federated Learning

- Federated Learning (FL) - a “new” distributed ML paradigm
 - many clients (e.g. mobile devices) collaboratively train a shared ML model under the orchestration of a central server (PS), while keeping the training data decentralized
- Advantage over traditional, centralized ML
 - embody the principles of focused data collection and minimization
 - mitigate many of the systemic privacy risks and costs

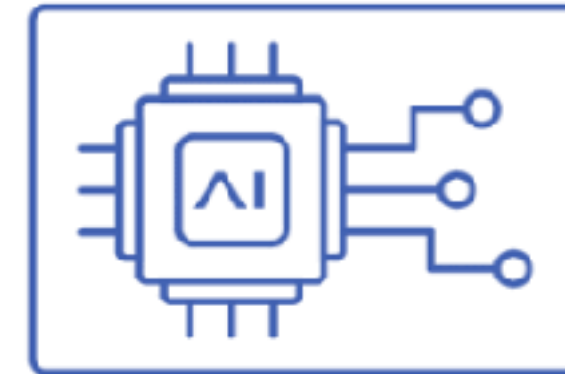


Src: https://qcon.ai/system/files/presentation-slides/qcon_-_federated_learning.pdf

Federated Learning Meets Edge Computing

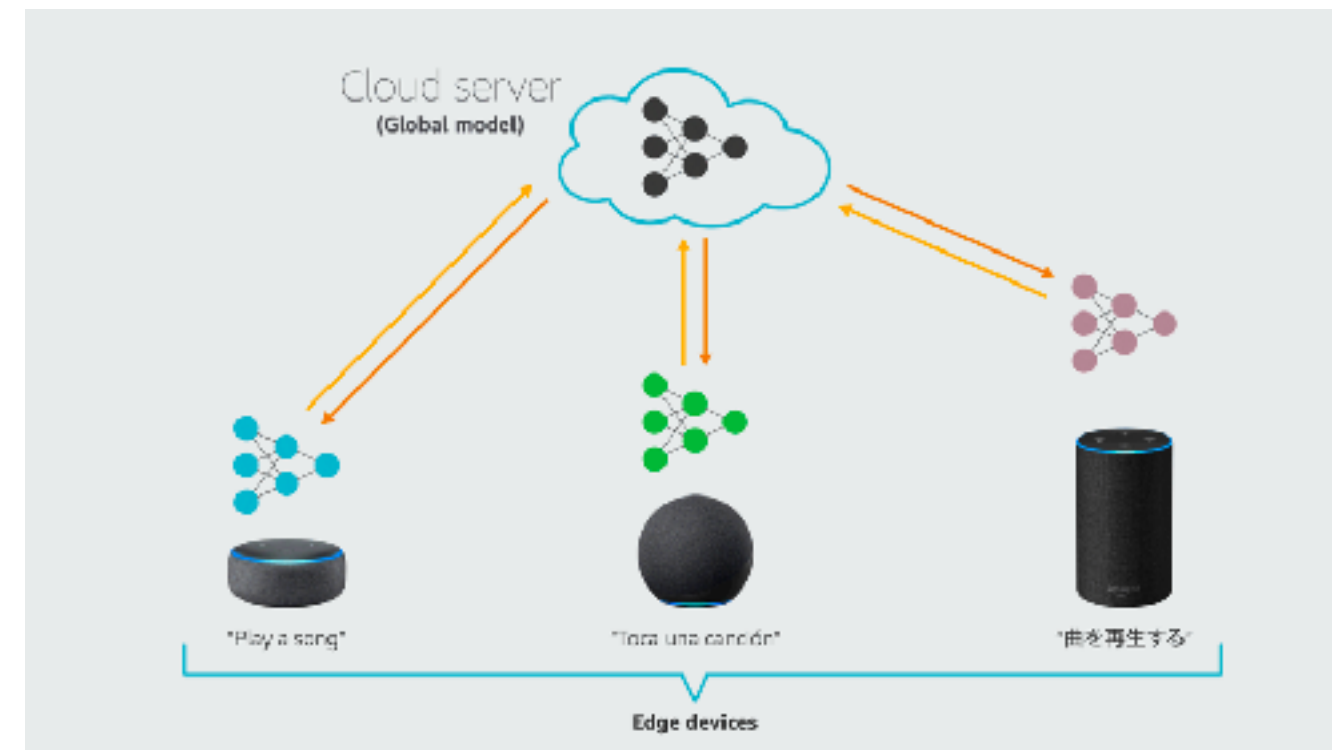
- MEC and FL share the same principles

- ▶ keep data/computing closer to users
- ▶ protect user privacy
- ▶ leverage distributed resources

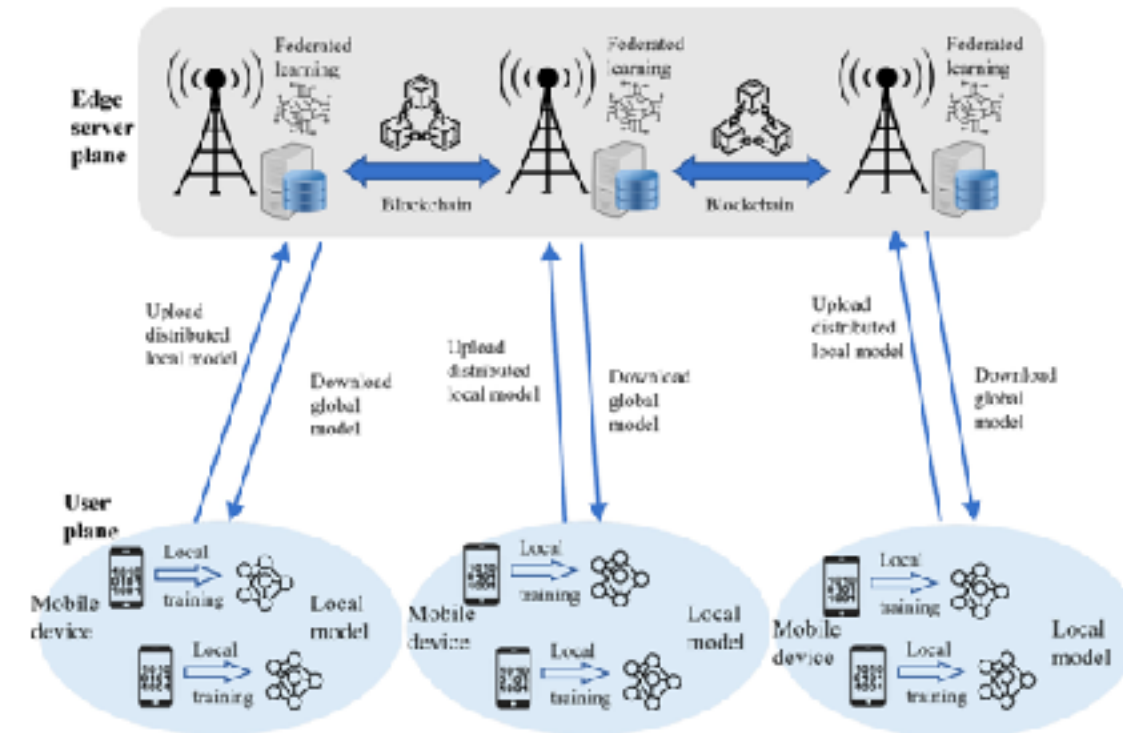


- Federated Edge Learning (FEL)

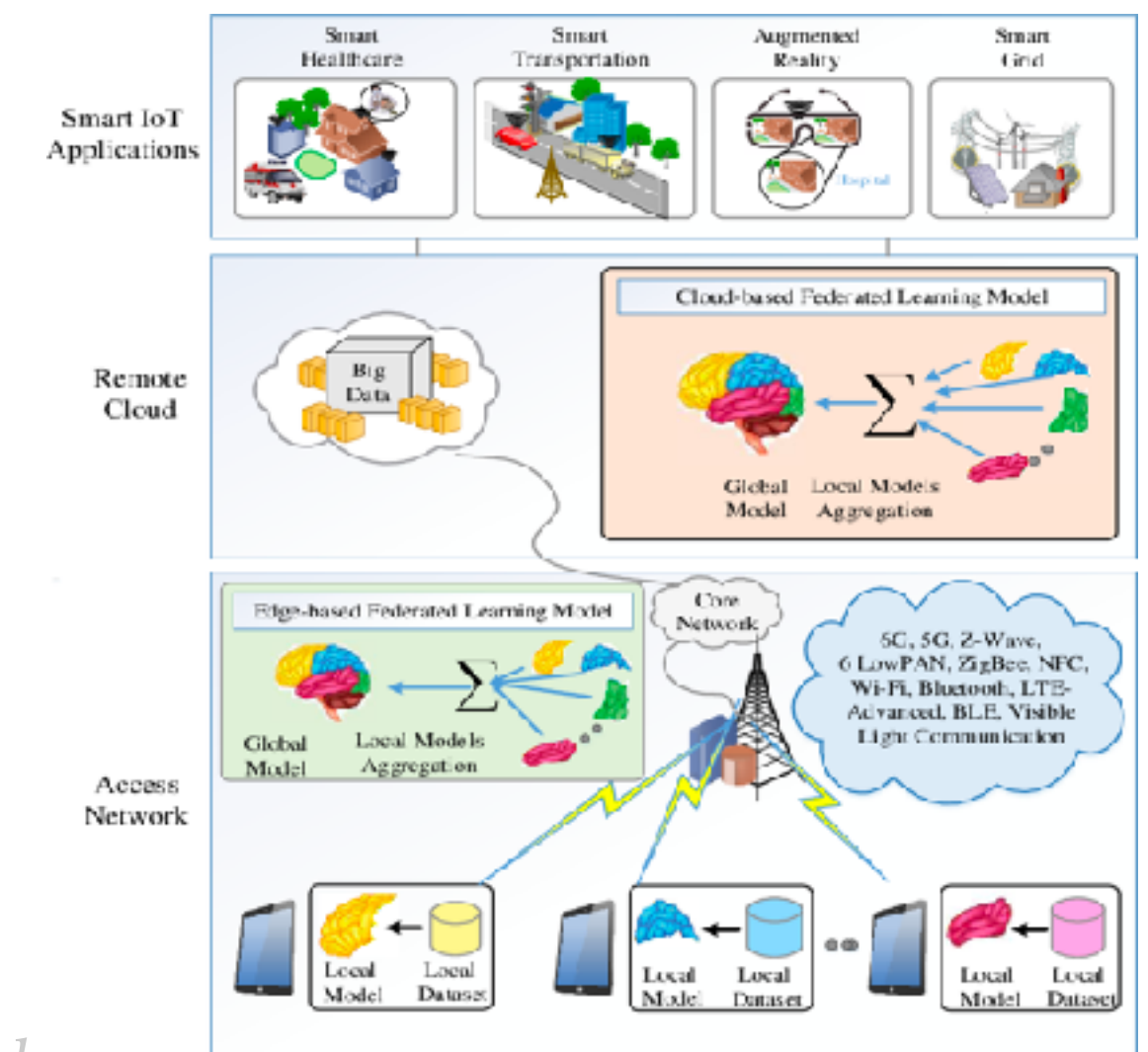
- ▶ widely studied and applied in many application scenarios recently
- ▶ enables **Edge AI** & **Edge Intelligence**



<https://www.amazon.science/blog/personalized-federated-learning-for-a-better-customer-experience>



<https://doi.org/10.1002/dac.5367>



<https://arxiv.org/pdf/1911.05642.pdf>

Federated Edge Learning: Current & Challenges

- Current works in FEL focus on
 - learning convergence and learning control [1,2]
 - communication and energy efficiency [3,4,5]
 - edge association and resource management [5,6]
 - client selection/sampling to battle non-IID data [7,8,9]
 - model aggregation and learning topology [5,10,11]
 - Most of them on a single shared global ML model with fixed learning topology

- Multi-model FEL
 - multiple FL models trained simultaneously (resource competition, affecting performance of each other)
 - different convergence performance with different learning rate settings or learning topologies

[1] Client-edge-cloud hierarchical federated learning, ICC, 2020.

[2] Adaptive federated learning in resource constrained edge computing systems, JSAC, 2019.

[3] Energy efficient federated learning over wireless communication networks, TWC, 2020.

[4] To talk or to work: flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices, INFOCOM, 2021.

[5] SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning, ICDCS 2021

[6] HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning, TWC 2020

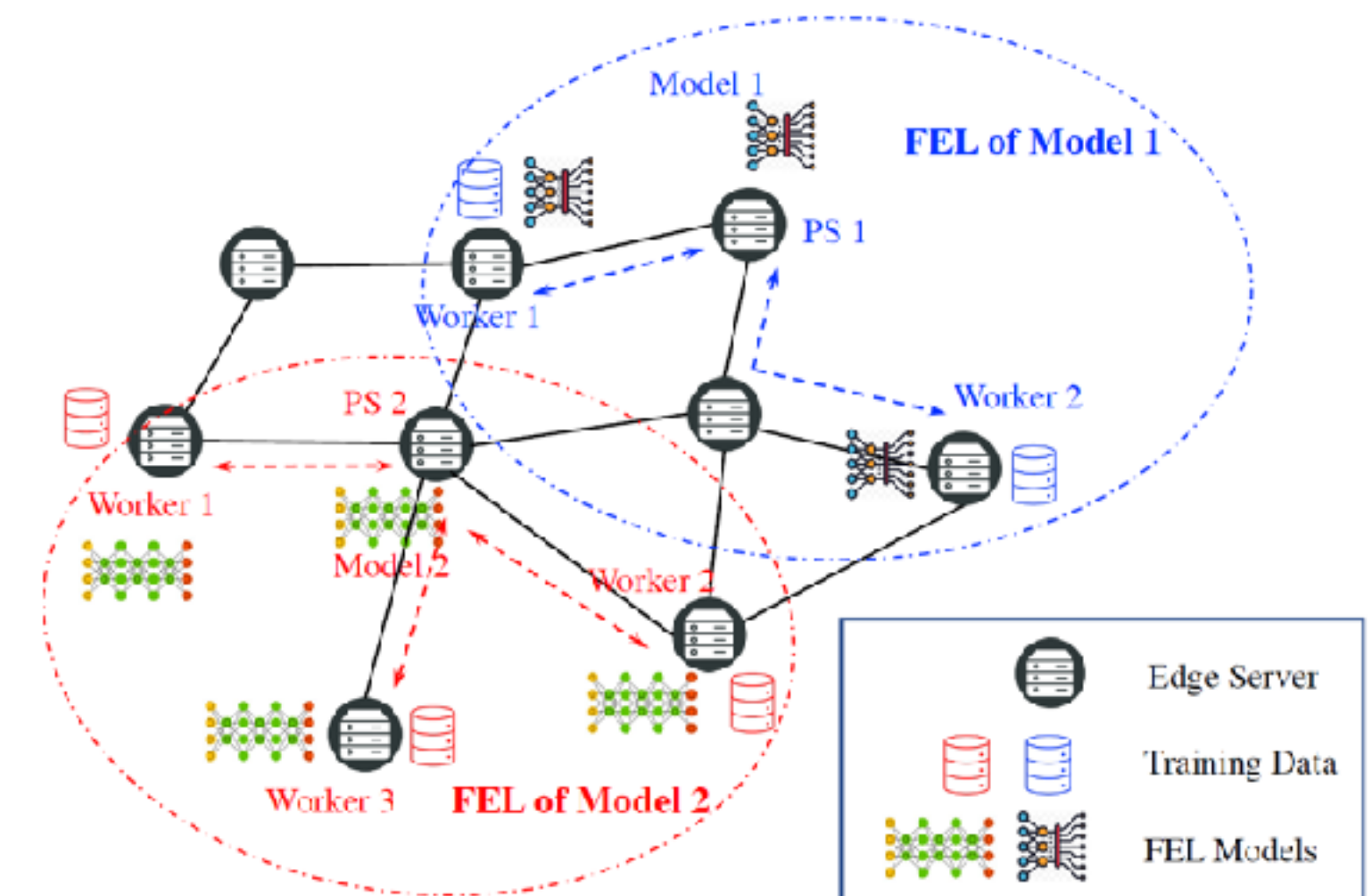
[7] Sample-level data selection for federated learning, INFOCOM 2021

[8] Power of redundancy: Surplus client scheduling for federated learning against user uncertainties, TMC 2022

[9] Client selection for federated learning with heterogeneous resources in mobile edge, ICC 2019

[10] Resource-efficient federated learning with hierarchical aggregation in edge computing, in INFOCOM 2021

[11] Learning-driven decentralized machine learning in resource-constrained wireless edge computing, INFOCOM 2021



Our Work on Federated Edge Learning

Perspective	Description
Participant Selection	How to select PS, workers to reduce total learning costs?
Learning Scheduling	How to adjust training iteration, convergence speed, etc.?
Data Distribution	How to handle different data distribution?
Learning Topology	How to determine the optimal learning topology (CFL, DFL, HFL)?
Learning Cost	Including communication cost, computing cost, rental cost, etc.?
...	...

Centralized Federated Learning

Joint Participant Selection and Learning Scheduling for Multi-Model FEL



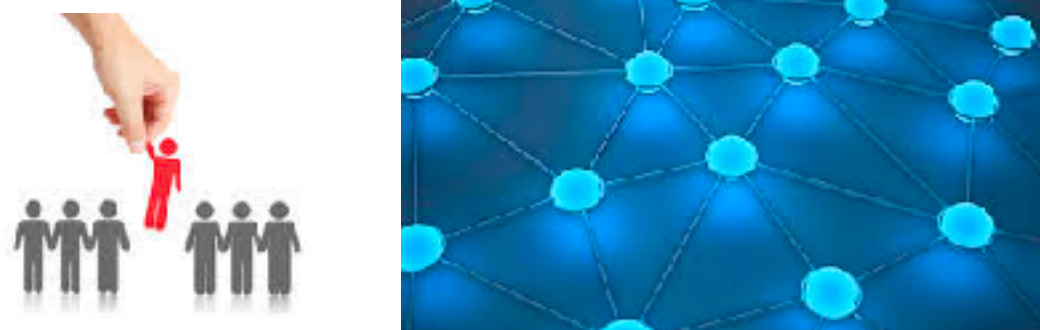
Qutaum-Assistant Fed Learning

Joint Participant Selection and FL Scheduling in Distributed Networks



Topology-Aware Fed Learning

Joint Participant and Learning Topology Selection in Multi-Model FEL



Hierarchical Federated Learning

Group Formation and Group Sampling for Gourd-based Hierarchical FEL

Outline

Joint Participant Selection and Scheduling in FEL

Joint Participant Selection and Learning Scheduling for Multi-Model Federated Edge Learning

Xinliang Wei, Jiyao Liu, Yu Wang
Department of Computer and Information Sciences, Temple University, Philadelphia, USA
{xinliang.wei,jiyao.liu,wangyu}@temple.edu

Abstract—As edge computing complements the cloud to enable computational services right at the network edge, federated learning (FL) can also benefit from close-by edge computing infrastructure. However, most prior works on federated edge learning (FEL) mainly focus on one shared global model during the federated training in edge systems. In a real edge computing scenario, there may co-exist multiple various FL models that are owned by different entities and used by different applications. Simultaneously training these models competes both computing and networking resources in the shared edge system. Therefore, in this work, we consider a multi-model federated edge learning where multiple FEL models are being trained in the edge network and edge servers can act as either parameter servers or workers of these FEL models. We formulate a joint participant selection and learning scheduling problem, which is a non-linear mixed-integer program, aiming to minimize the total cost of all FEL models while satisfying the desired convergence rate of trained FEL models and the constrained edge resources. We then design several algorithms by decoupling the original problem into two or three sub-problems which can be solved respectively and iteratively. Extensive simulations with real-world training datasets and FEL models show that our proposed algorithms can efficiently reduce the average total cost of all FEL models in a multi-model FEL setting compared with existing algorithms.

1. INTRODUCTION

With the advances of Internet of Things, smart sensing and artificial intelligence, there has been a tremendous trend that data sources shift from the cloud center to the network edge. Generally, in order to train a machine learning (ML) model, one needs to upload the collected training data to the cloud data center and train the model using the whole dataset there. However, it is non-trivial to send a large amount of data to the remote data center due to the limited network bandwidth and data privacy concerns. Therefore, an alternative solution is the distributed training of ML models at the network edge or even on the user devices. However, there are still major challenges to prevent users from performing efficient model training at the edge. On one hand, the computing capacity

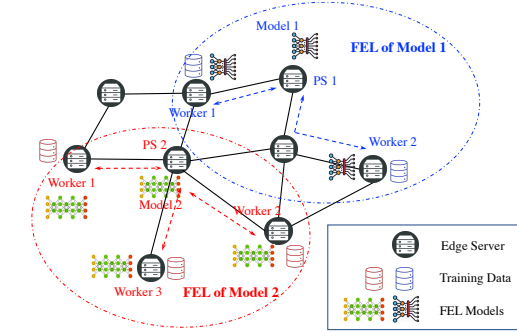


Fig. 1. Multi-model FEL example: two FEL models are trained with 3 and 4 participants (1 PS + 2 or 3 workers), respectively, in a shared edge network.

resources and the competition among various users, servers and applications.

Recently, *federated learning* (FL) has been emerging as a new distributed machine learning paradigm [1]–[3], which enables multiple servers collaboratively learn a shared ML model while keeping all training data on the local server. It is very natural to deploy the FL framework in edge computing to provide efficient distributed training at the network edge. Therefore, *federated edge learning* (FEL) has been proposed in various settings [4]–[12]. In FEL, edge servers can collaboratively train a shared global ML model by aggregating local models trained at individual local servers, decoupling the ability to do model training from the need to store data in 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 local training based on their local data. After that, edge servers will upload their local model to the parameter server which is responsible for aggregating parameters from different workers and sending the

Joint Participant and Topology Selection in FEL

Joint Participant and Learning Topology Selection in Federated Edge Learning

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

Abstract—Deploying federated learning (FL) in edge clouds is a challenging task, particularly when multiple models are trained concurrently in resource-constrained edge environments. Current research on federated edge learning primarily focuses on client selection for training a single FL model with a fixed learning topology. Our experiments demonstrate that FL models with adaptable topologies result in lower learning costs than those with fixed topologies. In this paper, we investigate the problem of jointly selecting participants and learning topologies for multiple FL models being trained simultaneously in the edge cloud. We formulate this as an integer programming problem, with the goal of minimizing total learning costs for all FL models, subject to edge resource constraints. We propose a two-stage algorithm that decouples the original problem into two sub-problems and addresses them iteratively. By allowing FL models to independently select participants and learning topologies, our method improves resource competition and load balancing in edge clouds. Our extensive experiments with real-world networks and FL datasets confirm the superior performance of our algorithm in terms of average total cost compared to prior methods for multi-model FL.

Index Terms—Edge Computing, Federated Learning, Participant Selection, Learning Topology

1 INTRODUCTION

Federated Learning (FL) [1]–[7] is an efficient approach for improving machine learning (ML) performance and providing better privacy solutions for data owners. It enables multiple devices to collaborate and train a shared global ML model by aggregating local models trained on each device. FL ensures that training data remains local to protect users’ privacy, and only transmits essential model data (e.g. gradients). With the growth of smart sensing, mobile computing, and wireless networking, there is also a trend of moving data sources and intelligent computation from centralized clouds to edge clouds, to provide agile services to mobile devices and users. Therefore, it is important to deploy FL frameworks on edge clouds and provide efficient distributed training for mobile devices at the network edge. Such solutions have been studied [8]–[12] and can support many emerging applications [13], such as mobile AI, AIoT or AR/XR applications.

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 [10] where

train the model by using their local data. After each worker performs several local updates, the local model will be forwarded to the PS for global aggregation. The potential bottleneck of CFL is the communication congestion at the PS since all workers have to communicate with the PS concurrently for multiple rounds. In addition, the PS in CFL may cause a single point of failure. Therefore, DFL [11], [14] has been proposed, where each worker only communicates with its neighbors (with mutual trust) by exchanging their local models and there is no centralized PS, as shown in Fig. 1(b). While such distributed P2P learning topology increases the robustness of FL, it might suffer from larger communication costs or slower convergence. Recently, HFL [15]–[17] has been proposed by introducing several middle-layer PSs (or called group leaders) in a hierarchical topology such that each of them only aggregates a group model from workers inside its group and sends the group model to the PS for global aggregation, as shown in Fig. 1(c). HFL can effectively hide local updates submitted by individual workers within a group, thereby enhancing privacy protection from malicious or honest-but-curious PS [15]. HFL can also provide better scalability with larger workers but may increase the total latency due to multiple exchanges

Qutaum-Assistant Federated Learning Scheduling

Quantum Assisted Scheduling Algorithm for Federated Learning in Distributed Networks

Xinliang Wei*, Lei Fan¹, Yuanxiong Guo², Yanming Gong³, Zhu Han⁴, Yu Wang*
*Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA
¹Department of Engineering Technology, University of Houston, Houston, TX, USA
²Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, USA
³Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX, USA
⁴Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA
{xinliang.wei, wangyu}@temple.edu, lfan8@central.uh.edu, {yuanxiong.guo, yanmin.gong}@utsa.edu, zhan2@uh.edu.

Abstract—The scheduling problem for federated learning (FL) with multiple models in a distributed network is challenging, as it involves NP-hard mixed-integer nonlinear programming. Moreover, it requires optimal participant selection and learning rate determination among multiple FL models to avoid high training costs and resource competition. To overcome those challenges, in literature the Benders’ decomposition algorithm (BD) can deal with mixed integer problems, however, it still suffers from limited scalability. To address this issue, in this paper, we present the Hybrid Quantum-Classical Benders’ Decomposition (HQCBD) algorithm, which combines the power of quantum and classical computing to solve the joint participant selection and learning scheduling problem in multi-model FL. HQCBD decomposes the optimization problem into a master problem with binary variables and small subproblems with continuous variables. This collaboration maximizes the potential of both quantum and classical computing, and optimizes the complex joint optimization problem. Simulation on the commercial D-Wave quantum annealing machine demonstrates the effectiveness and robustness of the proposed method, with up to 18% improvement of iterations and 81% improvement of computation time over BD algorithm on classical CPUs even at small scales.

Index Terms—Federated learning, participant selection, learning scheduling, hybrid quantum-classical optimization

1. INTRODUCTION

With the use of quantum superposition and entanglement, quantum computing (QC) has demonstrated a quantum advantage over classical computing in random quantum circuit sampling [1], Gaussian boson sampling [2], and combinatorial optimization [3]–[5]. In this paper, by leveraging the parallel computing capability of quantum computing, we focus on designing a new quantum-assisted scheduling algorithm to solve a complex joint participant selection and learning scheduling problem for federated learning (FL) in distributed networks.

Federated learning is emerging as an effective and privacy-preserving machine learning (ML) paradigm [6]–[9], which

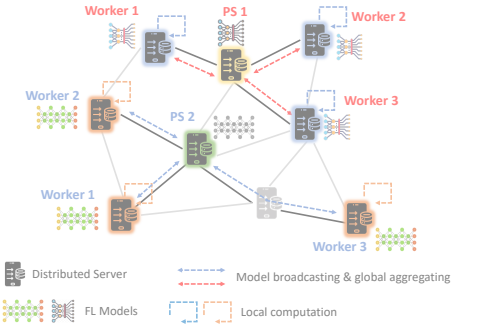


Fig. 1: The training process of multi-model federated learning.

the computing capability and network resources of servers and their data distribution are heterogeneous. Some low-performance servers may decelerate the convergence process and diminish the training performance. Also, the dispersed computing resources and large network latency may lead to high training costs. Second, for the practical scenario, training multiple different models in the shared distributed network simultaneously leads to competition for computing and communication resources. As shown in Fig. 1, two FL models are trained concurrently and each FL model requires one PS and three workers for model training. In this case, which FL model is preferentially served at which server directly affects the total training cost of all FL models. To this end, appropriate participant selection and learning schedules are fairly crucial for multi-model FL training.

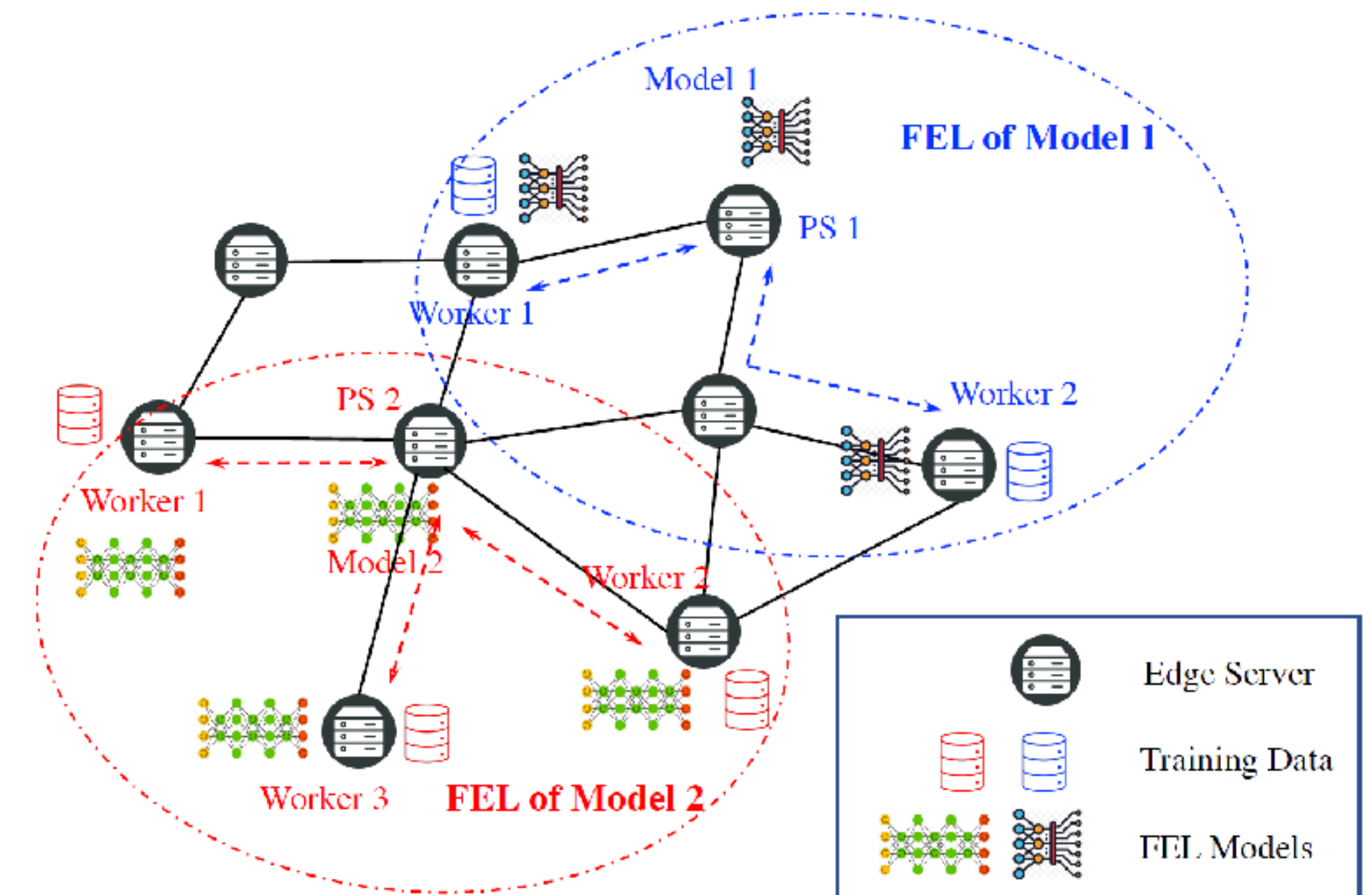
Therefore, we mainly concentrate on the joint participant selection and learning scheduling problem in multi-model FL training scenarios. It should be emphasized that each server in



Joint Participant Selection and Scheduling in FEL

Motivation

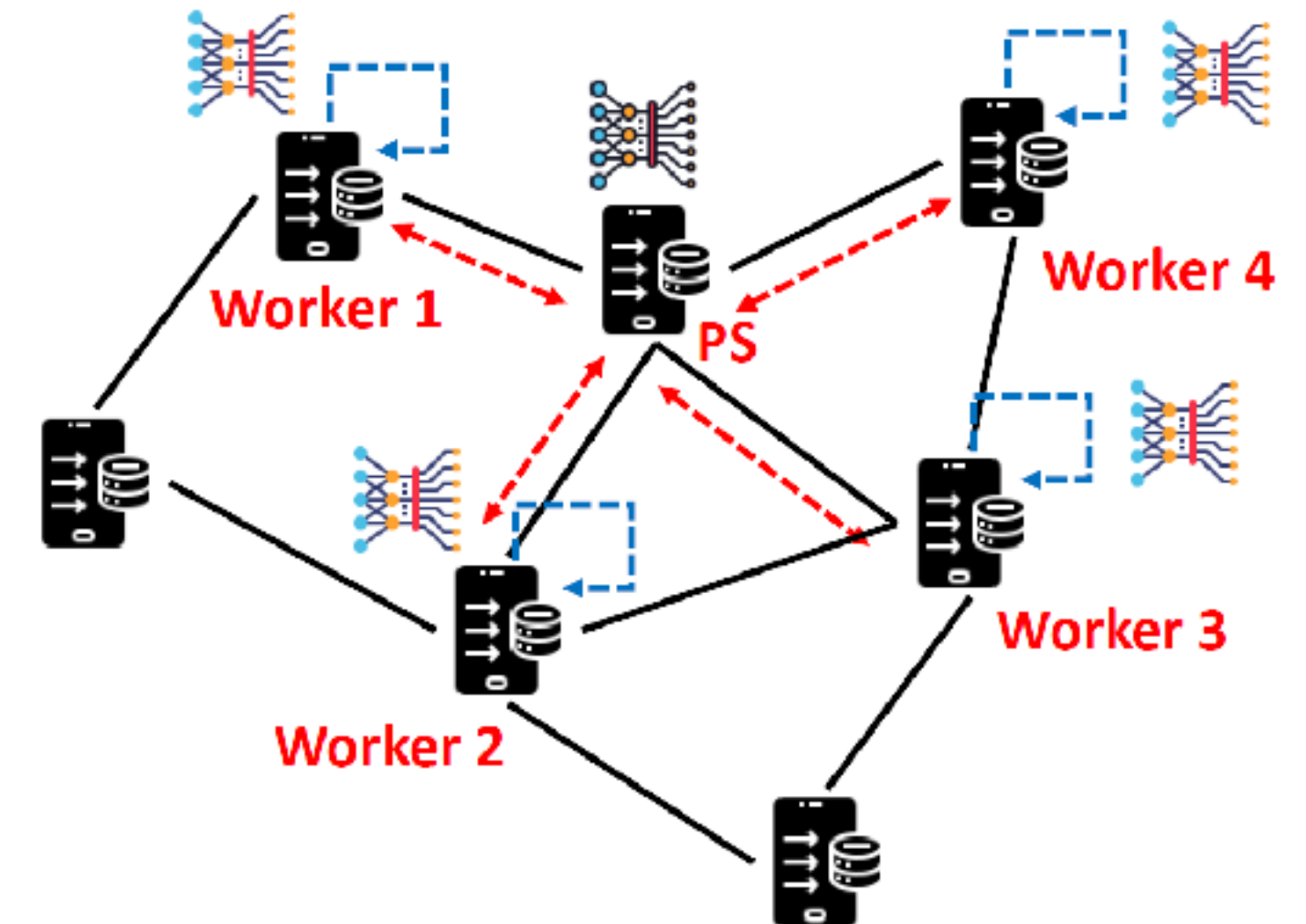
- Multi-model FEL
 - ▶ multiple FL models trained simultaneously (**resource competition**, affecting performance of each other)
 - ▶ **participant selection**: certain workers and a parameter server (PS) for each model
 - ▶ **learning scheduling**: adjust local learning rate of each model



Joint Participant Selection and Scheduling in FEL

FL Model

- Each global iteration includes four steps
 - Selected parameter server (PS) **initializes the global model**;
 - Selected workers **download the global model** from PS;
 - Each worker **runs the local updates** using its holding raw dataset for certain local iterations φ_j ;
 - Workers **upload the updated model** to PS for global aggregation to **update the global model**.



of global iterations: $\vartheta_j \geq \frac{2\lambda^2}{\gamma^2\xi} \ln\left(\frac{1}{\varsigma_j}\right) \frac{1}{1-\varrho_j} \triangleq \vartheta_0 \ln\left(\frac{1}{\varsigma_j}\right) \frac{1}{1-\varrho_j}$, local convergence rate

of local updates: $\varphi_j \geq \frac{2}{(2-\lambda\delta)\delta\gamma} \log_2\left(\frac{1}{\varrho_j}\right) \triangleq \varphi_0 \log_2\left(\frac{1}{\varrho_j}\right)$

[12] Learning for learning: predictive online control of federated learning with edge provisioning, InfoCom 21

Joint Participant Selection and Scheduling in FEL

Problem Formulation

mixed integer non-linear programming problem (MINLP)

- Joint participant selection and learning scheduling

Goal: minimize the total cost of all models

$$\min \sum_{j=1}^W \varpi_j^t \quad (3)$$

CPU frequency

$$x_{i,j}^t \mu_j \kappa_j \leq c_i^t, \quad x_{i,j}^t \chi_j \leq f_i^t \quad \forall i, j \quad (4)$$

$$y_{i,j}^t \mu_j \leq c_i^t, \quad y_{i,j}^t \chi_j \leq f_i^t \quad \forall i, j \quad (5)$$

$$w_{j,k} y_{i,j}^t z_{i,k} = 1 \quad \forall i, j, k \quad (6)$$

edge has dataset for model

$$\sum_i x_{i,j}^t = 1, \quad \sum_i y_{i,j}^t = \kappa_j \quad \forall j \quad (7)$$

One PS, κ_j FL workers

$$\sum_j (x_{i,j}^t + y_{i,j}^t) \leq 1 \quad \forall i \quad (8)$$

One edge server can at most work as one of two roles

$$x_{i,j}^t \in \{0, 1\}, y_{i,j}^t \in \{0, 1\}, \varrho_j^t \in [0, 1). \quad (9)$$

Storage constraints

Participant selection and learning schedule decision

Total learning cost

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

- Edge communication cost

Communication cost based on shortest path

$$C_j^{comm,t} = 2 \cdot \vartheta_j^t \sum_{k=1}^N \sum_{i=1}^N x_{k,j}^t \cdot y_{i,j}^t \cdot \rho_j(v_i, v_k)$$

- Local update cost

CPU cycles to process the sample data $D_{j,i}^t$

$$C_j^{local,t} = \vartheta_j^t \cdot \varphi_j^t \cdot \sum_{i=1}^N y_{i,j}^t \cdot \frac{\psi(D_{j,i}^t)}{f_i^t}$$

- Global aggregation cost

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

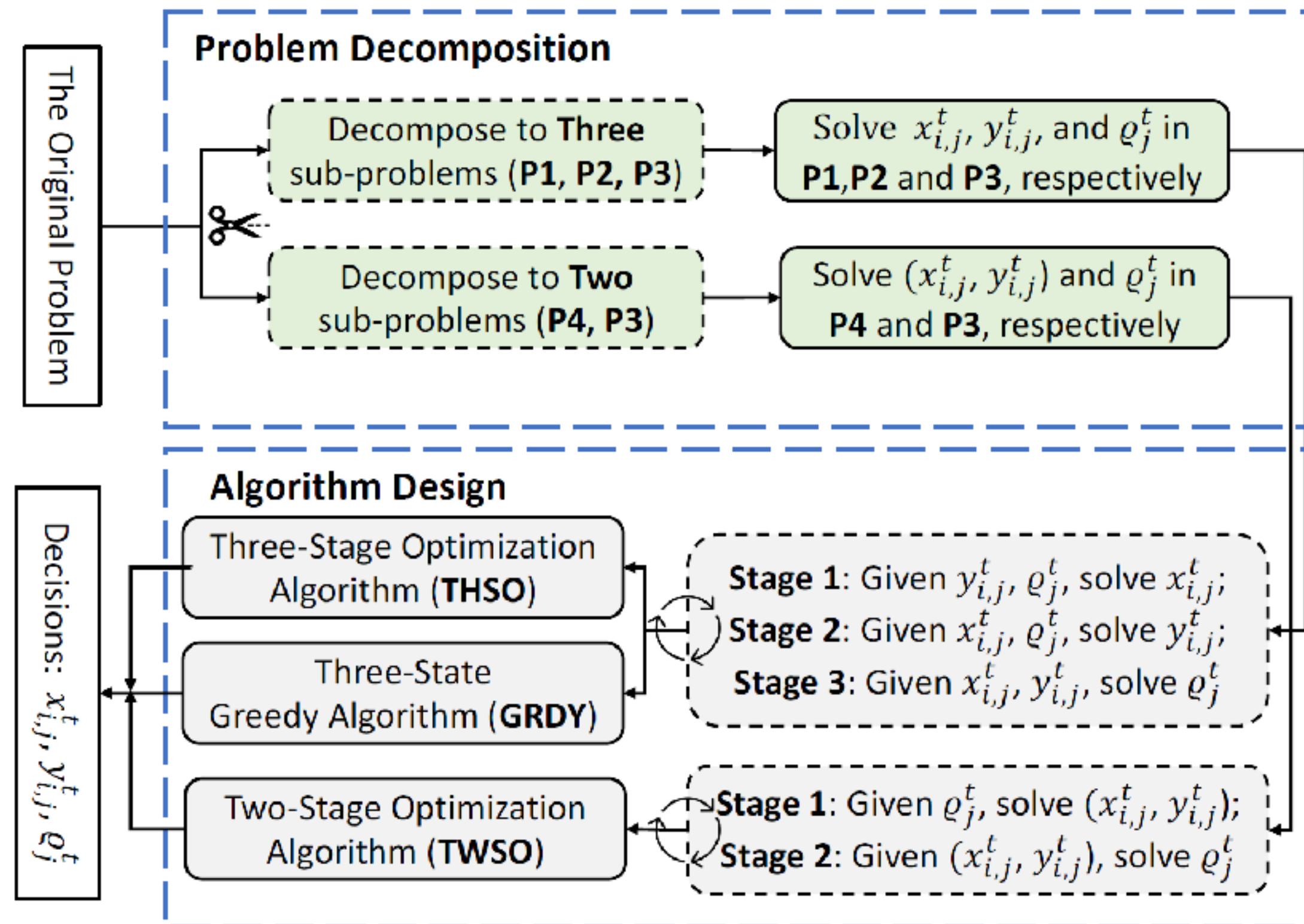
- Initialization cost

Downloading cost

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

Joint Participant Selection and Scheduling in FEL

Methodology



- Solving **P1 parameter server (PS) selection** with fixed workers selection and local convergence rate

$$\mathbf{P1:} \quad \min_{x_{i,j}^t} \sum_{j=1}^w \varpi_j^t$$

s.t. (4), (7), (9)

- Solving **P2 worker selection** with the latest PS and fixed local convergence rate

$$\mathbf{P2:} \quad \min_{y_{i,j}^t} \sum_{j=1}^w \varpi_j^t$$

s.t. (5) – (9)

- Solving **P3 local convergence rate** with the latest PS and worker decision

$$\mathbf{P3:} \quad \min_{\varrho_j^t} \sum_{j=1}^w \varpi_j^t$$

s.t. (9)

Joint Participant Selection and Scheduling in FEL

Methodology

Algorithm 1 Three-Stage Optimization Method

```

1: Initialize  $max\_itr$ ,  $max\_occur$ ,  $bound\_val$ 
2: Generate an random initial FEL worker selection decision  $y_{i,j}^{t,0}$  and local convergence rate  $\varrho_j^{t,0}$ 
3:  $\iota = 1$  and  $count\_num = 0$ ;
4: repeat
5:   Stage 1: Calculate  $x_{i,j}^{t,\iota}$  by solving P1 with fixed  $y_{i,j}^{t,\iota-1}$  and  $\varrho_j^{t,\iota-1}$ 
6:   Stage 2: Calculate  $y_{i,j}^{t,\iota}$  by solving P2 with fixed  $x_{i,j}^{t,\iota}$  and  $\varrho_j^{t,\iota-1}$ 
7:   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 cost of FEL models)
8:   if  $obj\_val > bound\_val$  then
9:      $bound\_val = obj\_val$ ;  $count\_num = 1$ 
10:     $x_{j,i}^t = x_{j,i}^{t,\iota}$ ;  $y_{k,i}^t = y_{k,i}^{t,\iota}$ ;  $\varrho_j^t = \varrho_j^{t,\iota}$ 
11:  else if  $obj\_val = bound\_val$  then
12:     $count\_num = count\_num + 1$ 
13:  end if
14:   $\iota = \iota + 1$ 
15: until  $count\_num = max\_occur$  or  $\iota = max\_itr$ 
16: return  $x_{i,j}^t$ ,  $y_{i,j}^t$  and  $\varrho_j^t$ 

```

Stage 1

➤ Solving **P1 parameter server (PS) selection** with fixed workers selection and local convergence rate

$$\begin{aligned}
 \mathbf{P1:} \quad & \min_{x_{i,j}^t} \sum_{j=1}^W \varpi_j^t \\
 & \text{s.t.} \quad (4), (7), (9)
 \end{aligned}$$

Joint Participant Selection and Scheduling in FEL

Methodology

Algorithm 1 Three-Stage Optimization Method

- 1: Initialize max_itr , max_occur , $bound_val$
- 2: Generate an random initial FEL worker selection decision $y_{i,j}^{t,0}$ and local convergence rate $\varrho_j^{t,0}$
- 3: $\iota = 1$ and $count_num = 0$;
- 4: **repeat**
- 5: Stage 1: Calculate $x_{i,j}^{t,\iota}$ by solving **P1** with fixed $y_{i,j}^{t,\iota-1}$ and $\varrho_j^{t,\iota-1}$
- 6: Stage 2: Calculate $y_{i,j}^{t,\iota}$ by solving **P2** with fixed $x_{i,j}^{t,\iota}$ and $\varrho_j^{t,\iota-1}$
- 7: 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 cost of FEL models)
- 8: **if** $obj_val > bound_val$ **then**
- 9: $bound_val = obj_val$; $count_num = 1$
- 10: $x_{j,i}^t = x_{j,i}^{t,\iota}$; $y_{k,i}^t = y_{k,i}^{t,\iota}$; $\varrho_j^t = \varrho_j^{t,\iota}$
- 11: **else if** $obj_val = bound_val$ **then**
- 12: $count_num = count_num + 1$
- 13: **end if**
- 14: $\iota = \iota + 1$
- 15: **until** $count_num = max_occur$ or $\iota = max_itr$
- 16: **return** $x_{i,j}^t$, $y_{i,j}^t$ and ϱ_j^t

Stage 2

- Solving **P2 worker selection** with the latest PS and fixed local convergence rate

$$\begin{aligned} \mathbf{P2} : \quad & \min_{y_{i,j}^t} \sum_{j=1}^W \varpi_j^t \\ & \text{s.t.} \quad (5) - (9) \end{aligned}$$

Joint Participant Selection and Scheduling in FEL

Methodology

Algorithm 1 Three-Stage Optimization Method

- 1: Initialize max_itr , max_occur , $bound_val$
- 2: Generate an random initial FEL worker selection decision $y_{i,j}^{t,0}$ and local convergence rate $\varrho_j^{t,0}$
- 3: $\iota = 1$ and $count_num = 0$;
- 4: **repeat**
- 5: Stage 1: Calculate $x_{i,j}^{t,\iota}$ by solving **P1** with fixed $y_{i,j}^{t,\iota-1}$ and $\varrho_j^{t,\iota-1}$
- 6: Stage 2: Calculate $y_{i,j}^{t,\iota}$ by solving **P2** with fixed $x_{i,j}^{t,\iota}$ and $\varrho_j^{t,\iota-1}$
- 7: 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 cost of FEL models)
- 8: **if** $obj_val > bound_val$ **then**
- 9: $bound_val = obj_val$; $count_num = 1$
- 10: $x_{j,i}^t = x_{j,i}^{t,\iota}$; $y_{k,i}^t = y_{k,i}^{t,\iota}$; $\varrho_j^t = \varrho_j^{t,\iota}$
- 11: **else if** $obj_val = bound_val$ **then**
- 12: $count_num = count_num + 1$
- 13: **end if**
- 14: $\iota = \iota + 1$
- 15: **until** $count_num = max_occur$ or $\iota = max_itr$
- 16: **return** $x_{i,j}^t$, $y_{i,j}^t$ and ϱ_j^t

Stage 3

- Solving **P3 local convergence rate** with the latest PS and worker decision

$$\mathbf{P3} : \quad \min_{\varrho_j^t} \quad \sum_{j=1}^W \varpi_j^t$$

s.t. (9)

Joint Participant Selection and Scheduling in FEL

Methodology

Algorithm 1 Three-Stage Optimization Method

```
1: Initialize  $max\_itr$ ,  $max\_occur$ ,  $bound\_val$ 
2: Generate an random initial FEL worker selection decision  $y_{i,j}^{t,0}$  and local convergence rate  $\varrho_j^{t,0}$ 
3:  $\iota = 1$  and  $count\_num = 0$ ;
4: repeat
5:   Stage 1: Calculate  $x_{i,j}^{t,\iota}$  by solving P1 with fixed  $y_{i,j}^{t,\iota-1}$  and  $\varrho_j^{t,\iota-1}$ 
6:   Stage 2: Calculate  $y_{i,j}^{t,\iota}$  by solving P2 with fixed  $x_{i,j}^{t,\iota}$  and  $\varrho_j^{t,\iota-1}$ 
7:   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 cost of FEL models)
8:   if  $obj\_val > bound\_val$  then
9:      $bound\_val = obj\_val$ ;  $count\_num = 1$ 
10:     $x_{j,i}^t = x_{j,i}^{t,\iota}$ ;  $y_{k,i}^t = y_{k,i}^{t,\iota}$ ;  $\varrho_j^t = \varrho_j^{t,\iota}$ 
11:  else if  $obj\_val = bound\_val$  then
12:     $count\_num = count\_num + 1$ 
13:  end if
14:   $\iota = \iota + 1$ 
15: until  $count\_num = max\_occur$  or  $\iota = max\_itr$ 
16: return  $x_{i,j}^t$ ,  $y_{i,j}^t$  and  $\varrho_j^t$ 
```

Stopping
Condition

- Update decision variables and iteratively repeat the process if the condition is not satisfied
- Either **no further improvement** of the objective value of the optimization or **reaching the maximal iteration number**

Joint Participant Selection and Scheduling in FEL

Evaluation

Network and Datasets

- Random edge networks with 20-40 edge servers from the [real-world EUA-Dataset](#)
- ML Model datasets: Fashion-MNIST, Speech Commands, AG_NEWS

Parameter	Value or Range
<i>Edge Network Parameter</i>	
# of edge servers N	20 ~ 40
v_i 's storage capacity c_i	512 ~ 1,024GB
v_i 's CPU frequency f_i	2 ~ 5GHz
e_i 's link bandwidth b_i	512 ~ 1,024Mbps
# of different dataset O	5
each dataset size $ S_{i,k} $	1 ~ 3GB
# of time period T	30
<i>Federated Edge Learning Parameter</i>	
# of FEL models W	1 ~ 5
# of m_j 's FEL workers κ_j	1 ~ 7
m_j 's model size μ_j	10 ~ 100MB
m_j 's CPU requirement χ_j	1 ~ 3GHz
m_j 's downloading cost η_j	1 ~ 5
m_j 's global convergence reqs. ς_j	0.001 ~ 0.1
constant FEL variables ϑ_0 and φ_0	15, 4

Methods and Baselines

- **THSO**: three-stage optimization method
- **TWSO**: two-stage optimization method
- **GRDY**: three-stage greedy method
- **RAND**: [randomly](#) generates decisions
- **ROUND**^[12]: selects workers and local convergence rate with [randomized rounding](#)
- **DATA**^[13]: prefers edge servers with [more data](#)
- **LOCAL**^[14]: selects edge servers that [complete the local training first](#)

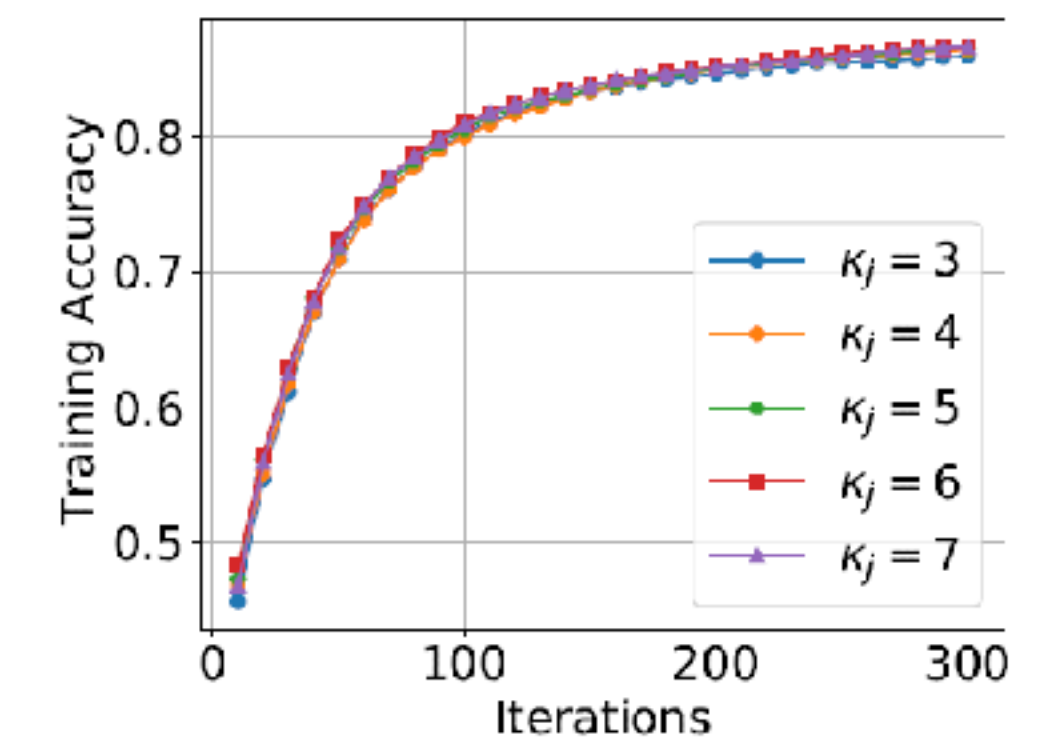
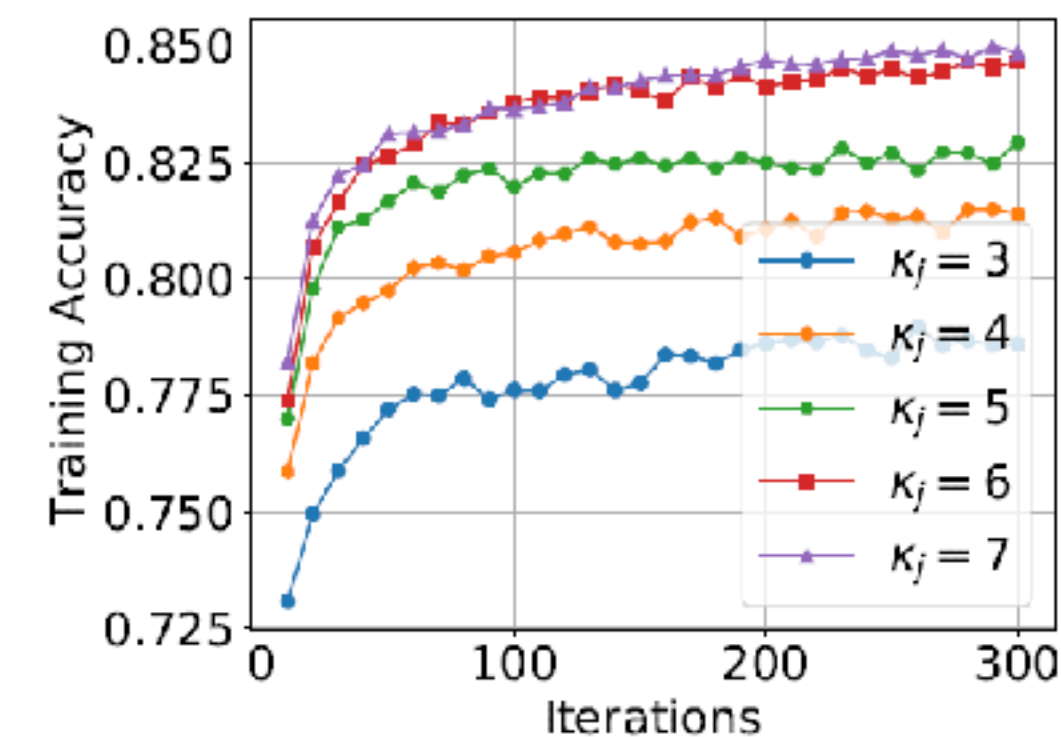
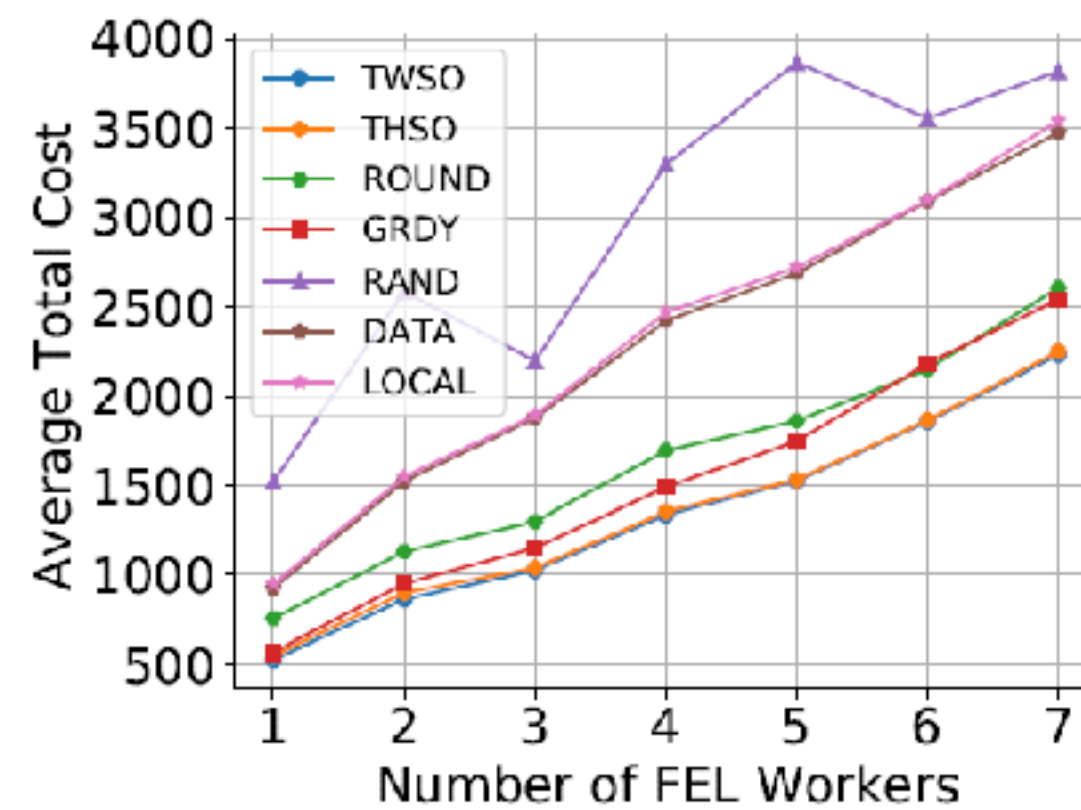
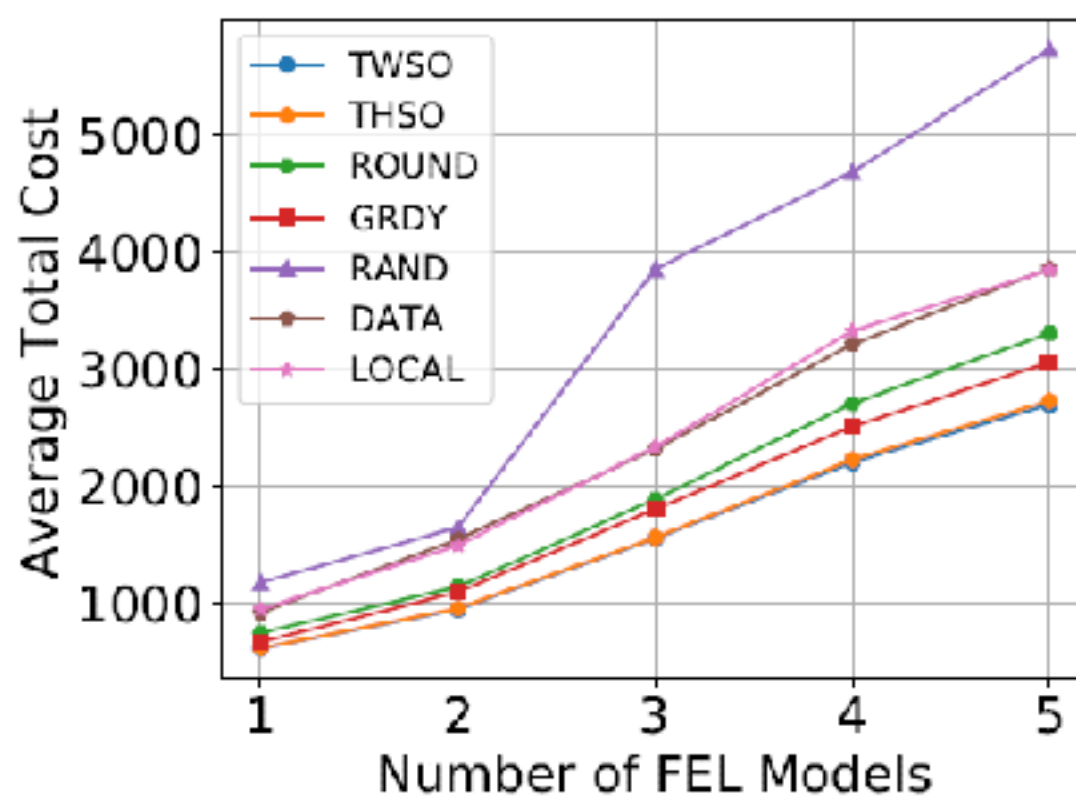
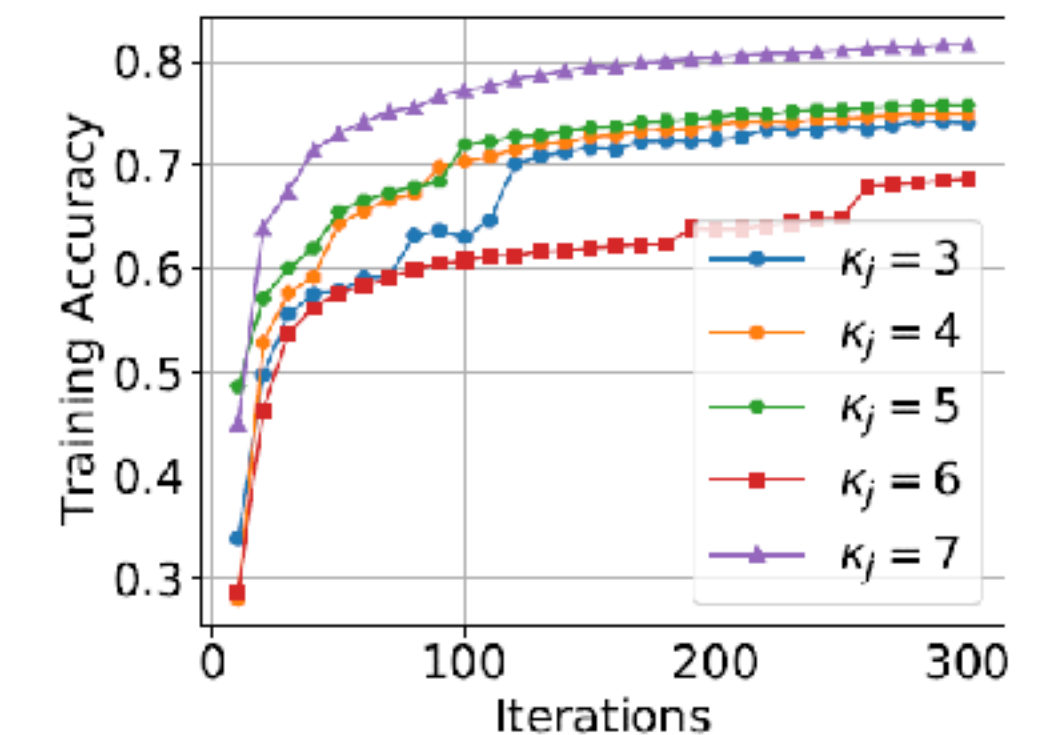
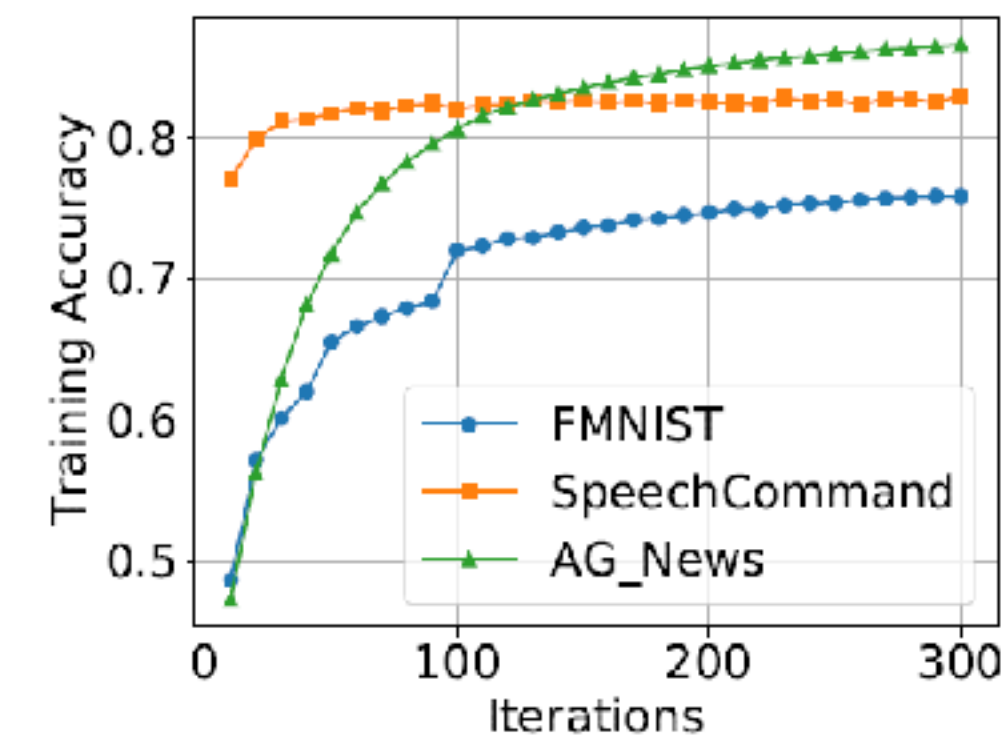
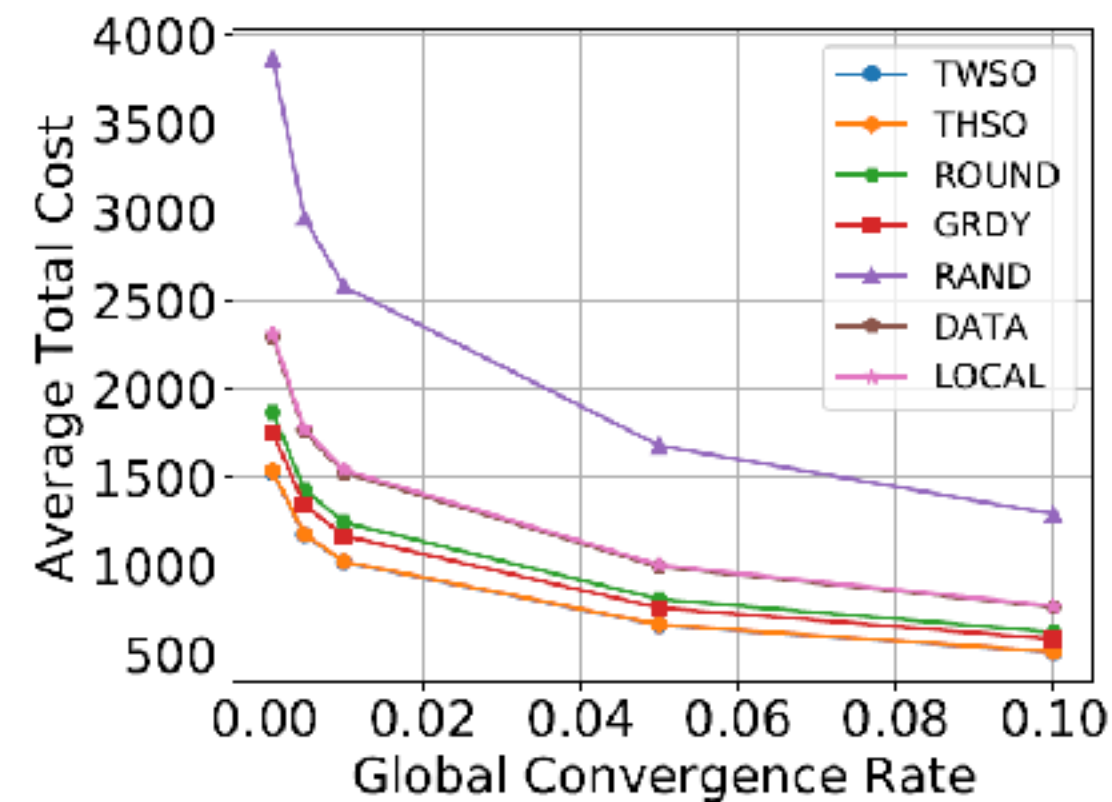
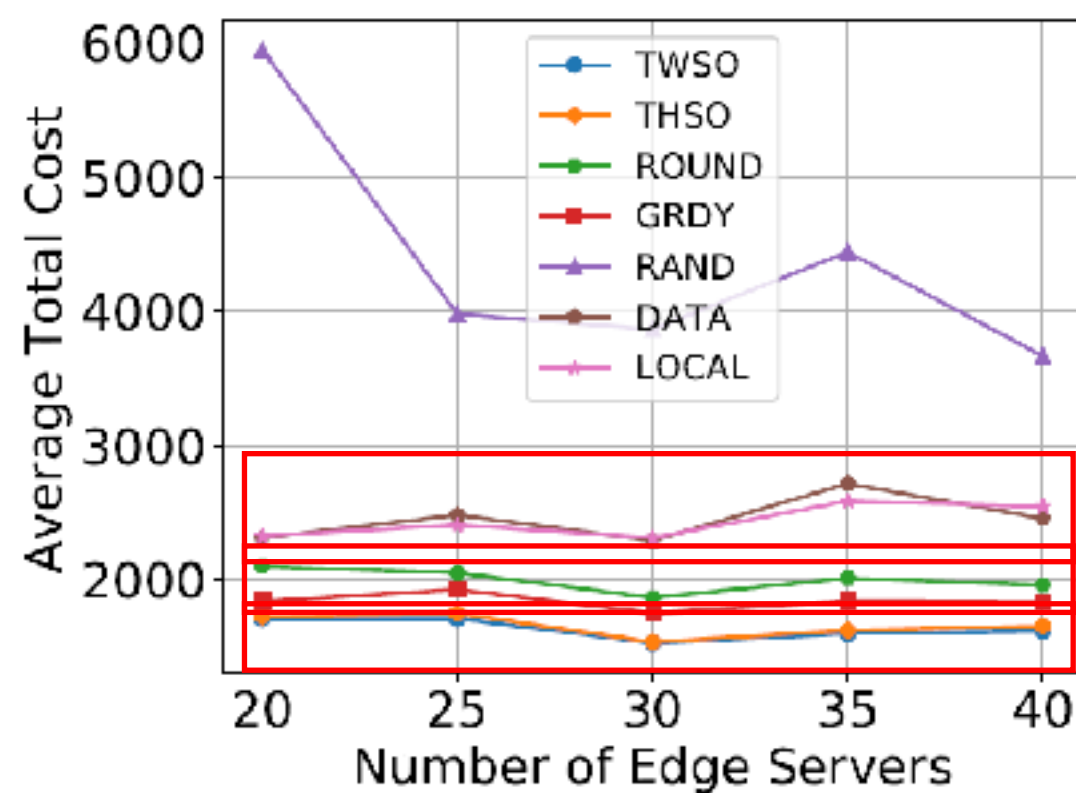
[12] Learning for learning: Predictive online control of federated learning with edge provisioning, IEEE INFOCOM, 2021.

[13] Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, arXiv:2010.01243, 2020.

[14] On the convergence of FedAvg on non-IID data, arXiv preprint arXiv:1907.02189, 2019.

Joint Participant Selection and Scheduling in FEL

Evaluation



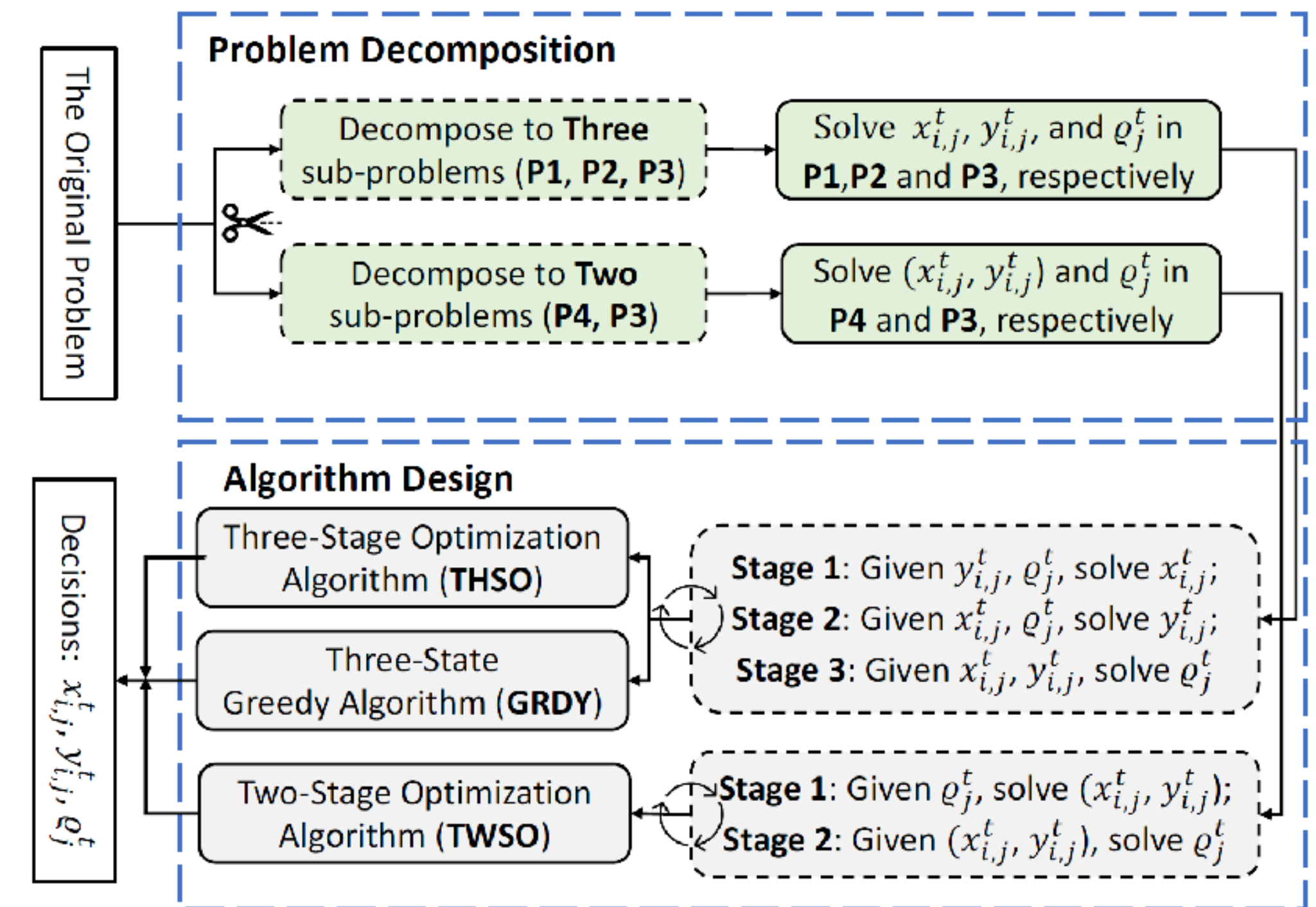
Proposed methods achieve best cost!

FL converges as expected & more workers lead better accuracy

Joint Participant Selection and Scheduling in FEL

Summary

- Study joint participant selection and learning scheduling of multi-model FEL in the edge cloud:
 - select **participants** (both PS and workers) and **local convergence rate** for each FL model
 - aim to minimize the **total learning cost** of all FL models
- Propose three algorithms:
 - **three-stage optimization**, solve one decision via optimization when fixing other two decisions at each stage
 - **three-stage greedy**, greedily make one decision when fixing other two decisions at each stage
 - **two-stage optimization**, consider participant selection in a single stage
- Conduct simulations to evaluate proposed methods:
 - the proposed methods can effectively reduce the total cost compared with existing methods



Outline

Joint Participant Selection and Scheduling in FEL

Joint Participant Selection and Learning Scheduling for Multi-Model Federated Edge Learning

Xinliang Wei, Jiyao Liu, Yu Wang
Department of Computer and Information Sciences, Temple University, Philadelphia, USA
{xinliang.wei,jiyao.liu,wangyu}@temple.edu

Abstract—As edge computing complements the cloud to enable computational services right at the network edge, federated learning (FL) can also benefit from close-by edge computing infrastructure. However, most prior works on federated edge learning (FEL) mainly focus on one shared global model during the federated training in edge systems. In a real edge computing scenario, there may co-exist multiple various FL models that are owned by different entities and used by different applications. Simultaneously training these models competes both computing and networking resources in the shared edge system. Therefore, in this work, we consider a multi-model federated edge learning where multiple FEL models are being trained in the edge network and edge servers can act as either parameter servers or workers of these FEL models. We formulate a joint participant selection and learning scheduling problem, which is a non-linear mixed-integer program, aiming to minimize the total cost of all FEL models while satisfying the desired convergence rate of trained FEL models and the constrained edge resources. We then design several algorithms by decoupling the original problem into two or three sub-problems which can be solved respectively and iteratively. Extensive simulations with real-world training datasets and FEL models show that our proposed algorithms can efficiently reduce the average total cost of all FEL models in a multi-model FEL setting compared with existing algorithms.

1. INTRODUCTION

With the advances of Internet of Things, smart sensing and artificial intelligence, there has been a tremendous trend that data sources shift from the cloud center to the network edge. Generally, in order to train a machine learning (ML) model, one needs to upload the collected training data to the cloud data center and train the model using the whole dataset there. However, it is non-trivial to send a large amount of data to the remote data center due to the limited network bandwidth and data privacy concerns. Therefore, an alternative solution is the distributed training of ML models at the network edge or even on the user devices. However, there are still major challenges to prevent users from performing efficient model training at the edge. On one hand, the computing capacity

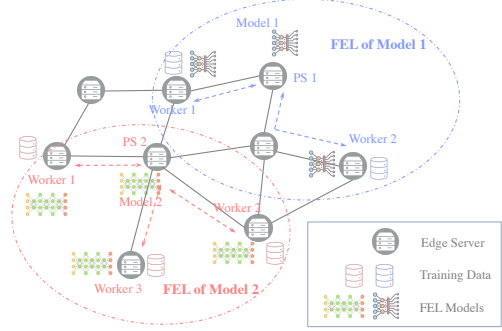


Fig. 1. Multi-model FEL example: two FEL models are trained with 3 and 4 participants (1 PS + 2 or 3 workers), respectively, in a shared edge network.

resources and the competition among various users, servers and applications.

Recently, *federated learning* (FL) has been emerging as a new distributed machine learning paradigm [1]–[3], which enables multiple servers collaboratively learn a shared ML model while keeping all training data on the local server. It is very natural to deploy the FL framework in edge computing to provide efficient distributed training at the network edge. Therefore, *federated edge learning* (FEL) has been proposed in various settings [4]–[12]. In FEL, edge servers can collaboratively train a shared global ML model by aggregating local models trained at individual local servers, decoupling the ability to do model training from the need to store data in 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 local training based on their local data. After that, edge servers will upload their local model to the parameter server which is responsible for aggregating parameters from different workers and sending the

Joint Participant and Topology Selection in FEL

Joint Participant and Learning Topology Selection in Federated Edge Learning

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

Abstract—Deploying federated learning (FL) in edge clouds is a challenging task, particularly when multiple models are trained concurrently in resource-constrained edge environments. Current research on federated edge learning primarily focuses on client selection for training a single FL model with a fixed learning topology. Our experiments demonstrate that FL models with adaptable topologies result in lower learning costs than those with fixed topologies. In this paper, we investigate the problem of jointly selecting participants and learning topologies for multiple FL models being trained simultaneously in the edge cloud. We formulate this as an integer programming problem, with the goal of minimizing total learning costs for all FL models, subject to edge resource constraints. We propose a two-stage algorithm that decouples the original problem into two sub-problems and addresses them iteratively. By allowing FL models to independently select participants and learning topologies, our method improves resource competition and load balancing in edge clouds. Our extensive experiments with real-world networks and FL datasets confirm the superior performance of our algorithm in terms of average total cost compared to prior methods for multi-model FL.

Index Terms—Edge Computing, Federated Learning, Participant Selection, Learning Topology

1 INTRODUCTION

Federated Learning (FL) [1]–[7] is an efficient approach for improving machine learning (ML) performance and providing better privacy solutions for data owners. It enables multiple devices to collaborate and train a shared global ML model by aggregating local models trained on each device. FL ensures that training data remains local to protect users’ privacy, and only transmits essential model data (e.g. gradients). With the growth of smart sensing, mobile computing, and wireless networking, there is also a trend of moving data sources and intelligent computation from centralized clouds to edge clouds, to provide agile services to mobile devices and users. Therefore, it is important to deploy FL frameworks on edge clouds and provide efficient distributed training for mobile devices at the network edge. Such solutions have been studied [8]–[12] and can support many emerging applications [13], such as mobile AI, AIoT or AR/XR applications.

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 [10] where

train the model by using their local data. After each worker performs several local updates, the local model will be forwarded to the PS for global aggregation. The potential bottleneck of CFL is the communication congestion at the PS since all workers have to communicate with the PS concurrently for multiple rounds. In addition, the PS in CFL may cause a single point of failure. Therefore, DFL [11], [14] has been proposed, where each worker only communicates with its neighbors (with mutual trust) by exchanging their local models and there is no centralized PS, as shown in Fig. 1(b). While such distributed P2P learning topology increases the robustness of FL, it might suffer from larger communication costs or slower convergence. Recently, HFL [15]–[17] has been proposed by introducing several middle-layer PSs (or called group leaders) in a hierarchical topology such that each of them only aggregates a group model from workers inside its group and sends the group model to the PS for global aggregation, as shown in Fig. 1(c). HFL can effectively hide local updates submitted by individual workers within a group, thereby enhancing privacy protection from malicious or honest-but-curious PS [15]. HFL can also provide better scalability with larger workers but may increase the total latency due to multiple exchanges

Qutaum-Assistant Federated Learning Scheduling

Quantum Assisted Scheduling Algorithm for Federated Learning in Distributed Networks

Xinliang Wei^{*}, Lei Fan[†], Yuanxiong Guo[‡], Yanming Gong[§], Zhu Han[¶], Yu Wang^{*}
^{*}Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA
[†]Department of Engineering Technology, University of Houston, Houston, TX, USA
[‡]Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, USA
[§]Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX, USA
[¶]Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA
{xinliang.wei, wangyu}@temple.edu, lfan8@central.uh.edu, {yuanxiong.guo, yanmin.gong}@utsa.edu, zhan2@uh.edu.

Abstract—The scheduling problem for federated learning (FL) with multiple models in a distributed network is challenging, as it involves NP-hard mixed-integer nonlinear programming. Moreover, it requires optimal participant selection and learning rate determination among multiple FL models to avoid high training costs and resource competition. To overcome those challenges, in literature the Benders’ decomposition algorithm (BD) can deal with mixed integer problems, however, it still suffers from limited scalability. To address this issue, in this paper, we present the Hybrid Quantum-Classical Benders’ Decomposition (HQCBD) algorithm, which combines the power of quantum and classical computing to solve the joint participant selection and learning scheduling problem in multi-model FL. HQCBD decomposes the optimization problem into a master problem with binary variables and small subproblems with continuous variables. This collaboration maximizes the potential of both quantum and classical computing, and optimizes the complex joint optimization problem. Simulation on the commercial D-Wave quantum annealing machine demonstrates the effectiveness and robustness of the proposed method, with up to 18% improvement of iterations and 81% improvement of computation time over BD algorithm on classical CPUs even at small scales.

Index Terms—Federated learning, participant selection, learning scheduling, hybrid quantum-classical optimization

1. INTRODUCTION

With the use of quantum superposition and entanglement, quantum computing (QC) has demonstrated a quantum advantage over classical computing in random quantum circuit sampling [1], Gaussian boson sampling [2], and combinatorial optimization [3]–[5]. In this paper, by leveraging the parallel computing capability of quantum computing, we focus on designing a new quantum-assisted scheduling algorithm to solve a complex joint participant selection and learning scheduling problem for federated learning (FL) in distributed networks.

Federated learning is emerging as an effective and privacy-preserving machine learning (ML) paradigm [6]–[9], which

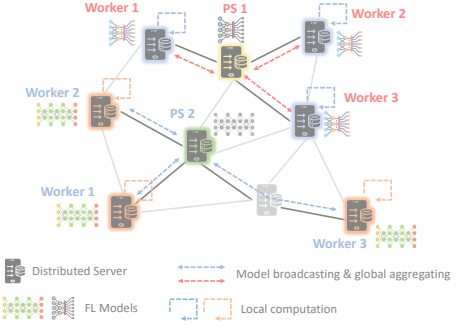
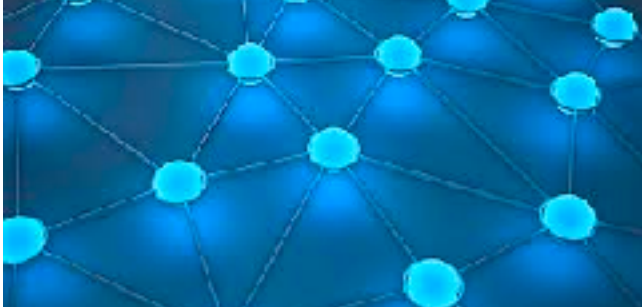


Fig. 1: The training process of multi-model federated learning.

the computing capability and network resources of servers and their data distribution are heterogeneous. Some low-performance servers may decelerate the convergence process and diminish the training performance. Also, the dispersed computing resources and large network latency may lead to high training costs. Second, for the practical scenario, training multiple different models in the shared distributed network simultaneously leads to competition for computing and communication resources. As shown in Fig. 1, two FL models are trained concurrently and each FL model requires one PS and three workers for model training. In this case, which FL model is preferentially served at which server directly affects the total training cost of all FL models. To this end, appropriate participant selection and learning schedules are fairly crucial for multi-model FL training.

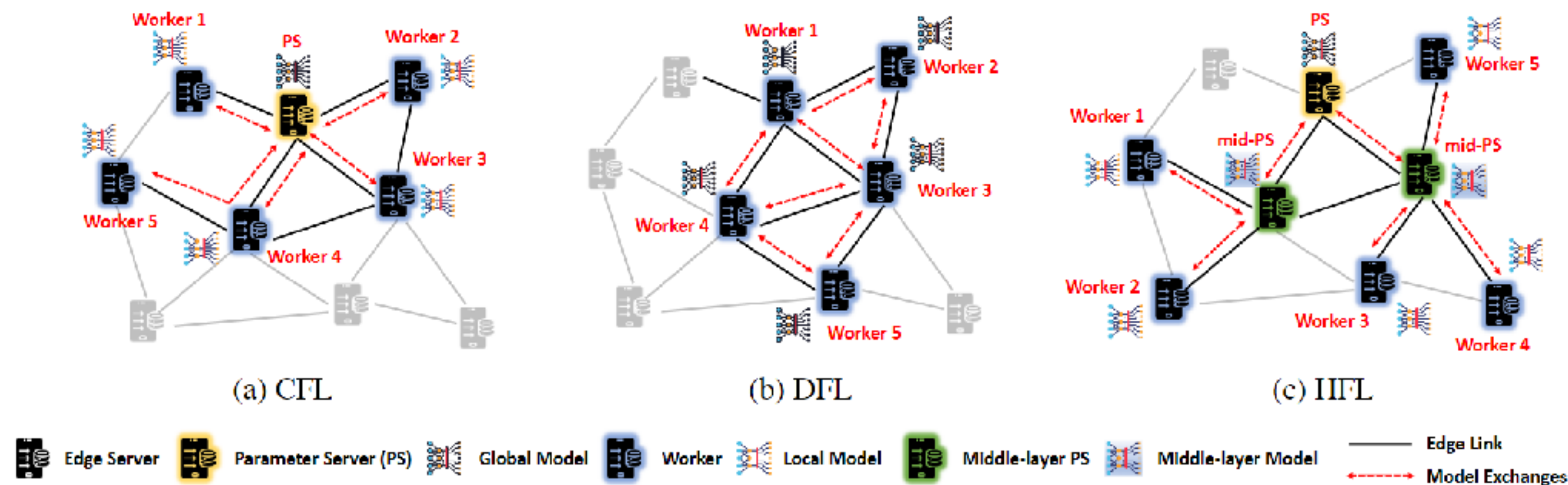
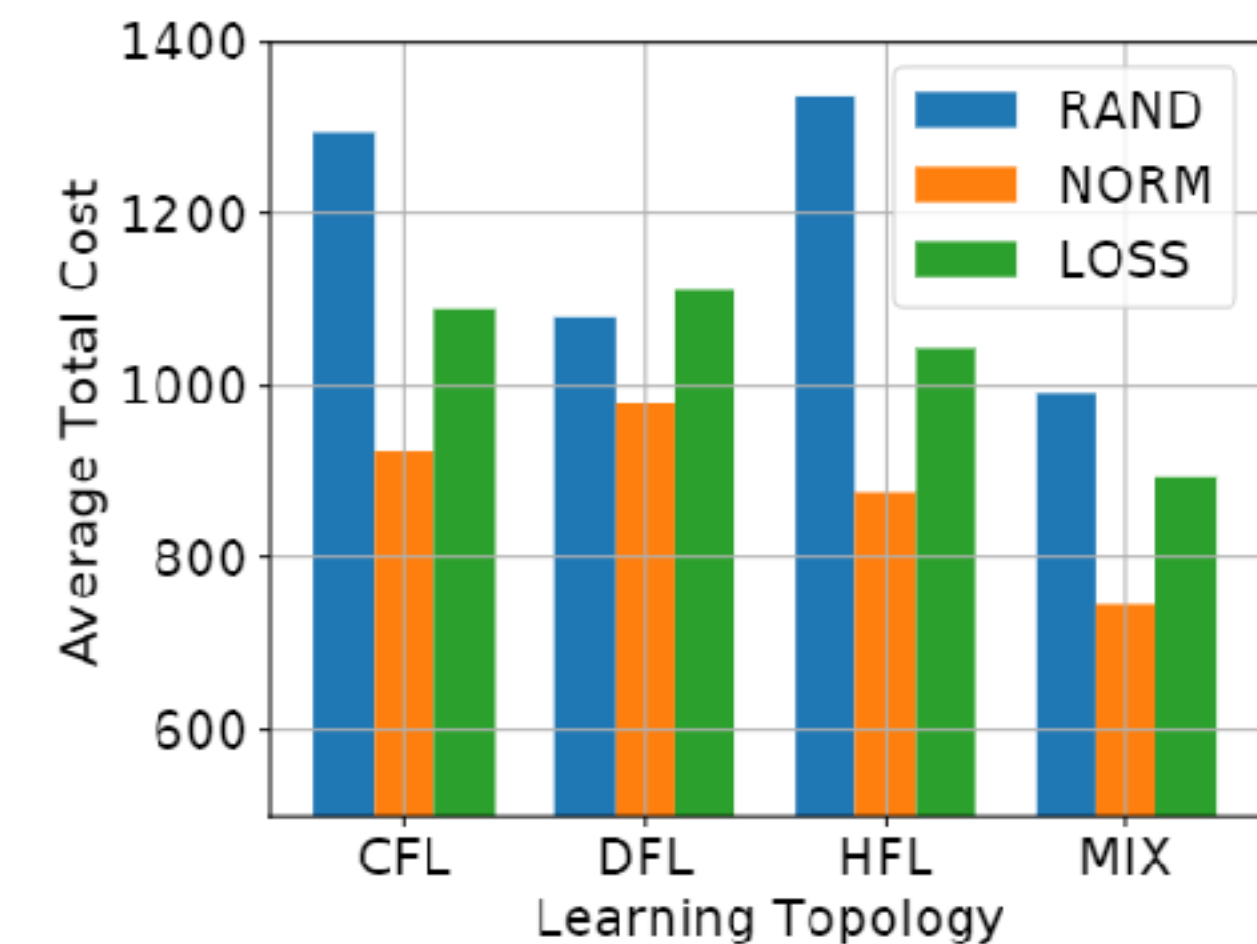
Therefore, we mainly concentrate on the joint participant selection and learning scheduling problem in multi-model FL training scenarios. It should be emphasized that each server in



Joint Participant and Learning Topology Selection

Motivation

- Existing FEL works use a specific **learning topology** for all models
- However, different learning topology (CFL, DFL, and HFL) will lead to different learning costs and performances



Joint participant and learning topology selection

LOSS: [13] Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, arXiv:2010.01243, 2020.
Norm: [15] Client selection in federated learning based on gradients importance, arXiv:2111.11204, 2021

Joint Participant and Learning Topology Selection

FL Model

- Each FL model needs to
 - select learning topology from CFL, DFL, HFL $b_{j,1}, b_{j,2}, b_{j,3}$
 - select FL participants, including PS (or mid-PS) and workers $a_{i,j}, x_{i,j}, y_{i,j}, z_{i,j}$

FL Convergence Bound

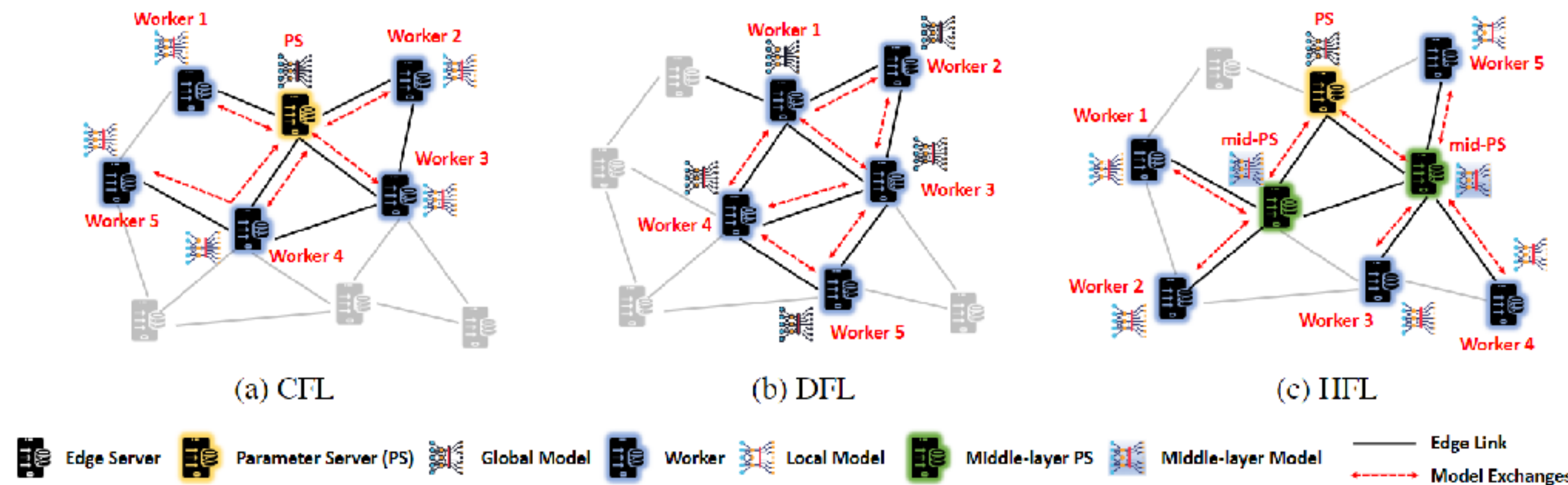
Lemma 1: Analogous to previous works [1], [9], [12] with Assumption 1 and the local convergence rate (Equ. (19)), the number of local training to achieve a θ -local convergence accuracy is

$$\hat{\beta} \geq \frac{2}{(2 - \zeta\eta)\eta\varepsilon} \log \frac{1}{\theta}, \quad (20)$$

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

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

- [1] Client-edge-cloud hierarchical federated learning, ICC, 2020.
 [9] Client selection for federated learning with heterogeneous resources in mobile edge, ICC 2019
 [12] Learning for learning: Predictive online control of federated learning with edge provisioning, IEEE INFOCOM, 2021.



Joint Participant and Learning Topology Selection

Problem Formulation

Learning cost: computation + communication

of global aggregation participant selection

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

learning topology selection

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

of local updates

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

$$C_{i,k}^{comm} = \sum_{e_l \in P_{i,k}} \frac{\mu_j}{b_l}$$

PS assignment

participant & topology selection

- Joint participant and topology selection

$$\min C^{comp} + C^{comm} \quad \text{storage, CPU constraints} \quad (10)$$

$$\text{s.t. } a_{i,j} \mu_j \leq c_i, \quad a_{i,j} \chi_j \leq f_i, \quad \forall i, j \quad (11)$$

$$\sum_i a_{i,j} = \kappa_j + b_{j,1} + (1 + \psi_j) b_{j,3}, \quad \forall j \quad (12)$$

of participants

$$\sum_j a_{i,j} \leq 1, \quad \forall i \quad (13)$$

each participant only works for one model

$$\sum_i x_{i,j} b_{j,1} + 1 - b_{j,1} = 1, \quad \forall j \quad (14)$$

of PS/mid-PSs needed

$$\sum_i y_{i,j} b_{j,3} + 1 - b_{j,3} = 1, \quad \forall j \quad (15)$$

$$\sum_i z_{i,j} b_{j,3} + (1 - b_{j,3}) \psi_j = \psi_j, \quad \forall j \quad (16)$$

$$\sum_j (y_{i,j} + z_{i,j}) b_{j,3} \leq 1, \quad \forall i \quad (17)$$

PS & mid-PS are different for HFL

$$\sum_k b_{j,k} = 1, \quad \forall j \quad (18)$$

each model needs a learning topology

$$a_{i,j} \in \{0, 1\}, b_{j,k} \in \{0, 1\}, \quad (19)$$

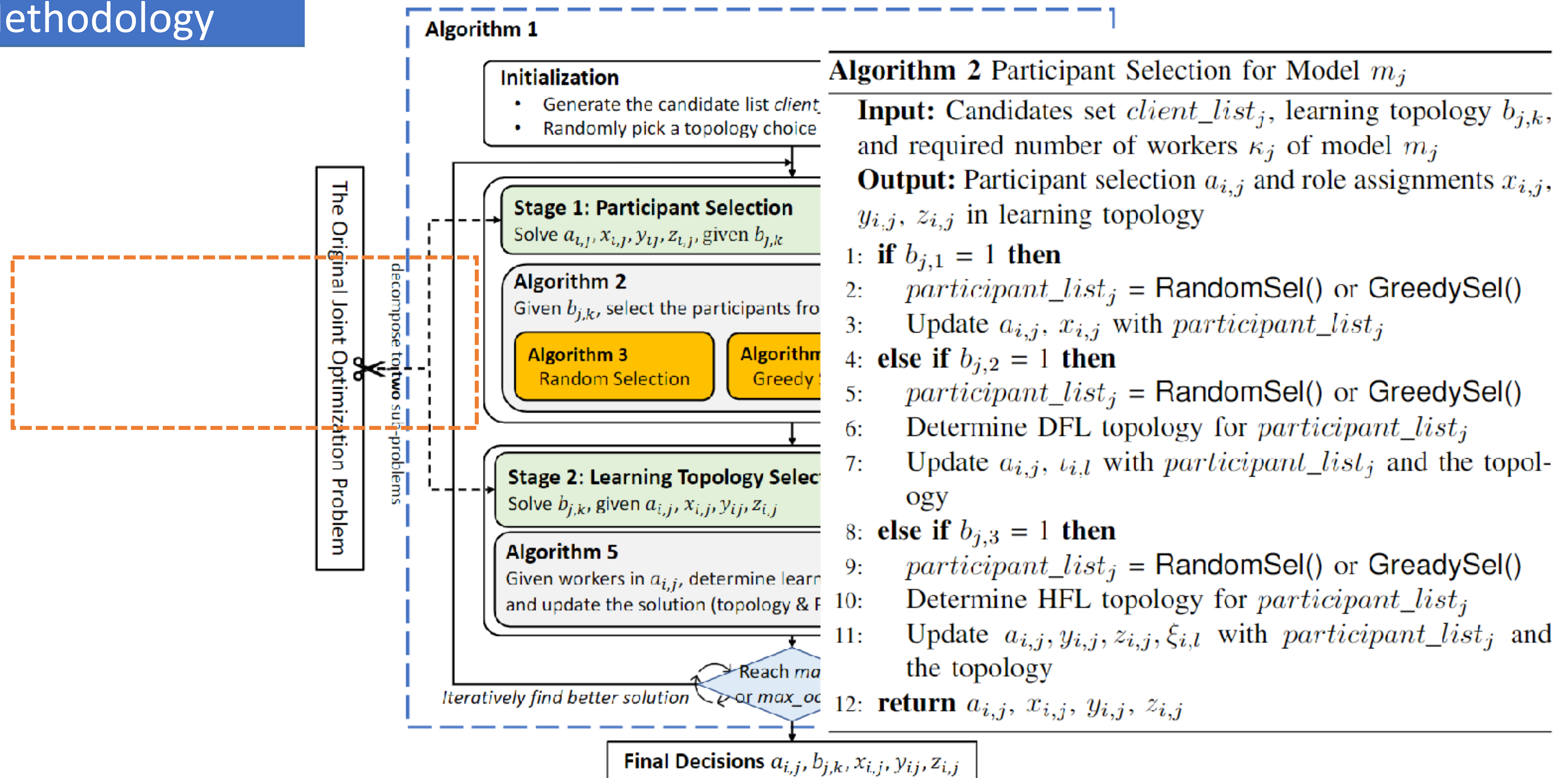
$$x_{i,j} \in \{0, 1\}, y_{i,j} \in \{0, 1\}, z_{i,j} \in \{0, 1\}, \quad (20)$$

$$i \in [1, \dots, N], j \in [1, \dots, H], k \in [1, \dots, 3] \quad (21)$$

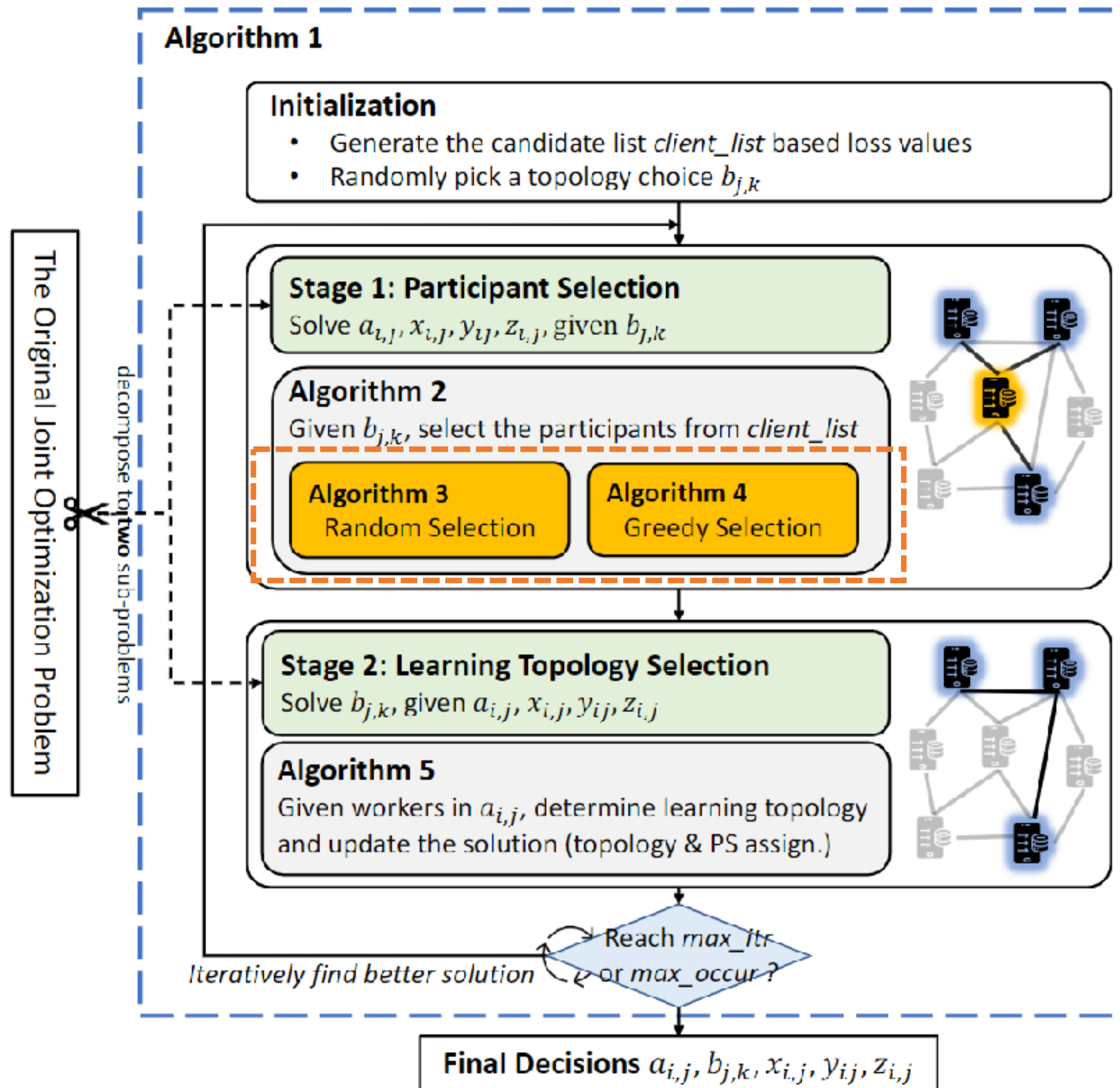
PS assignment

Joint Participant and Learning Topology Selection

Methodology



Joint Participant and Learning Topology 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)

- $worker_list = \text{random}(client_list_j, \kappa_j)$
- if** $b_{j,1} = 1$ **then**
- ps = the closest one to the selected workers among all remaining edge servers outside $worker_list$
- else if** $b_{j,3} = 1$ **then**
- mid_ps = the closest ψ_j servers to the selected workers among servers outside $worker_list$
- top_ps = the $(\psi_j + 1)$ -th closest server to the selected workers among servers outside $worker_list$
- return** $participant_list_j$ (i.e., $worker_list$ and PS lists)

Algorithm 4 GreedySel() - Greedy Selection

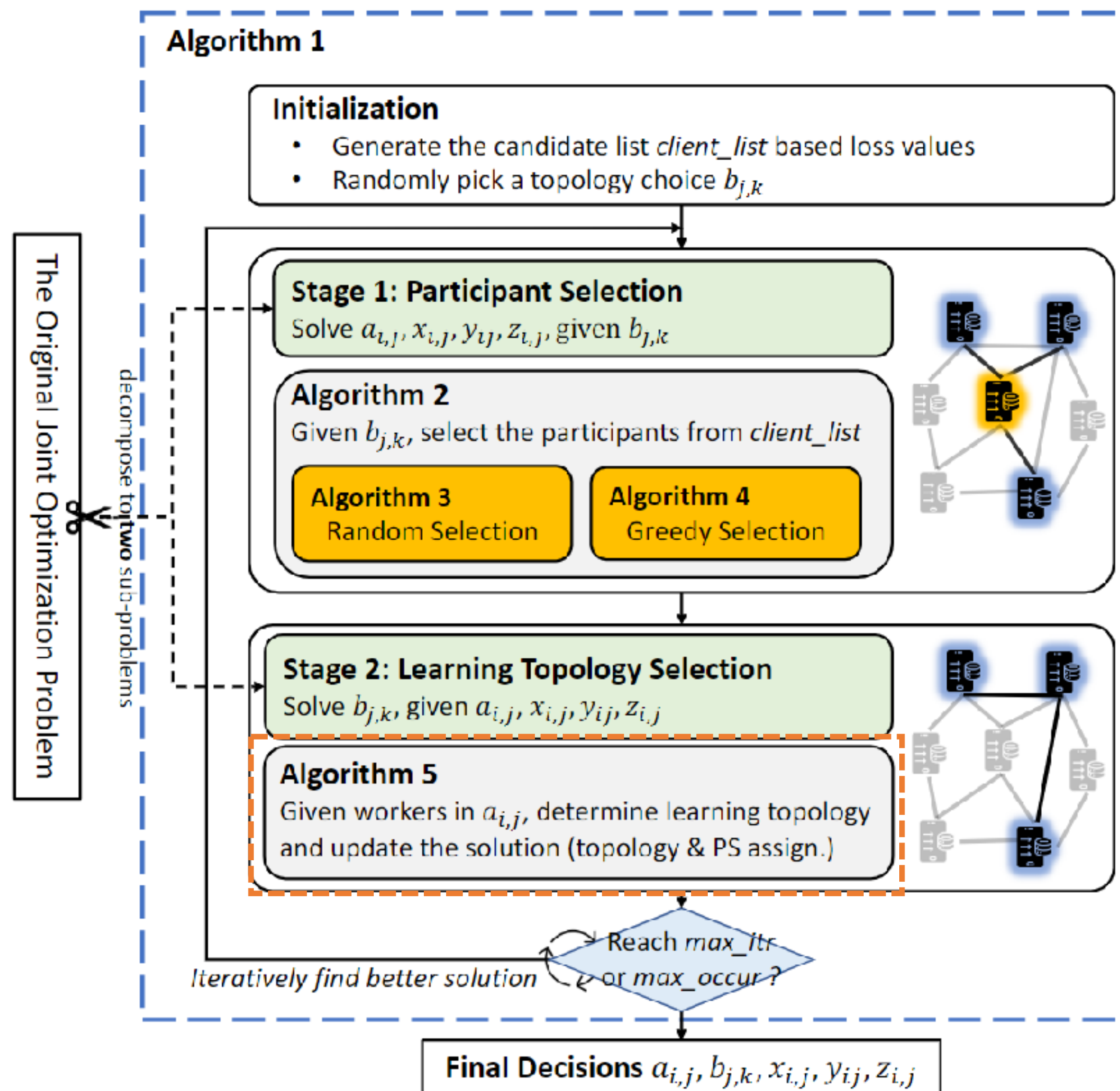
Input: Candidates set $client_list_j$, learning topology $b_{j,k}$, and required number of workers κ_j of model m_j

Output: Participant list $participant_list_j$

- $worker_list = \emptyset$
- ps = the closest server outside $client_list_j$ to $client_list_j$
- mid_ps, top_ps = the closest $(\psi_j + 1)$ edge servers outside $client_list_j$ to $client_list_j$
- for** $i = 1$ to κ_j **do**
- for** c_i in $client_list_j$ but not in $worker_list$ **do**
- Calculate $total_cost(c_i)$ if adding c_i to $worker_list$ based on the topology choice $b_{j,k}$
- Let c_i^* be the one leading to minimal $total_cost(c_i)$, then add c_i^* to $worker_list$
- return** $participant_list_j$ (i.e., $worker_list$ and PS lists)

FL workers
selection

Joint Participant and Learning Topology Selection



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

Joint Participant and Learning Topology Selection

Evaluation

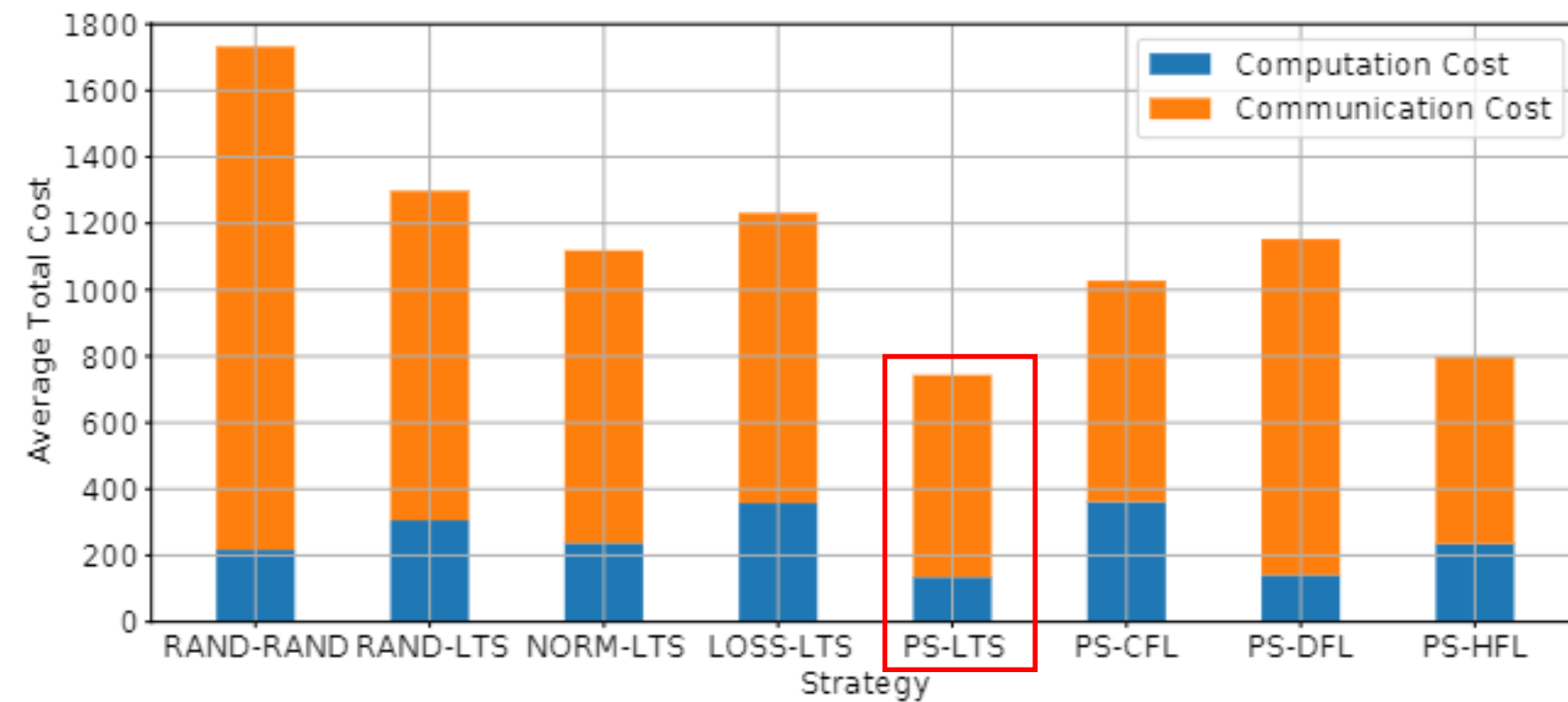
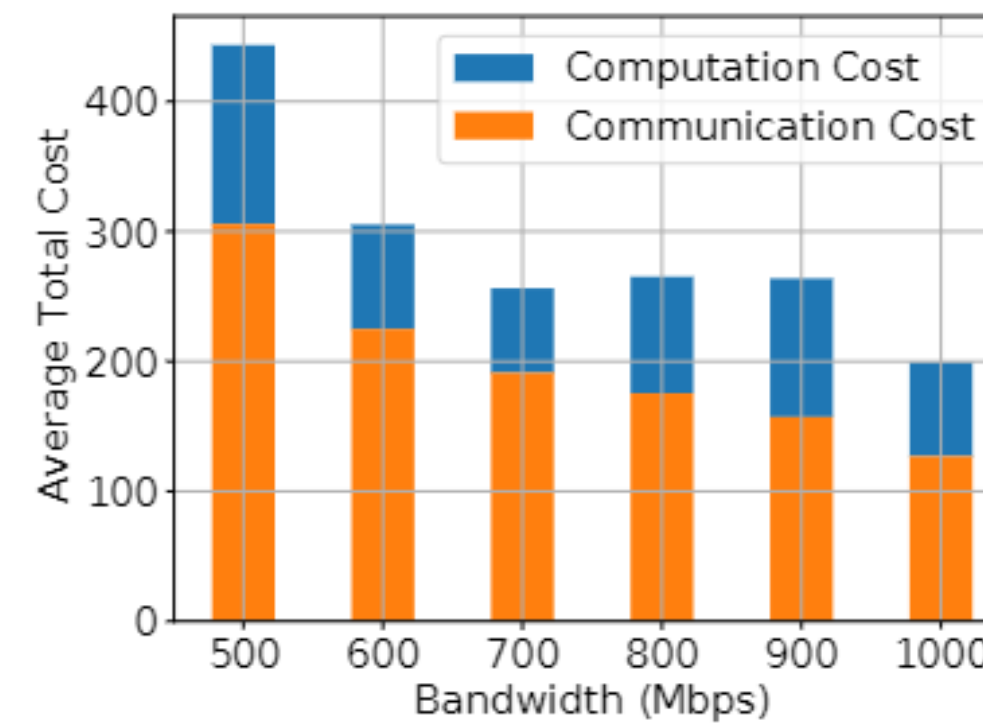


Fig. 4. Performance comparison (i.e., total cost = computation cost + communication cost) of all methods.

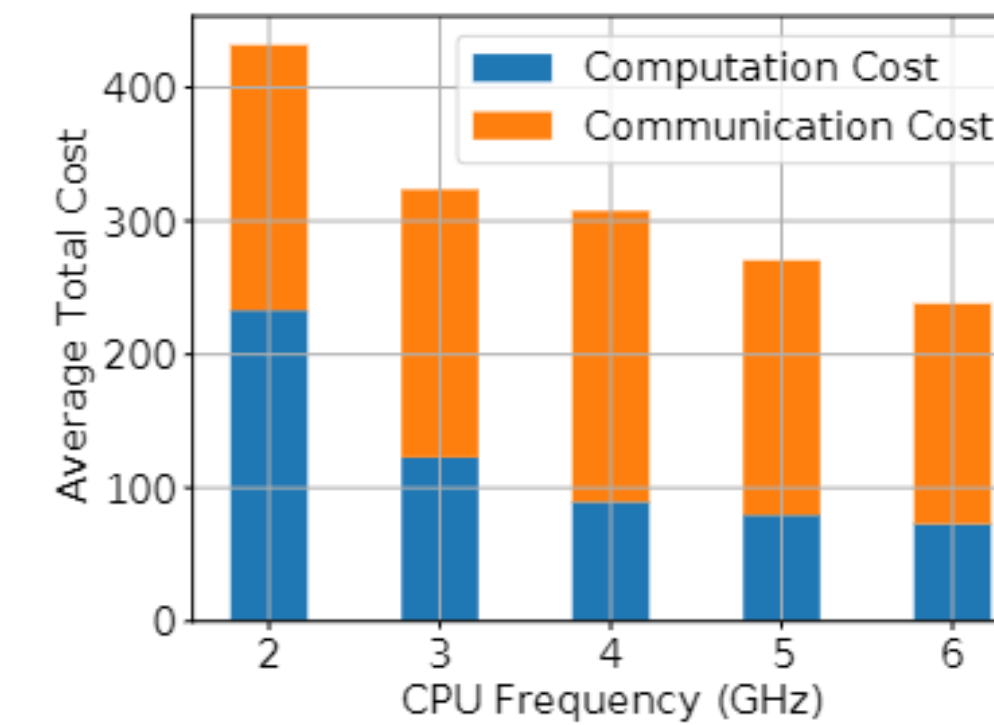
Proposed method achieve best cost!

Methods and Baselines

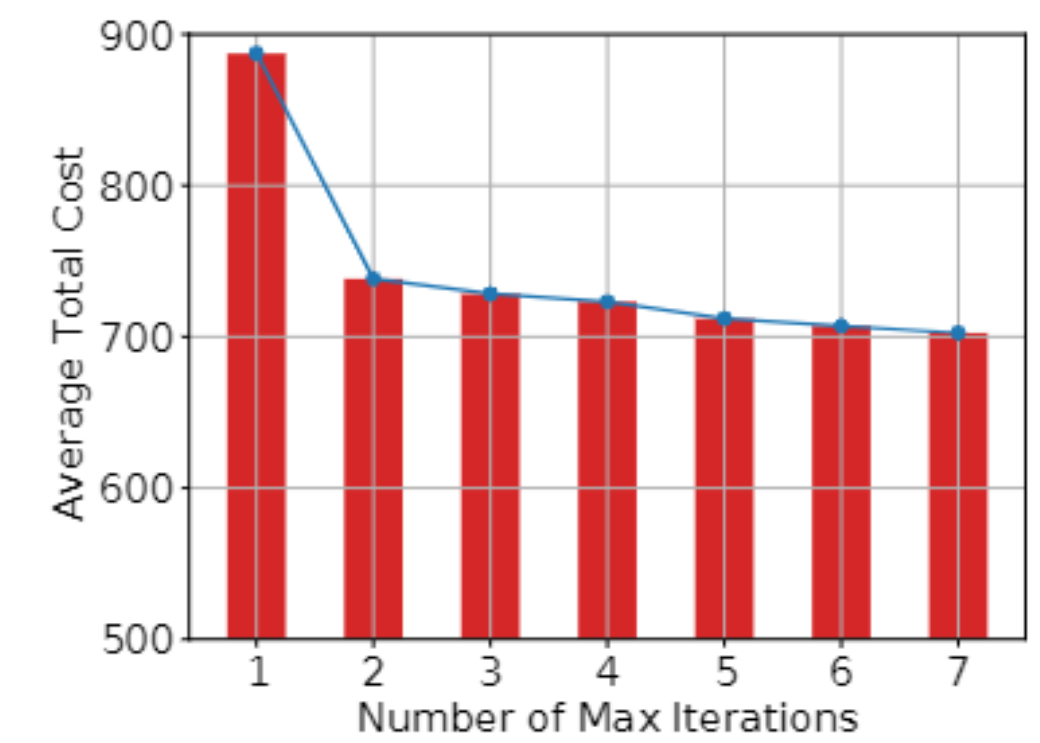
- **PS-LTS**: the proposed method
- **PS-CFL**: adopt **CFL** learning topology
- **PS-DFL**: adopt **DFL** learning topology
- **PS-HFL**: adopt **HFL** learning topology
- **RAND-LTS**: **randomly** select participants
- **NORM-LTS**: select workers with **highest gradient norm values** [15]
- **LOSS-LTS**: select workers based on the fraction of data and **highest loss values** [13]



(a)

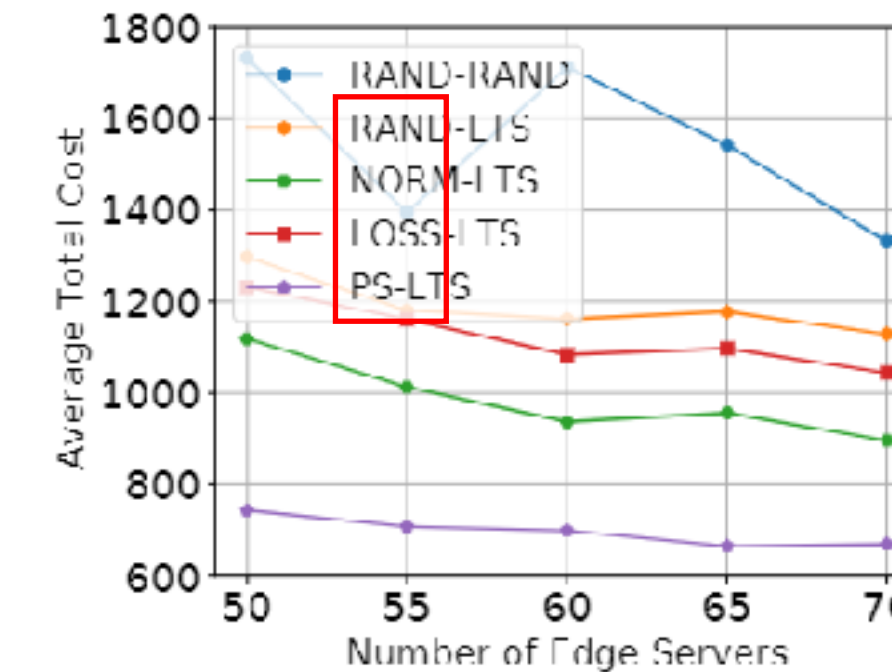


(b)

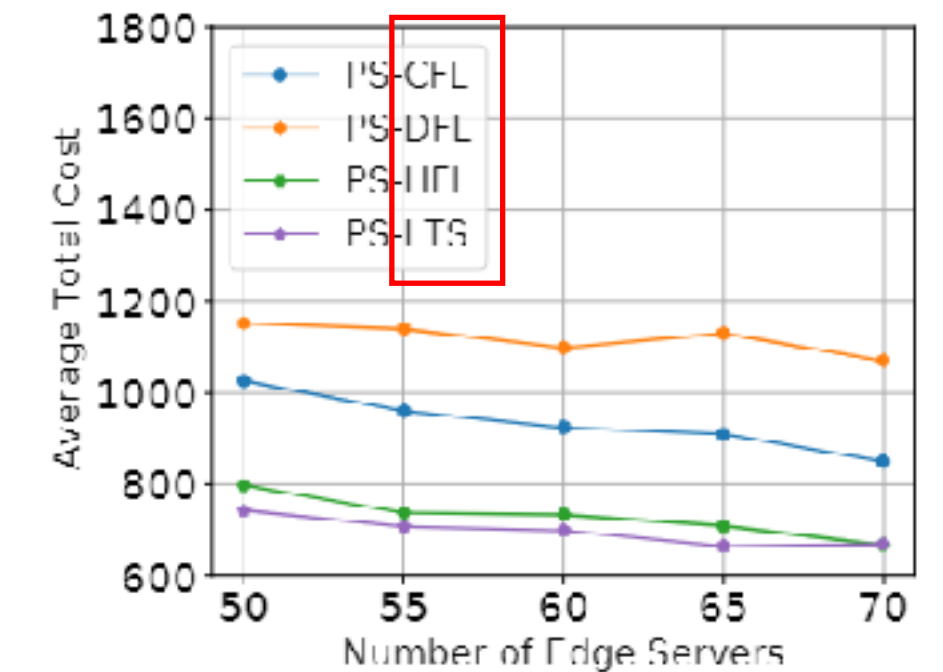


(c)

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



(a) total cost - group 1



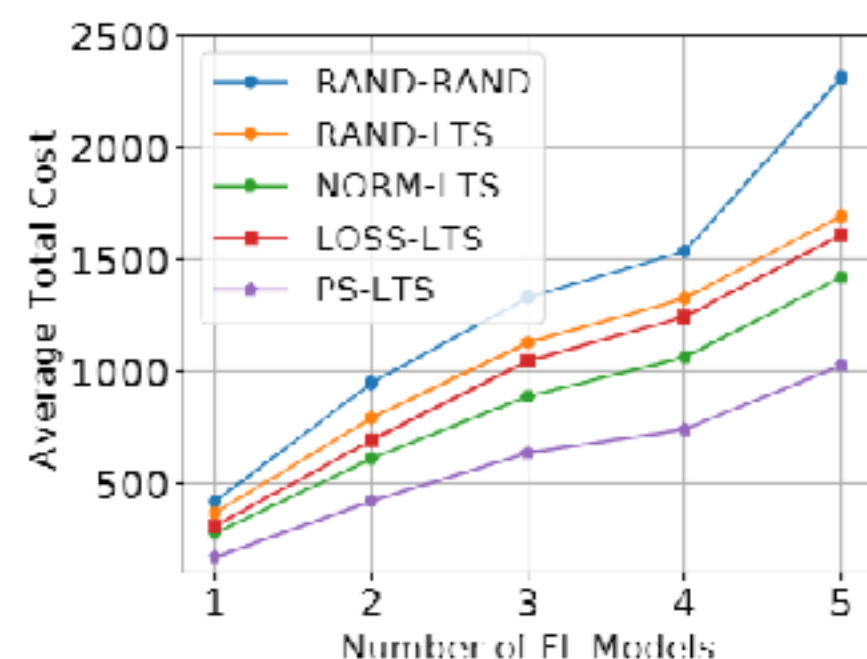
(b) total cost - group 2

Fig. 6. Impact of number of edge servers on costs

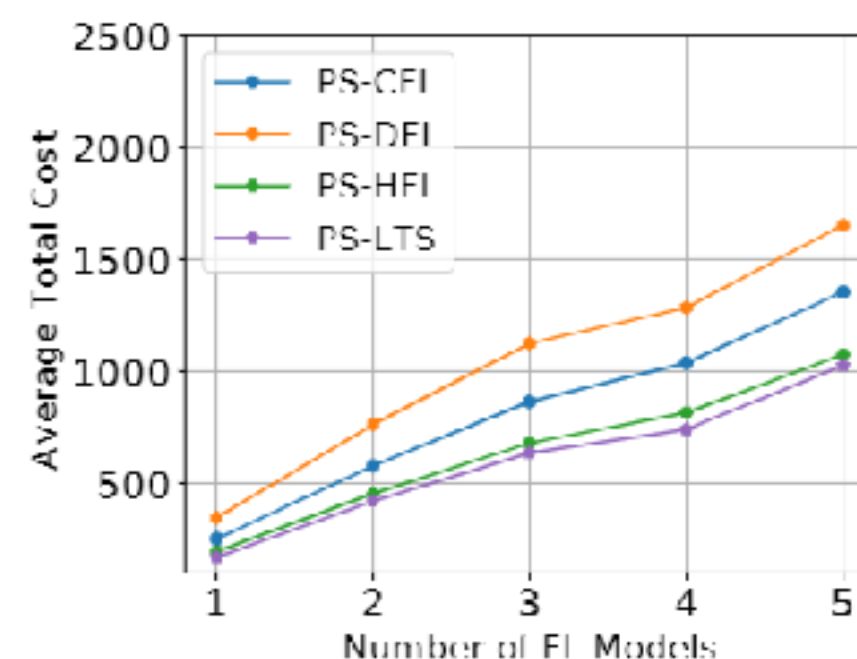
Joint Participant and Learning Topology Selection

Evaluation

More models -> higher cost & more workers -> higher cost, but better accuracy

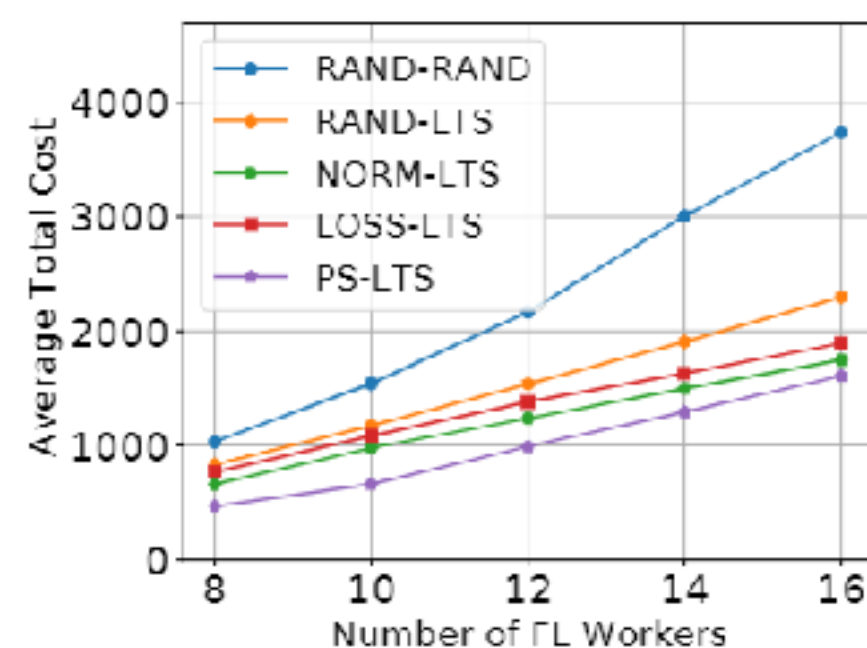


(a) total cost - group 1

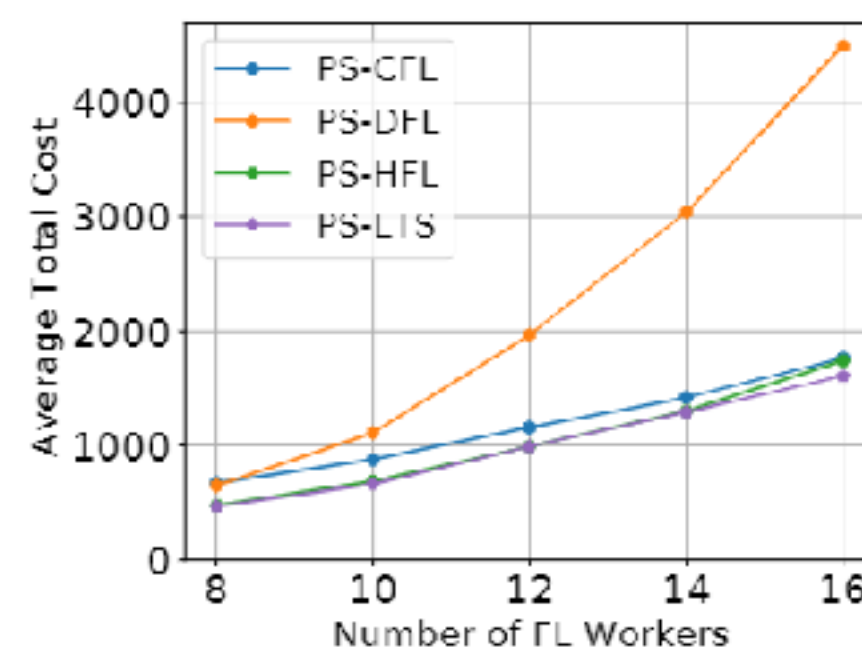


(b) total cost - group 2

Fig. 7. Impact of number of models on costs

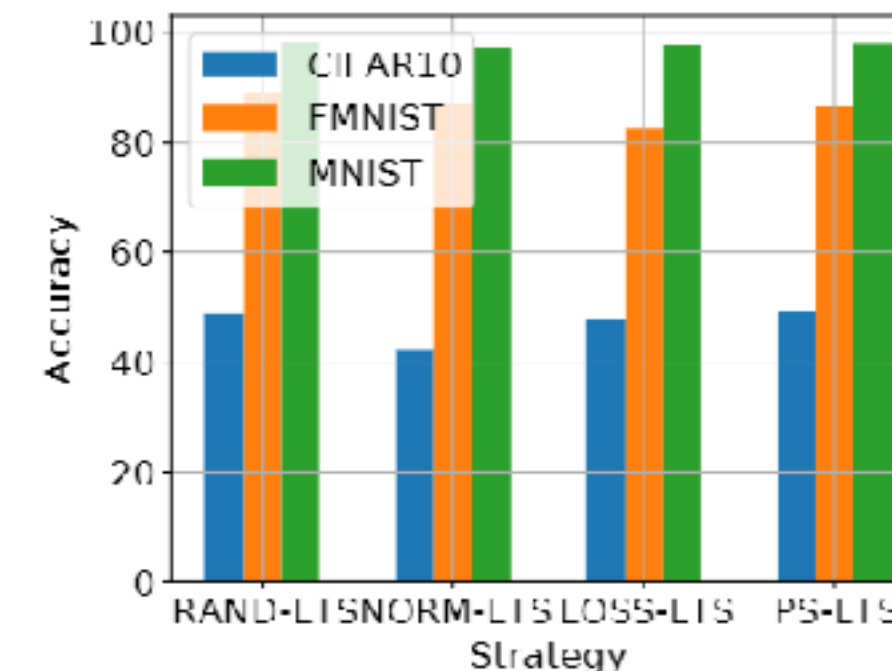


(a) total cost - group 1

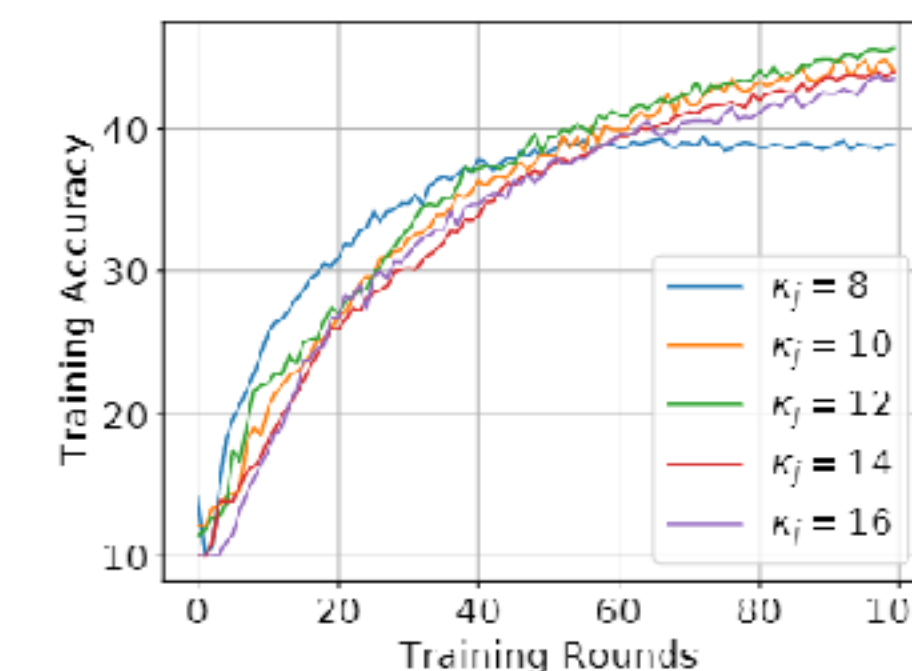


(b) total cost - group 2

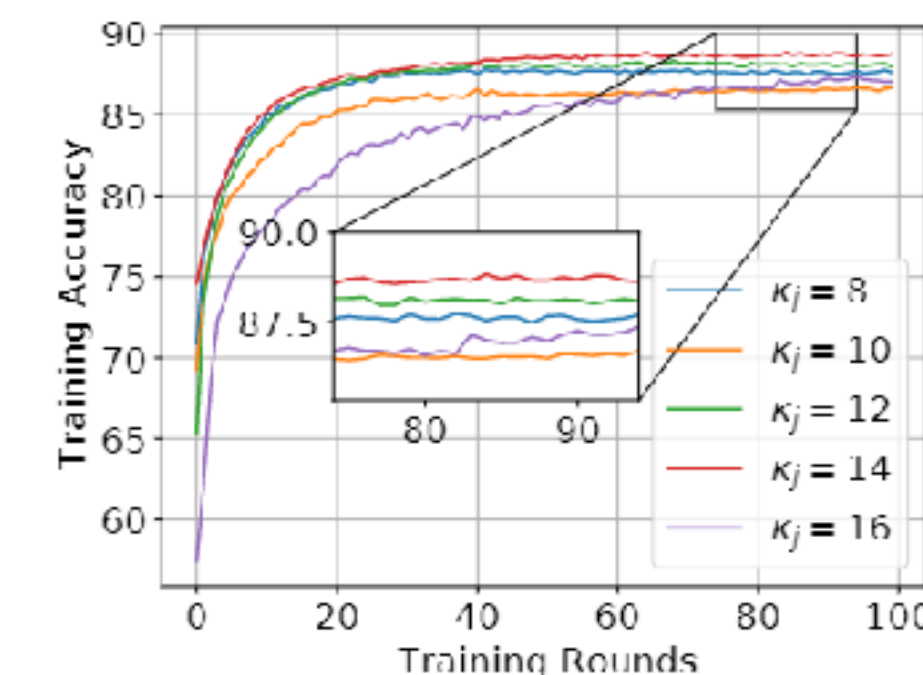
Fig. 8. Impact of number of workers on costs



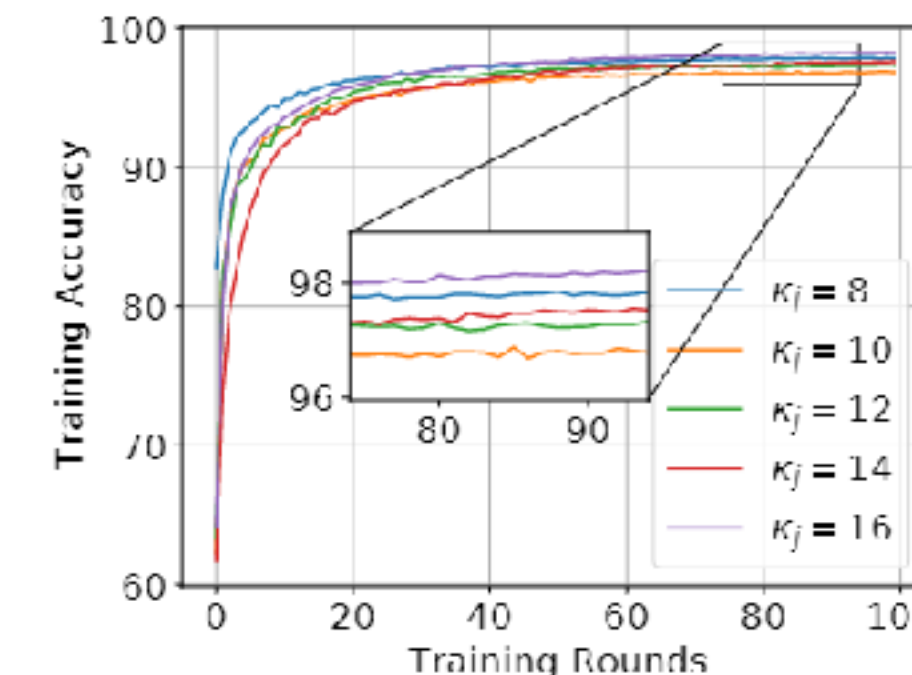
(a) with 10 workers



(b) CNN over CIFAR10



(c) CNN over FMNIST



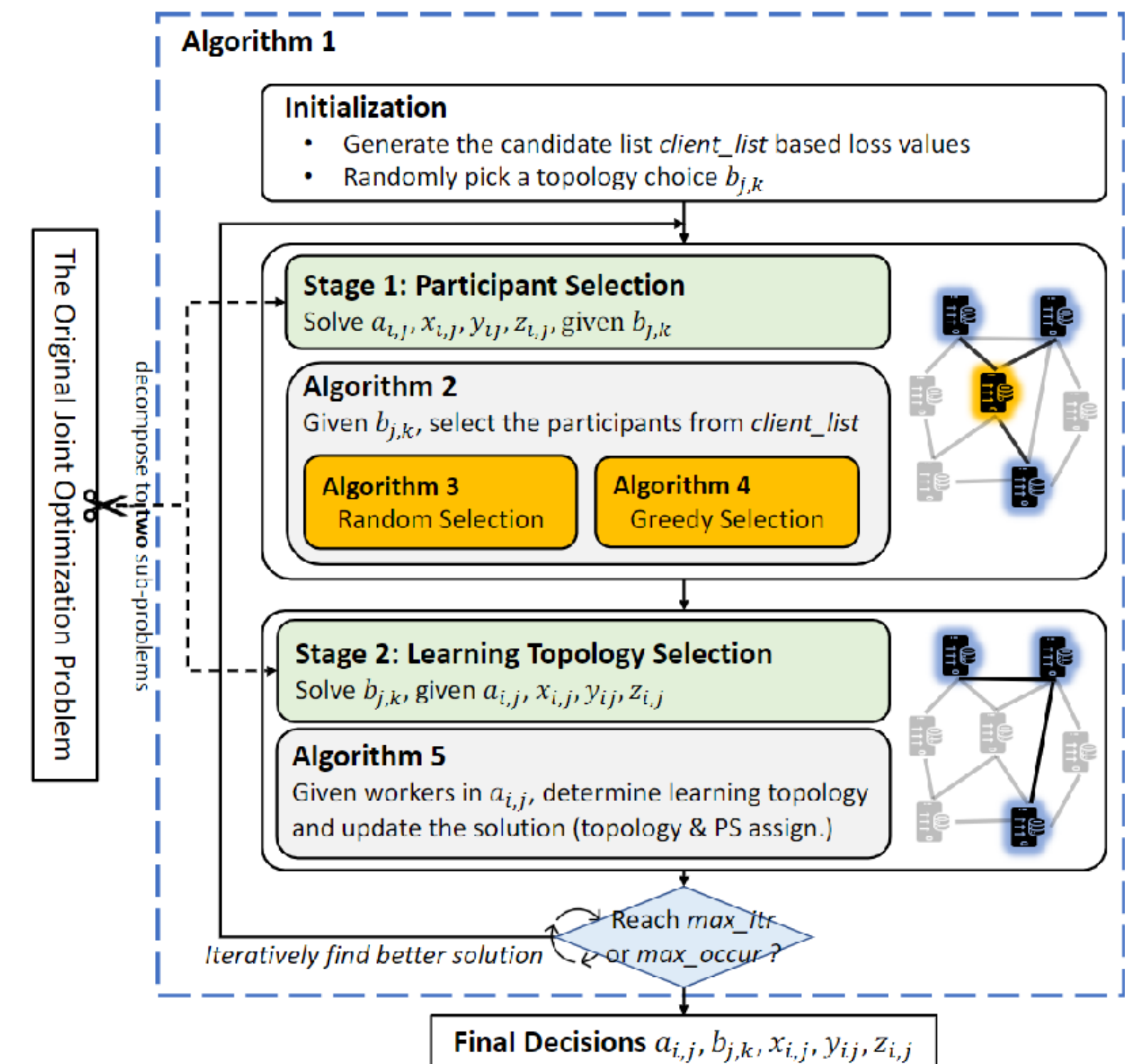
(d) CNN over MNIST

Fig. 9. Training accuracy and impact from number of workers: (a) accuracy of all strategies with 10 workers; (b)-(d) accuracy of three models using our method with different number of workers.

Joint Participant and Learning Topology Selection

Summary

- Study joint participant and learning topology selection of multi-model FEL in the edge cloud:
 - ▶ select **participants** (both PS/midPS and workers) and **learning topology** (CFL, DFL, HFL) for each FL model
 - ▶ aim to minimize the **total learning cost** of all FL models
- Propose two-stage iterative algorithm:
 - ▶ Stage 1 – **participant selection**, two additional selection strategies (Random, Greedy) based on high loss value
 - ▶ Stage 2 – **learning topology selection**, pick the learning topology with the minimum cost for each model
- Conduct simulations to evaluate proposed methods:
 - ▶ the proposed methods can effectively reduce the total cost compared with existing methods



Outline

Joint Participant Selection and Scheduling in FEL

Joint Participant Selection and Learning Scheduling for Multi-Model Federated Edge Learning

Xinliang Wei, Jiyao Liu, Yu Wang
Department of Computer and Information Sciences, Temple University, Philadelphia, USA
{xinliang.wei,jiyao.liu,wangyu}@temple.edu

Abstract—As edge computing complements the cloud to enable computational services right at the network edge, federated learning (FL) can also benefit from close-by edge computing infrastructure. However, most prior works on federated edge learning (FEL) mainly focus on one shared global model during the federated training in edge systems. In a real edge computing scenario, there may co-exist multiple various FL models that are owned by different entities and used by different applications. Simultaneously training these models competes both computing and networking resources in the shared edge system. Therefore, in this work, we consider a multi-model federated edge learning where multiple FEL models are being trained in the edge network and edge servers can act as either parameter servers or workers of these FEL models. We formulate a joint participant selection and learning scheduling problem, which is a non-linear mixed-integer program, aiming to minimize the total cost of all FEL models while satisfying the desired convergence rate of trained FEL models and the constrained edge resources. We then design several algorithms by decoupling the original problem into two or three sub-problems which can be solved respectively and iteratively. Extensive simulations with real-world training datasets and FEL models show that our proposed algorithms can efficiently reduce the average total cost of all FEL models in a multi-model FEL setting compared with existing algorithms.

1. INTRODUCTION

With the advances of Internet of Things, smart sensing and artificial intelligence, there has been a tremendous trend that data sources shift from the cloud center to the network edge. Generally, in order to train a machine learning (ML) model, one needs to upload the collected training data to the cloud data center and train the model using the whole dataset there. However, it is non-trivial to send a large amount of data to the remote data center due to the limited network bandwidth and data privacy concerns. Therefore, an alternative solution is the distributed training of ML models at the network edge or even on the user devices. However, there are still major challenges to prevent users from performing efficient model training at the edge. On one hand, the computing capacity

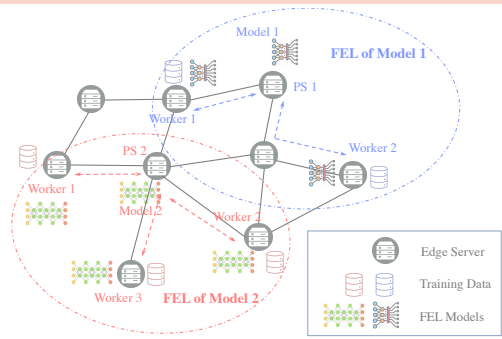


Fig. 1. Multi-model FEL example: two FEL models are trained with 3 and 4 participants (1 PS + 2 or 3 workers), respectively, in a shared edge network.

resources and the competition among various users, servers and applications.

Recently, *federated learning* (FL) has been emerging as a new distributed machine learning paradigm [1]–[3], which enables multiple servers collaboratively learn a shared ML model while keeping all training data on the local server. It is very natural to deploy the FL framework in edge computing to provide efficient distributed training at the network edge. Therefore, *federated edge learning* (FEL) has been proposed in various settings [4]–[12]. In FEL, edge servers can collaboratively train a shared global ML model by aggregating local models trained at individual local servers, decoupling the ability to do model training from the need to store data in 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 local training based on their local data. After that, edge servers will upload their local model to the parameter server which is responsible for aggregating parameters from different workers and sending the

Joint Participant and Topology Selection in FEL

Joint Participant and Learning Topology Selection in Federated Edge Learning

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

Abstract—Deploying federated learning (FL) in edge clouds is a challenging task, particularly when multiple models are trained concurrently in resource-constrained edge environments. Current research on federated edge learning primarily focuses on client selection for training a single FL model with a fixed learning topology. Our experiments demonstrate that FL models with adaptable topologies result in lower learning costs than those with fixed topologies. In this paper, we investigate the problem of jointly selecting participants and learning topologies for multiple FL models being trained simultaneously in the edge cloud. We formulate this as an integer programming problem, with the goal of minimizing total learning costs for all FL models, subject to edge resource constraints. We propose a two-stage algorithm that decouples the original problem into two sub-problems and addresses them iteratively. By allowing FL models to independently select participants and learning topologies, our method improves resource competition and load balancing in edge clouds. Our extensive experiments with real-world networks and FL datasets confirm the superior performance of our algorithm in terms of average total cost compared to prior methods for multi-model FL.

Index Terms—Edge Computing, Federated Learning, Participant Selection, Learning Topology

1 INTRODUCTION

Federated Learning (FL) [1]–[7] is an efficient approach for improving machine learning (ML) performance and providing better privacy solutions for data owners. It enables multiple devices to collaborate and train a shared global ML model by aggregating local models trained on each device. FL ensures that training data remains local to protect users’ privacy, and only transmits essential model data (e.g. gradients). With the growth of smart sensing, mobile computing, and wireless networking, there is also a trend of moving data sources and intelligent computation from centralized clouds to edge clouds, to provide agile services to mobile devices and users. Therefore, it is important to deploy FL frameworks on edge clouds and provide efficient distributed training for mobile devices at the network edge. Such solutions have been studied [8]–[12] and can support many emerging applications [13], such as mobile AI, AIoT or AR/XR applications.

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 [10] where

train the model by using their local data. After each worker performs several local updates, the local model will be forwarded to the PS for global aggregation. The potential bottleneck of CFL is the communication congestion at the PS since all workers have to communicate with the PS concurrently for multiple rounds. In addition, the PS in CFL may cause a single point of failure. Therefore, DFL [11], [14] has been proposed, where each worker only communicates with its neighbors (with mutual trust) by exchanging their local models and there is no centralized PS, as shown in Fig. 1(b). While such distributed P2P learning topology increases the robustness of FL, it might suffer from larger communication costs or slower convergence. Recently, HFL [15]–[17] has been proposed by introducing several middle-layer PSs (or called group leaders) in a hierarchical topology such that each of them only aggregates a group model from workers inside its group and sends the group model to the PS for global aggregation, as shown in Fig. 1(c). HFL can effectively hide local updates submitted by individual workers within a group, thereby enhancing privacy protection from malicious or honest-but-curious PS [15]. HFL can also provide better scalability with larger workers but may increase the total latency due to multiple exchanges

Qutaum-Assistant Federated Learning Scheduling

Quantum Assisted Scheduling Algorithm for Federated Learning in Distributed Networks

Xinliang Wei*, Lei Fan¹, Yuanxiong Guo², Yanming Gong³, Zhu Han⁴, Yu Wang*

*Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA

¹Department of Engineering Technology, University of Houston, Houston, TX, USA

²Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, USA

³Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX, USA

⁴Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA

{xinliang.wei, wangyu}@temple.edu, lfan8@central.uh.edu, {yuanxiong.guo, yanmin.gong}@utsa.edu, zhan2@uh.edu.

Abstract—The scheduling problem for federated learning (FL) with multiple models in a distributed network is challenging, as it involves NP-hard mixed-integer nonlinear programming. Moreover, it requires optimal participant selection and learning rate determination among multiple FL models to avoid high training costs and resource competition. To overcome those challenges, in literature the Benders’ decomposition algorithm (BD) can deal with mixed integer problems, however, it still suffers from limited scalability. To address this issue, in this paper, we present the Hybrid Quantum-Classical Benders’ Decomposition (HQCBD) algorithm, which combines the power of quantum and classical computing to solve the joint participant selection and learning scheduling problem in multi-model FL. HQCBD decomposes the optimization problem into a master problem with binary variables and small subproblems with continuous variables. This collaboration maximizes the potential of both quantum and classical computing, and optimizes the complex joint optimization problem. Simulation on the commercial D-Wave quantum annealing machine demonstrates the effectiveness and robustness of the proposed method, with up to 18% improvement of iterations and 81% improvement of computation time over BD algorithm on classical CPUs even at small scales.

Index Terms—Federated learning, participant selection, learning scheduling, hybrid quantum-classical optimization

1. INTRODUCTION

With the use of quantum superposition and entanglement, quantum computing (QC) has demonstrated a quantum advantage over classical computing in random quantum circuit sampling [1], Gaussian boson sampling [2], and combinatorial optimization [3]–[5]. In this paper, by leveraging the parallel computing capability of quantum computing, we focus on designing a new quantum-assisted scheduling algorithm to solve a complex joint participant selection and learning scheduling problem for federated learning (FL) in distributed networks.

Federated learning is emerging as an effective and privacy-preserving machine learning (ML) paradigm [6]–[9], which

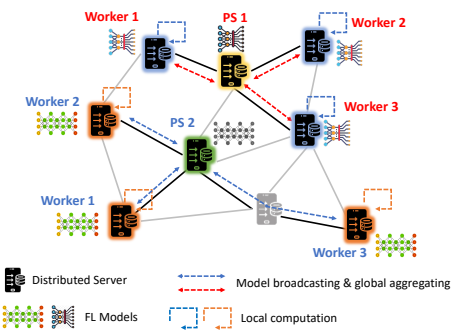


Fig. 1: The training process of multi-model federated learning.

the computing capability and network resources of servers and their data distribution are heterogeneous. Some low-performance servers may decelerate the convergence process and diminish the training performance. Also, the dispersed computing resources and large network latency may lead to high training costs. Second, for the practical scenario, training multiple different models in the shared distributed network simultaneously leads to competition for computing and communication resources. As shown in Fig. 1, two FL models are trained concurrently and each FL model requires one PS and three workers for model training. In this case, which FL model is preferentially served at which server directly affects the total training cost of all FL models. To this end, appropriate participant selection and learning schedules are fairly crucial for multi-model FL training.

Therefore, we mainly concentrate on the joint participant selection and learning scheduling problem in multi-model FL training scenarios. It should be emphasized that each server in



Hybrid Quantum-Classical FL Optimization

Motivation

- Quantum Computing (QC)
 - has brought new computing capability
 - has been used as a powerful tool for optimization^[16,17], especially, Quantum Annealing (QA)
- Applying QC to our FL scheduling problem
 - aim to solve the MINLP faster
 - propose a Hybrid Quantum-Classical Bender's Decomposition (HQCBD)

[16] Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems, Comp. & Chem. Eng., 2020.

[17] Hybrid quantum benders' decomposition for mixed-integer linear programming, IEEE WCNC, 2022

Joint participant selection and learning scheduling

$$\min_{x,y,\rho} \sum_{j=1}^W (C_j^{trans} + C_j^{local} + C_j^{global} + C_j^{rent}) \quad (5)$$

$$\text{s.t. } x_{i,j} \mu_j \kappa_j \leq c_i, \quad x_{i,j} \chi_j \leq f_i, \quad \forall i, j, \quad (6)$$

$$y_{i,j} \mu_j \leq c_i, \quad y_{i,j} \chi_j \leq f_i, \quad \forall i, j, \quad (7)$$

$$\sum_{i=1}^N x_{i,j} = 1, \quad \sum_{i=1}^N y_{i,j} = \kappa_j, \quad \forall j, \quad (8)$$

$$\sum_{j=1}^W (x_{i,j} + y_{i,j}) \leq 1, \quad \forall i, \quad (9)$$

$$i \in (1, \dots, N), j \in (1, \dots, W), \quad (10)$$

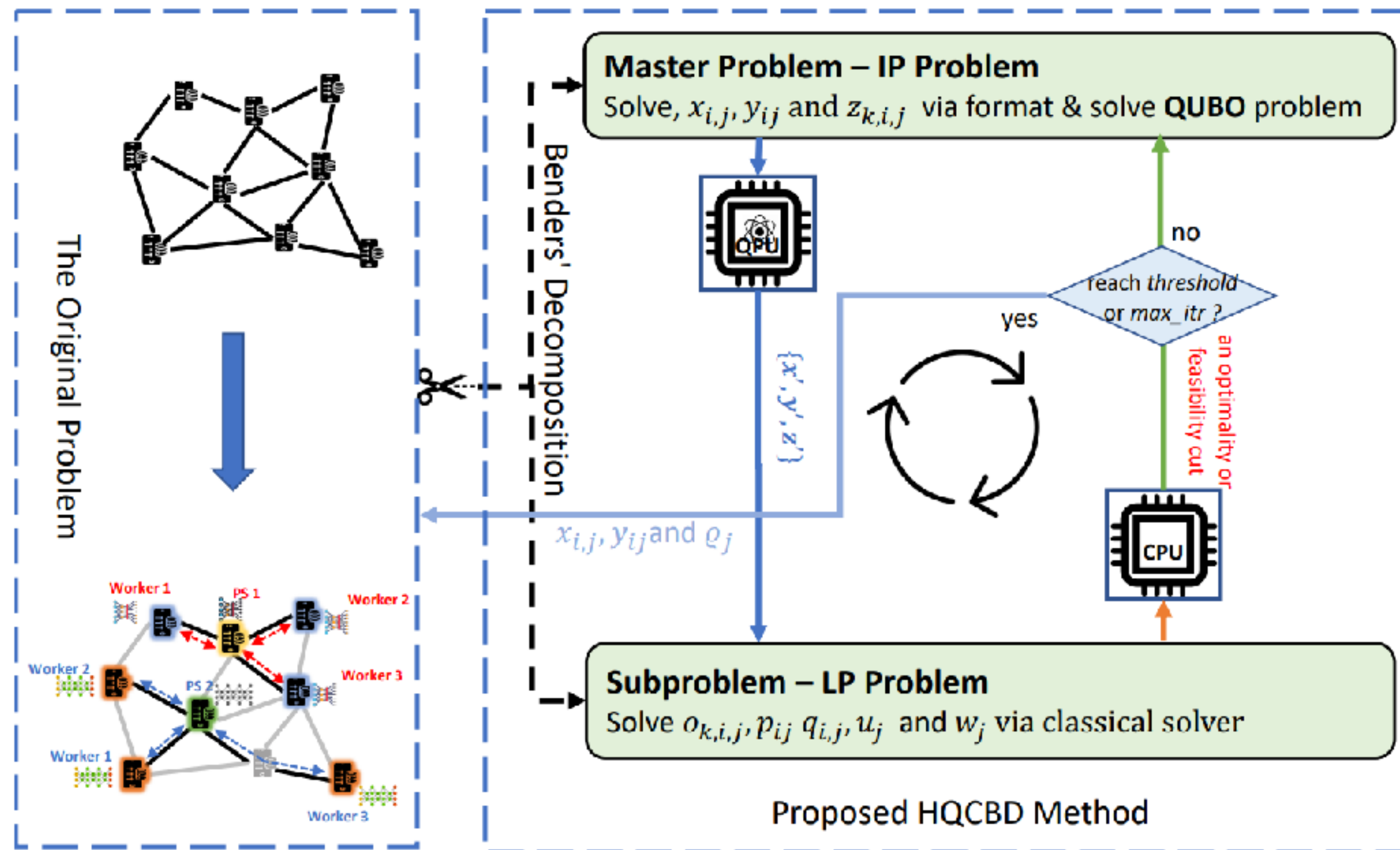
$$x_{i,j}, y_{i,j} \in \{0, 1\}, \rho_j \in [0.01, 0.99]. \quad (11)$$



$$C_j^{rent} = \sum_{i=1}^N (x_{i,j} + y_{i,j}) \cdot p_j \cdot f_i.$$

Hybrid Quantum-Classical FL Optimization

Methodology



QUBO: Quadratic Unconstrained Binary Optimization

Algorithm 1 Hybrid Quantum-Classical Benders' Decomposition (HQCBD) Method

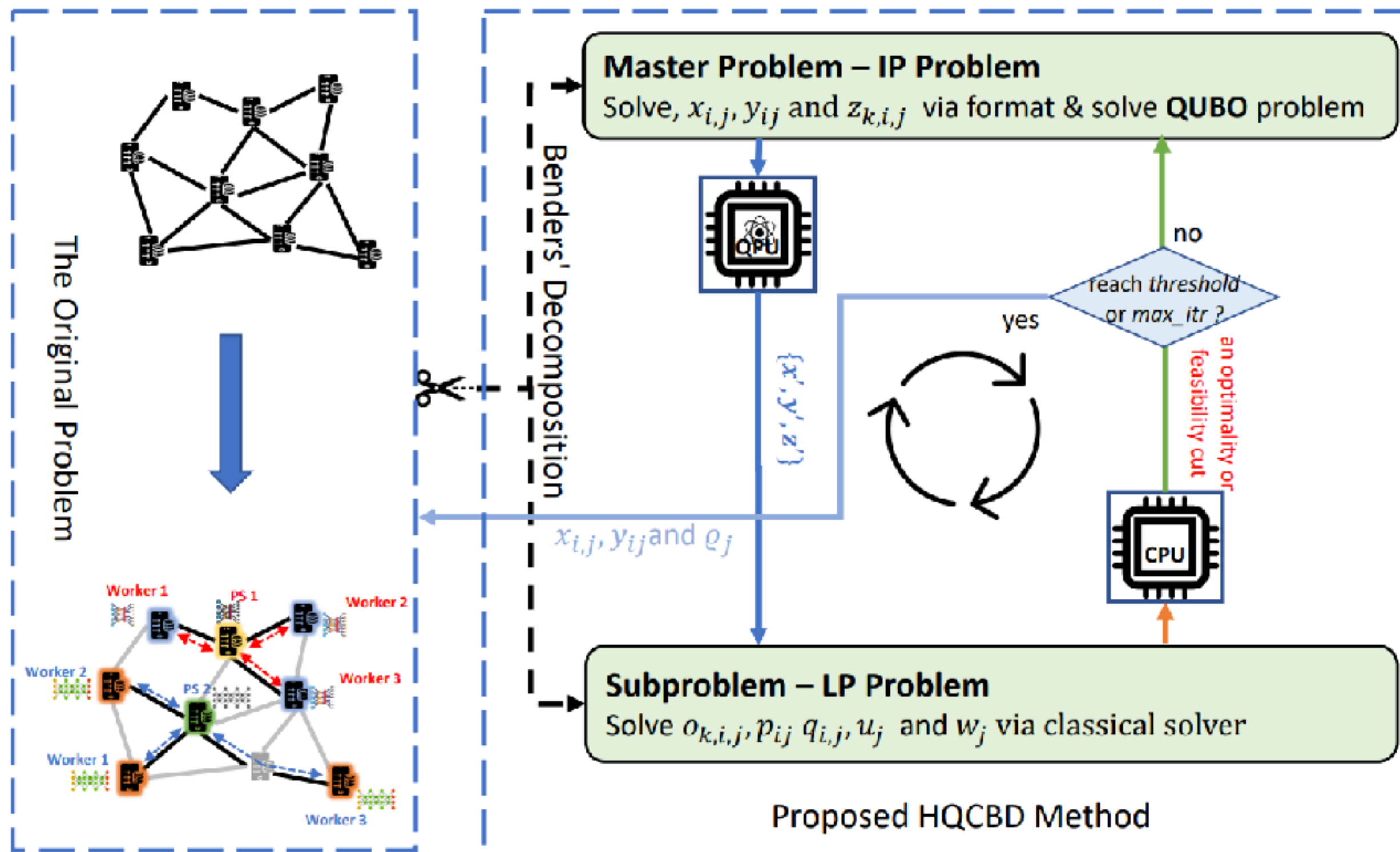
Input: Distributed network with N servers V , W FL models M , Coefficient of the objective function and constraints in master problem and subproblem

Output: PS selection x' , worker selection y' , and local convergence rate ϱ'

- 1: Initialize upper/lower bound of λ , $\bar{\lambda} = +\infty$, $\underline{\lambda} = -\infty$
- 2: Initialize threshold $\epsilon = 0.001$, $max_itr = 100$, $itr = 1$
- 3: **while** $|\bar{\lambda} - \underline{\lambda}| > \epsilon$ and $itr < max_itr$ **do**
- 4: $\mathbf{P} \leftarrow$ Appropriate penalty numbers or arrays
- 5: $\mathbf{Q} \leftarrow$ Reformulate both objective and constraints in (32) and construct QUBO formulation as (41)
- 6: $x', y', z' \leftarrow$ Solve problem (41) by quantum computer
- 7: $\lambda \leftarrow$ Extract w and replace λ with $\hat{\lambda}(w)$ as (40)
- 8: $SUP(x, y, z) \leftarrow$ Solve problem (29) with fixed x', y', z'
- 9: Extract ϱ' from $SUP(x, y, z)$
- 10: $\bar{\lambda} \leftarrow SUP(x, y, z)$
- 11: Add a new benders' cut to the master problem as (38)
- 12: $itr++ = 1$
- 13: **end while**
- 14: **return** x', y', ϱ'

Hybrid Quantum-Classical FL Optimization

Methodology



- Challenges in Quantum based solution

- How to **decompose** this original problem?
- How to **convert the problem into QUBO form** as an input to the D-wave Quantum Annealing computer?
- How to design a novel hybrid quantum-classical strategy that solves the corresponding problem **in fewer iteration?**

QUBO: Quadratic Unconstrained Binary Optimization

Hybrid Quantum-Classical FL Optimization

Benders' Decomposition

1. Reformulate the original problem

$$\begin{aligned}
 \min_{x,y,\rho} \quad & \sum_{j=1}^W \left[\frac{1}{1-\rho_j} \sum_{k=1}^N \sum_{i=1}^N a_{1,i,j,k} \cdot x_{k,j} \cdot y_{i,j} \right. \\
 & + \frac{1}{1-\rho_j} \cdot \log_2\left(\frac{1}{\rho_j}\right) \cdot \sum_{i=1}^N a_{2,i,j} \cdot y_{i,j} \\
 & + \frac{1}{1-\rho_j} \cdot \sum_{i=1}^N a_{3,i,j} \cdot x_{i,j} \\
 & \left. + \sum_{i=1}^N a_{4,i} \cdot (x_{i,j} + y_{i,j}) \right] \\
 \text{s.t.} \quad & (6) - (11),
 \end{aligned} \tag{12}$$

2. Linearize the objective function and constraints

$$\begin{aligned}
 \min_{x,y,z,u,w,o,p,q} \quad & \sum_{j=1}^W \left[\sum_{k=1}^N \sum_{i=1}^N a_{1,i,j,k} \cdot o_{k,i,j} + \sum_{i=1}^N a_{2,i,j} \cdot p_{i,j} \right. \\
 & \left. + \sum_{i=1}^N a_{3,i,j} \cdot q_{i,j} + \sum_{i=1}^N a_{4,i} \cdot (x_{i,j} + y_{i,j}) \right] \tag{17} \\
 \text{s.t.} \quad & (6) - (11), (14) - (16), \\
 & b_1 z_{k,i,j} \leq o_{k,i,j} \leq b_2 z_{k,i,j}, \tag{18} \\
 & u_j - o_{k,i,j} \leq b_2(1 - z_{k,i,j}), \tag{19} \\
 & u_j - o_{k,i,j} \geq b_1(1 - z_{k,i,j}), \tag{20} \\
 & b_3 y_{i,j} \leq p_{i,j} \leq b_4 y_{i,j}, \tag{21} \\
 & w_j - p_{i,j} \leq b_4(1 - y_{i,j}), \tag{22} \\
 & w_j - p_{i,j} \geq b_3(1 - y_{i,j}), \tag{23} \\
 & b_1 x_{i,j} \leq q_{i,j} \leq b_2 x_{i,j}, \tag{24} \\
 & u_j - q_{i,j} \leq b_4(1 - x_{i,j}), \tag{25} \\
 & u_j - q_{i,j} \geq b_3(1 - x_{i,j}). \tag{26}
 \end{aligned}$$

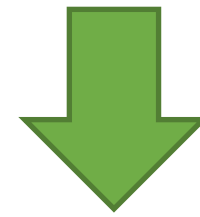
1. The **binary variables** will be solved in the **master problem**.
2. The **continuous variables** will be solved in the **subproblem**.

Hybrid Quantum-Classical FL Optimization

Subproblem

$$\min_{u,w,o,p,q} \sum_{j=1}^W \left(\sum_{k=1}^N a_{1,i,j,k} \cdot o_{k,i,j} + a_{2,i,j} \cdot p_{i,j} + a_{3,i,j} \cdot q_{i,j} \right) \quad (27)$$

$$\text{s.t. } (16), (18) - (26). \quad (28)$$



$$SUP(x, y, z) = \sum_{i=1}^N f_i(x, y, z)$$

The general form $f_i(x, y, z) = \min \mathbf{d}_i^T \mathcal{Y}_i$
 s.t. $\mathcal{A} \mathcal{Y}_i \geq \mathcal{B} \mathcal{X}_i + \mathcal{C},$

The dual problem $\max (\mathcal{B} \mathcal{X}_i + \mathcal{C})^T \pi_i \quad (29)$

$$\text{s.t. } \mathcal{A}^T \pi_i \leq \mathbf{d}_i, \quad (30)$$

$$\pi_i \geq 0, \quad (31)$$

This problem can be solved by a **classical LP solver** (e.g. Scipy, Gurobi).

Master Problem

The overall master problem (MILP)

$$\min_{x,y,z} \sum_{j=1}^W \left[\sum_{i=1}^N a_{4,i} \cdot (x_{i,j} + y_{i,j}) + \lambda \right] \quad (32)$$

$$\text{s.t. } (6) - (10), \quad x_{i,j} \in \{0, 1\}, y_{i,j} \in \{0, 1\}, \quad \forall i, j, \quad (33)$$

$$z_{k,i,j} \leq y_{i,j}, \quad \forall i, j, k, \quad (34)$$

$$z_{k,i,j} \leq x_{k,j}, \quad \forall i, j, k, \quad (35)$$

$$z_{k,i,j} \geq x_{k,j} + y_{i,j} - 1, \quad \forall i, j, k, \quad (36)$$

$$\lambda \geq \lambda^{\text{down}}, \quad (37)$$

$$\lambda \geq (\mathcal{B} \mathcal{X} + \mathcal{C})^T \pi^l, \quad (38)$$

Benders' cuts

Hybrid Quantum-Classical FL Optimization

QUBO Formulation

Master problem need to be reformulated as a pure ILP

1. Find the **best penalty coefficients** of the constraints.

$$(6) \Rightarrow \xi_1 : P_{i,j}^1 (x_{i,j} \mu_j \kappa_j - c_i + \sum_{l=0}^{\bar{l}^1} 2^l s_l^1)^2,$$

where $\bar{l}^1 = \lceil \log_2 [c_i - \min_x (x_{i,j} \mu_j \kappa_j)] \rceil$.

$$(6) \Rightarrow \xi_2 : P_{i,j}^2 (x_{i,j} \chi_j - f_i + \sum_{l=0}^{\bar{l}^2} 2^l s_l^2)^2,$$

where $\bar{l}^2 = \lceil \log_2 [f_i - \min_x (x_{i,j} \chi_j)] \rceil$.

...

$$(37) \Rightarrow \xi_{11} : P^{11} (\lambda^{down} - \lambda + \sum_{l=0}^{\bar{l}^{11}} 2^l s_l^{11})^2,$$

where $\bar{l}^{11} = \lceil \log_2 (\lambda - \lambda^{down}) \rceil$.

$$(38) \Rightarrow \xi_{12} : P^{12} ((\mathcal{B}\mathcal{X} + \mathcal{C})^T \pi^l - \lambda + \sum_{l=0}^{\bar{l}^{12}} 2^l s_l^{12})^2,$$

where $\bar{l}^{12} = \lceil \log_2 [\lambda - \min_{x,y,z,\pi} ((\mathcal{B}\mathcal{X} + \mathcal{C})^T \pi^l)] \rceil$.

2. Use a **binary vector** w to replace the continuous variable λ

$$\lambda = \sum_{ii=-\underline{m}}^{\bar{m}_+} 2^{ii} w_{ii+\underline{m}} - \sum_{jj=0}^{\bar{m}_-} 2^{jj} w_{jj+1+\underline{m}+\bar{m}_+} = \hat{\lambda}(w) \quad (40)$$

\bar{m}_+ : # of bits of positive integer part \mathbb{Z}_+ .

\underline{m} : # of bits of the positive decimal part.

$\bar{m}_- + 1$: # of bits of negative integer part \mathbb{Z}_- .

3. The final QUBO formulation of the master problem

$$\min_{x,y,z,w} \sum_{j=1}^W \left[\sum_{i=1}^N a_{4,i} \cdot (x_{i,j} + y_{i,j}) + \hat{\lambda}(w) + \xi_1 + \xi_2 + \xi_3 + \xi_4 + \xi_5 + \xi_6 + \xi_7 + \xi_8 + \xi_9 + \xi_{10} + \xi_{11} + \xi_{12} \right] \quad (41)$$

QUBO (Quadratic Unconstrained Binary Optimization) can be applied now.

Hybrid Quantum-Classical FL Optimization

Methodology

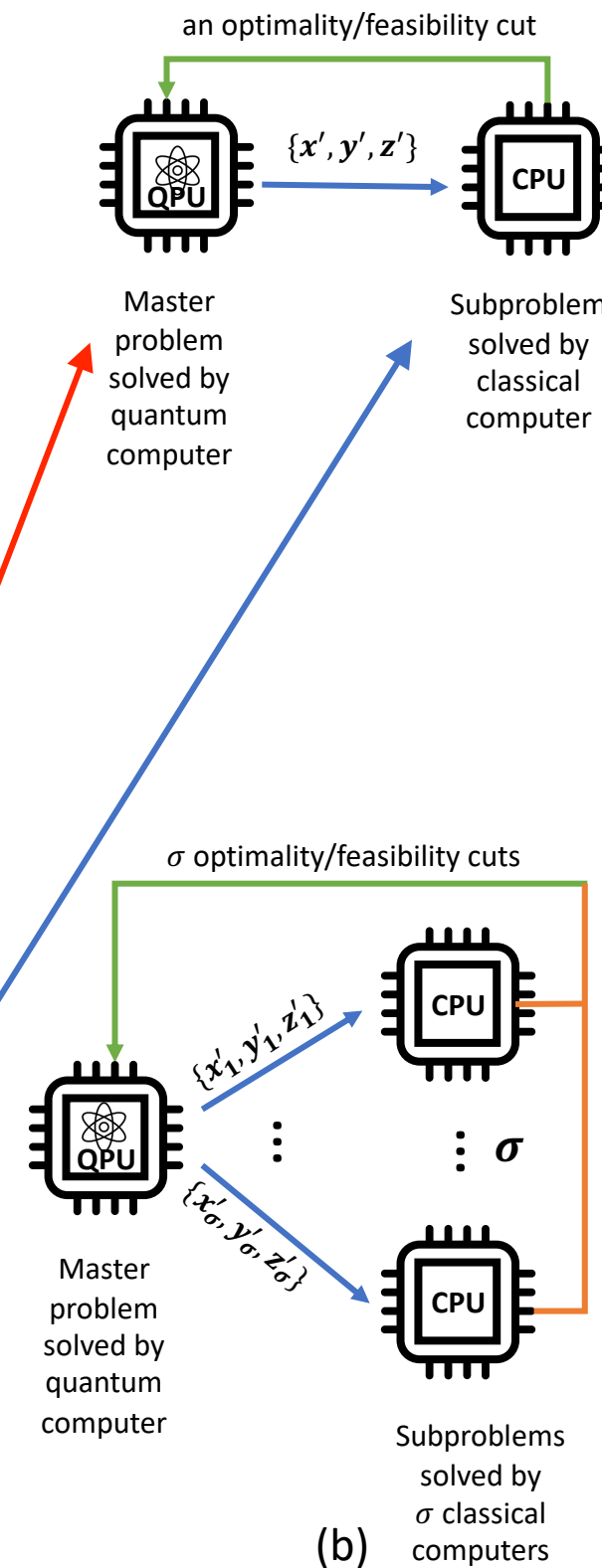
Algorithm 1 Hybrid Quantum-Classical Benders' Decomposition (HQCBD) Method

Input: Distributed network with N servers V , W FL models M , Coefficient of the objective function and constraints in master problem and subproblem

Output: PS selection x' , worker selection y' , and local convergence rate ϱ'

- 1: Initialize upper/lower bound of λ , $\bar{\lambda} = +\infty$, $\underline{\lambda} = -\infty$
- 2: Initialize threshold $\epsilon = 0.001$, $max_itr = 100$, $itr = 1$
- 3: **while** $|\bar{\lambda} - \underline{\lambda}| > \epsilon$ and $itr < max_itr$ **do**
- 4: $\mathbf{P} \leftarrow$ Appropriate penalty numbers or arrays
- 5: $\mathbf{Q} \leftarrow$ Reformulate both objective and constraints in (32) and construct QUBO formulation as (41)
- 6: $x', y', z' \leftarrow$ Solve problem (41) by quantum computer
- 7: $\underline{\lambda} \leftarrow$ Extract \mathbf{w} and replace $\underline{\lambda}$ with $\hat{\lambda}(\mathbf{w})$ as (40)
- 8: $SUP(x, y, z) \leftarrow$ Solve problem (29) with fixed x', y', z'
- 9: Extract ϱ' from $SUP(x, y, z)$
- 10: $\bar{\lambda} \leftarrow SUP(x, y, z)$
- 11: Add a new benders' cut to the master problem as (38)
- 12: $itr++ = 1$
- 13: **end while**
- 14: **return** x', y', ϱ'

Initialize parameters



Algorithm 2 Multiple-cuts Benders' Decomposition (MBD)

Input: Distributed network with N servers V , W FL models M , Coefficient of the objective function and constraints in master problem and subproblem, **number of cuts σ**

Output: PS selection x' , worker selection y' , and local convergence rate ϱ'

- 1: Initialize upper/lower bound of λ , $\bar{\lambda} = +\infty$, $\underline{\lambda} = -\infty$
- 2: Initialize threshold $\epsilon = 0.001$, $max_itr = 100$, $itr = 1$
- 3: **while** $|\bar{\lambda} - \underline{\lambda}| > \epsilon$ and $itr < max_itr$ **do**
- 4: $\mathbf{P} \leftarrow$ Appropriate penalty numbers or arrays
- 5: $\mathbf{Q} \leftarrow$ Reformulate both objective and constraints in (32) and construct QUBO formulation as (41)
- 6: $\{x', y'\}_\sigma \leftarrow$ Solve problem (41) by quantum computer and return σ feasible solutions
- 7: $\underline{\lambda} \leftarrow$ Extract \mathbf{w} with highest value and replace $\underline{\lambda}$ with $\hat{\lambda}(\mathbf{w})$ as (40)
- 8: $\{SUP(x, y)\}_\sigma \leftarrow$ Solve σ subproblems (29) with fixed x', y' in parallel
- 9: Extract ϱ' from $\{SUP(x, y)\}_\sigma$ with lowest value
- 10: $\bar{\lambda} \leftarrow \{SUP(x, y)\}_\sigma$ with lowest value
- 11: Add all σ benders' cut to the master problem as (38)
- 12: $itr++ = 1$
- 13: **end while**
- 14: **return** x', y', ϱ'

Hybrid Quantum-Classical FL Optimization

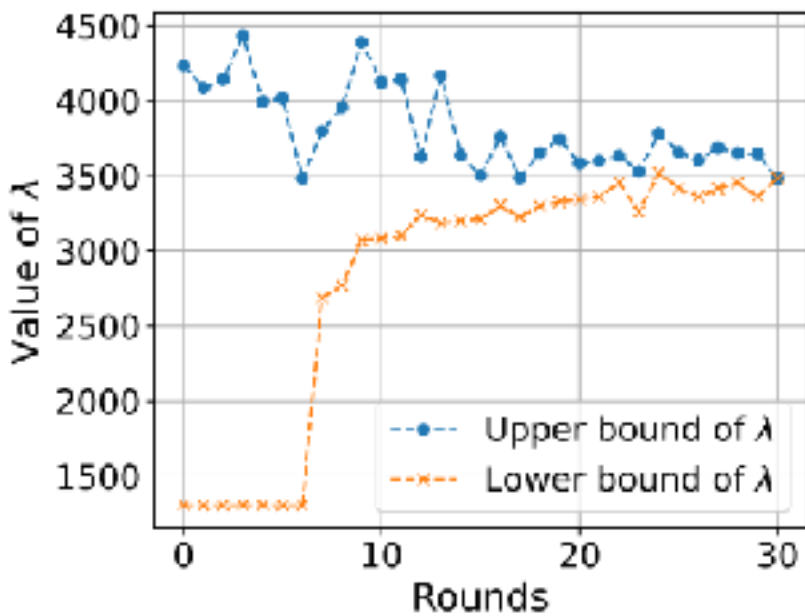
Evaluation



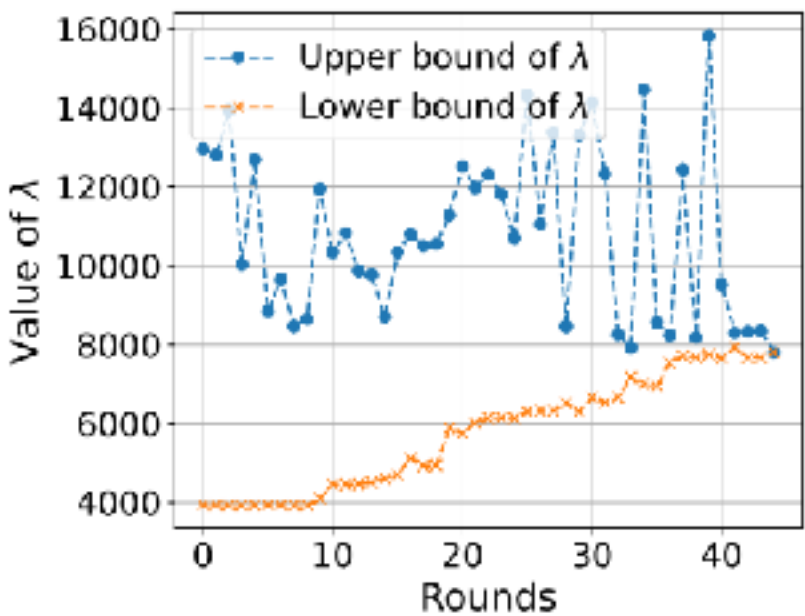
TABLE 1: Iteration of CBD and HQCBD over three different cases. Here, the set up column shows {# of servers, # of models, # of workers per model} used in each case.

Case #	Set up	# of Variables	Itr. of CBD	Itr. of HQCBD
1	{7, 1, 3}	63	32	31
2	{7, 2, 2}	126	55	45
3	{9, 2, 3}	198	91	89

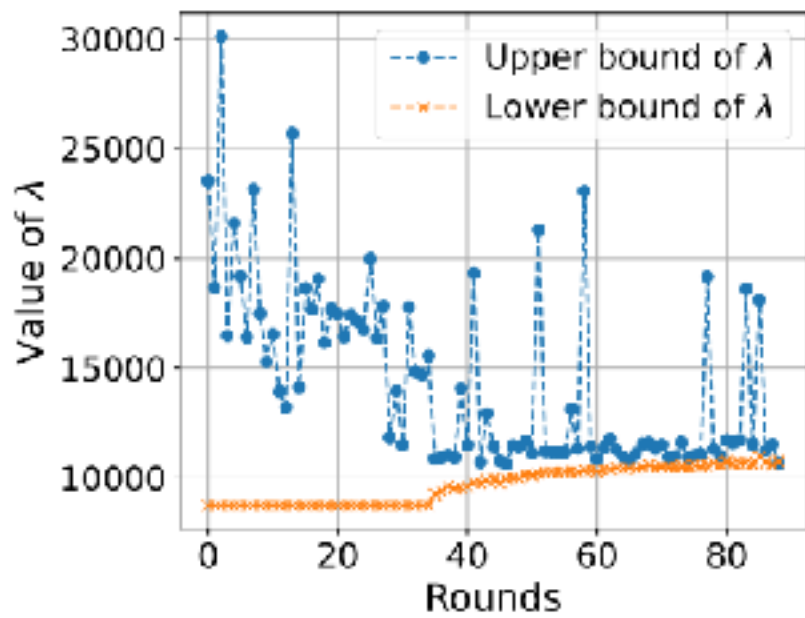
HQCBD takes few iterations to converge.



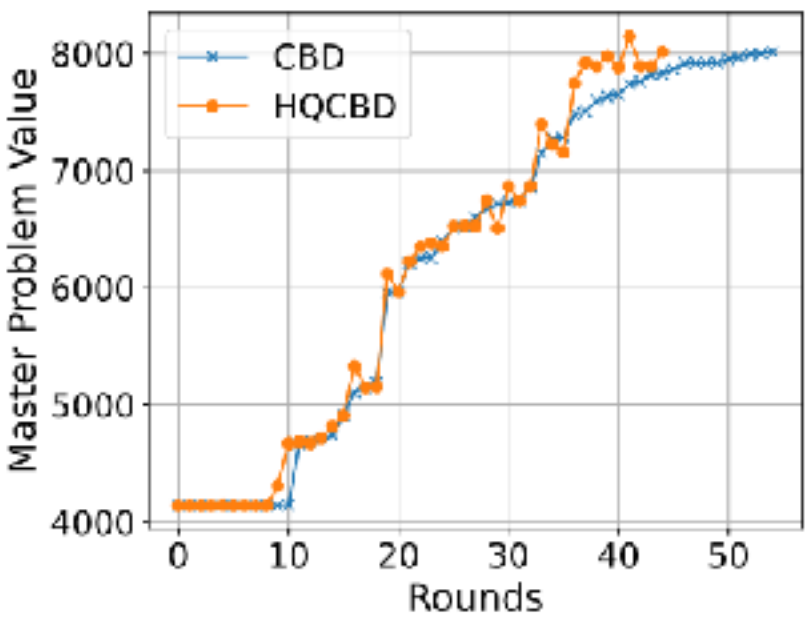
(a) Case 1



(b) Case 2



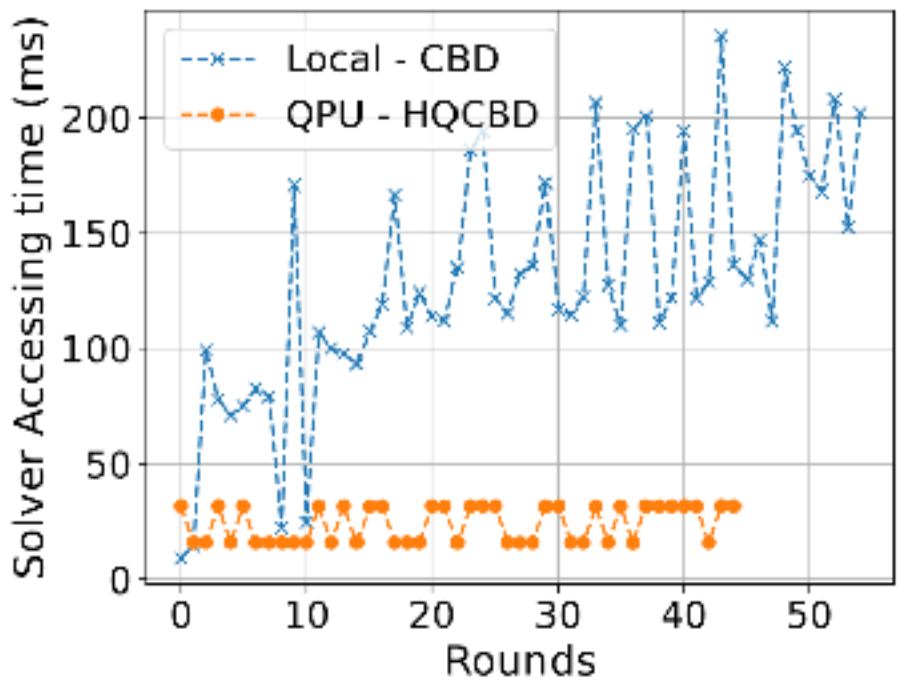
(c) Case 3



(d) Master problem value

TABLE 2: Solver accessing time (ms) comparison of CBD and HQCBD.

Case #	CBD				HQCBD			
	Max.	Min.	Mean	Std.	Max.	Min.	Mean	Std.
1	190.47	6.706	117.14	50.12	32.104	15.932	31.486	2.794
2	235.29	9.105	129.56	50.04	32.105	15.922	24.181	7.984
3	395.48	14.45	120.25	63.19	32.107	16.003	25.528	7.853



QPU take less time to solve the problem compared with CPU.

CBD: classical Benders' decomposition

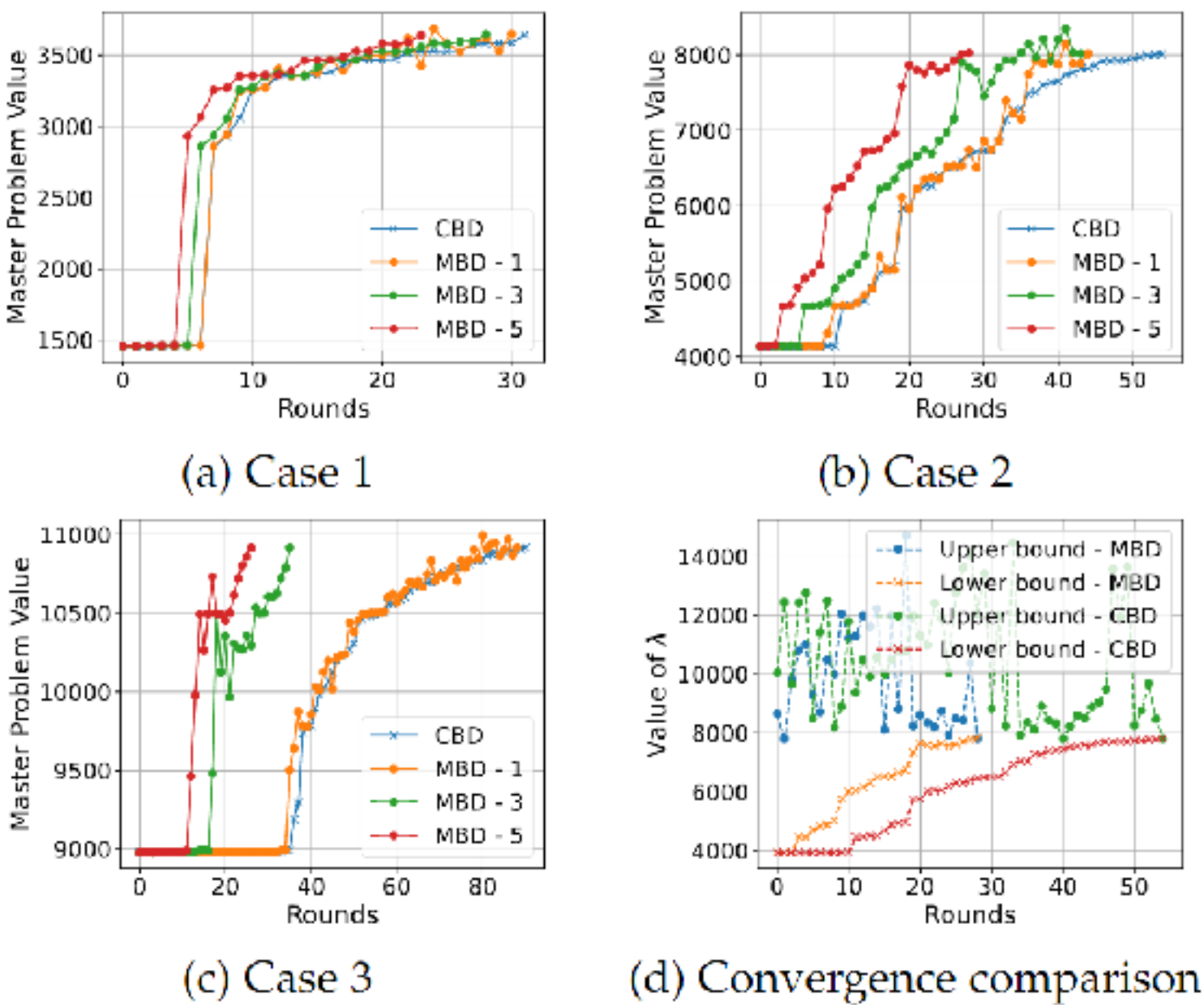
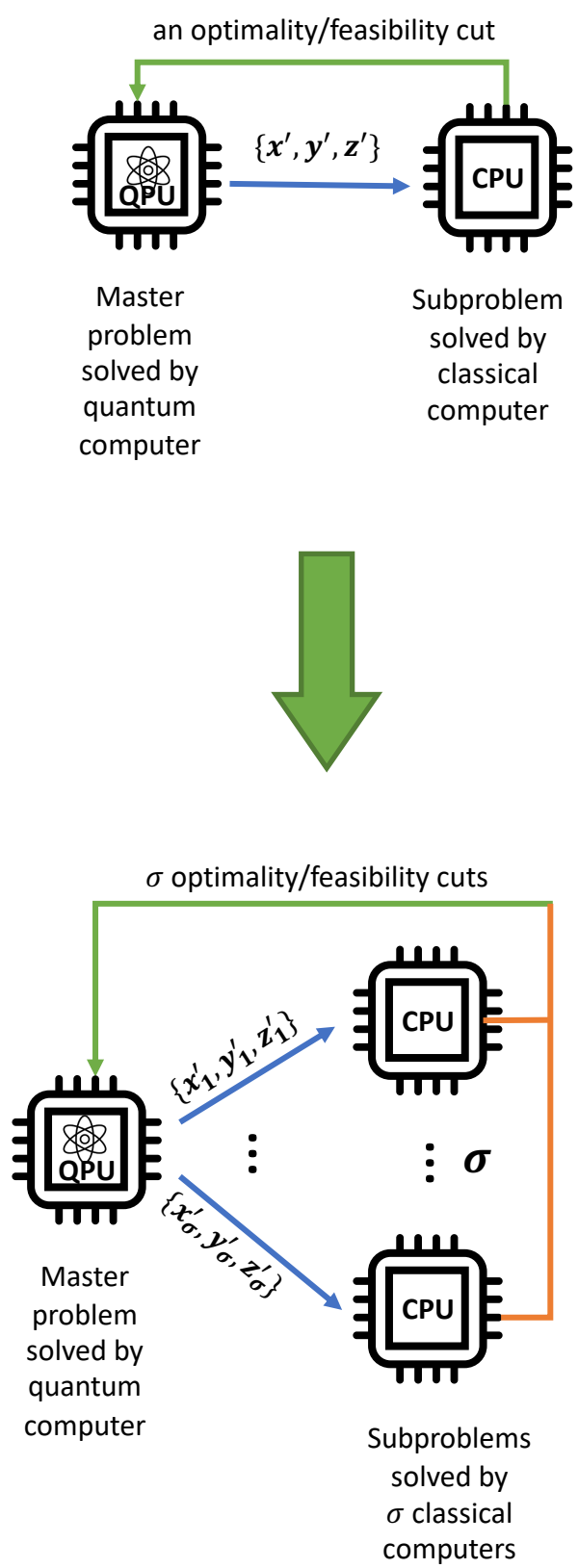
Hybrid Quantum-Classical FL Optimization

Evaluation

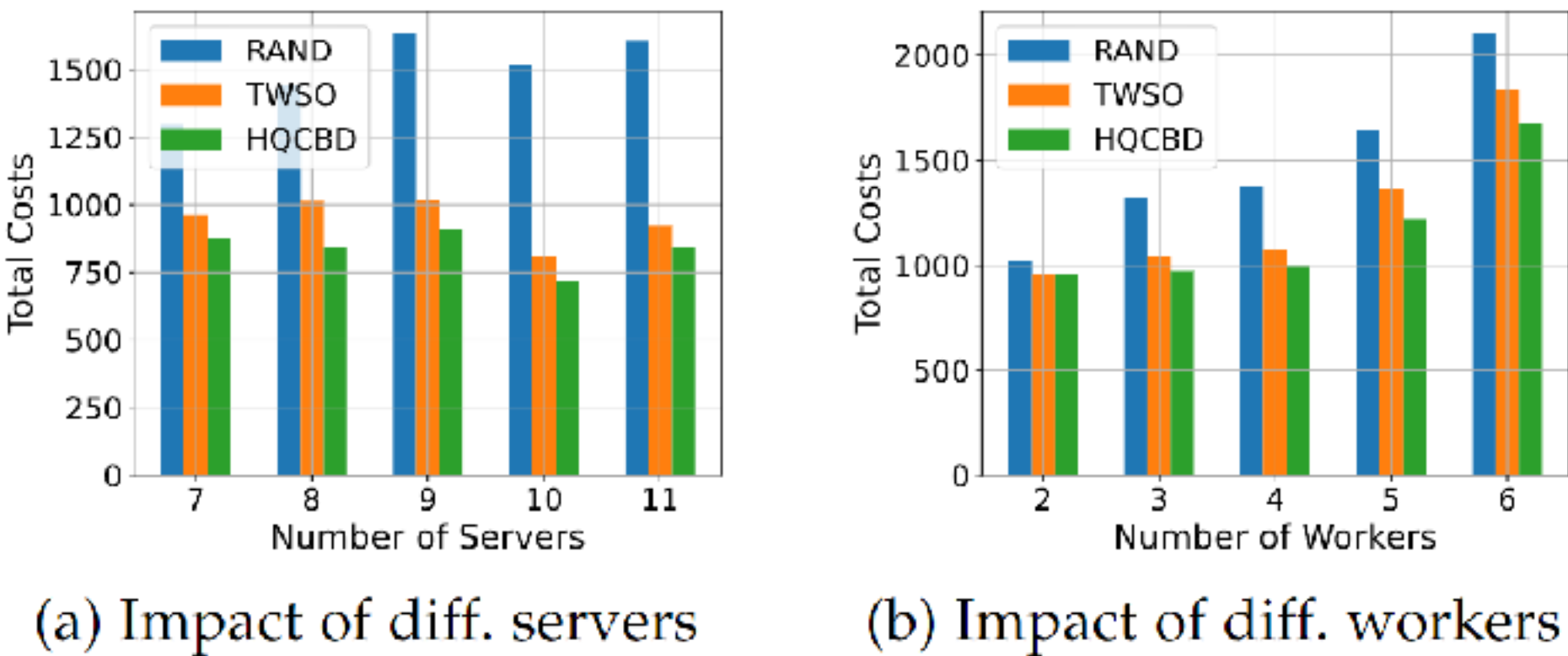
TABLE 3: Iteration comparison of CBD and MBD with different cut numbers.

Case #	Set up	# of Binary Var.	Itr. of CBD	Itr. of MBD ($\sigma = 1$)	Itr. of MBD ($\sigma = 3$)	Itr. of MBD ($\sigma = 5$)
1	{7, 1, 3}	63	32	31	29	24
2	{7, 2, 2}	126	55	45	44	29
3	{9, 2, 3}	198	91	89	36	27

MBD takes further few iterations to converge.



Performance comparison with existing methods

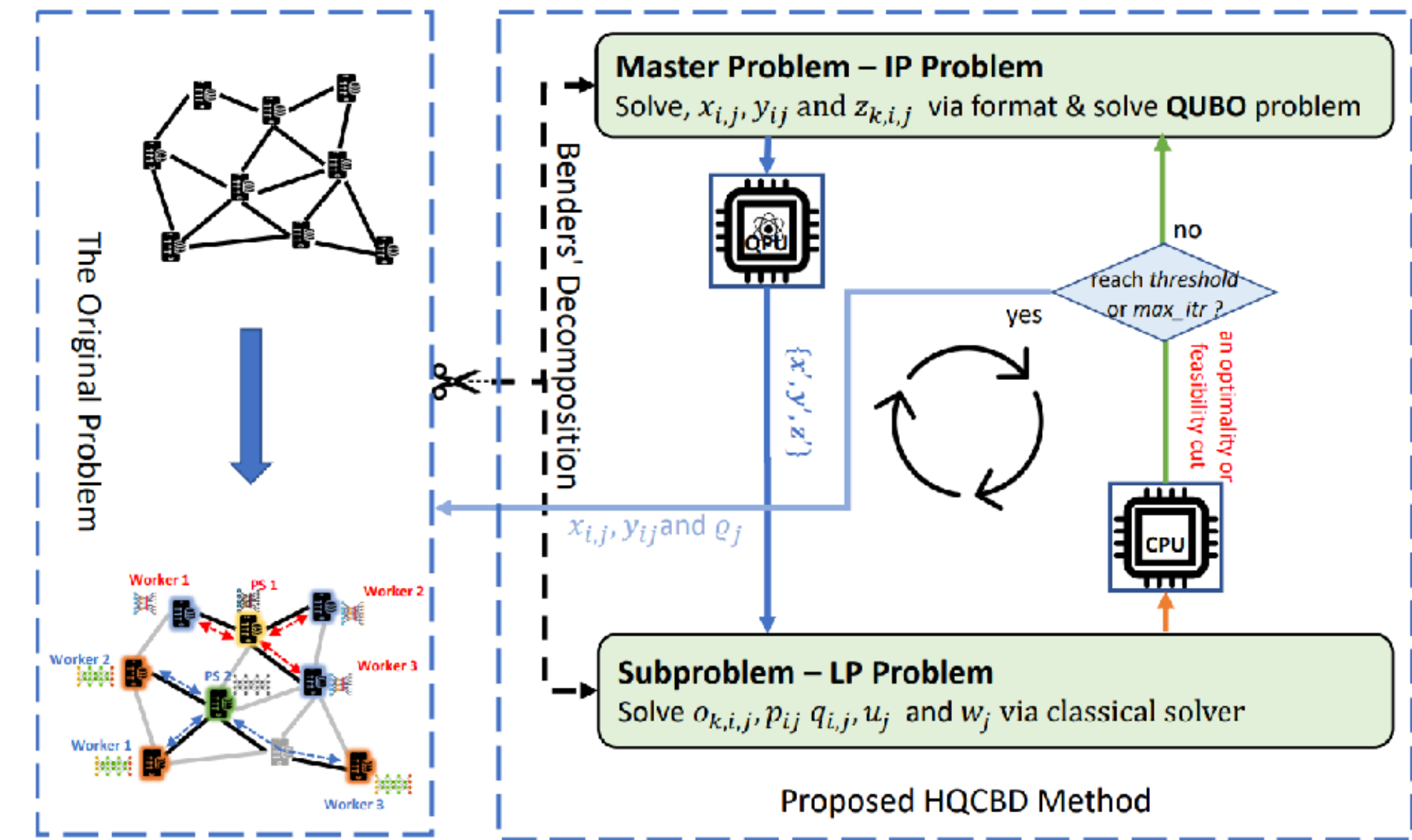


HQCBD gets further improvements compared with TWSO.

Hybrid Quantum-Classical FL Optimization

Summary

- Study joint participant selection and learning scheduling of multi-model FEL in the edge cloud:
 - select **participants** (both PS and workers) and **local convergence rate** for each FL model
 - aim to minimize the **total learning cost** of all FL models
- Propose a hybrid quantum-classical method:
 - **Hybrid Quantum-Classical Bender's Decomposition (HQCBD)**, using benders' decomposition, solving master problem via quantum annealing while solving subproblem via classical solver
 - **Multiple-cuts version**, return multiple cuts from QC
- Conduct simulations to evaluate proposed methods:
 - the proposed methods can solving the problem faster even at small scales.



Recap and Take Away

- FEL combines **edge computing** and **federated learning**, and leads to both new applications and challenges
- We have studied joint optimization of **participant selection** and **learning scheduling** in FEL
 - participant (PS/worker) selection + learning rate
 - participant selection + learning topology
 - hybrid quantum and classical BD method
- There are many other topics and aspects of FEL which are not covered by our study, such as
 - data distributions, dynamic environment, privacy and security
- There are always trade-offs between quality and cost in optimization

Acknowledgement



Joint works with:

Students: Xinliang Wei, Jiyao Liu

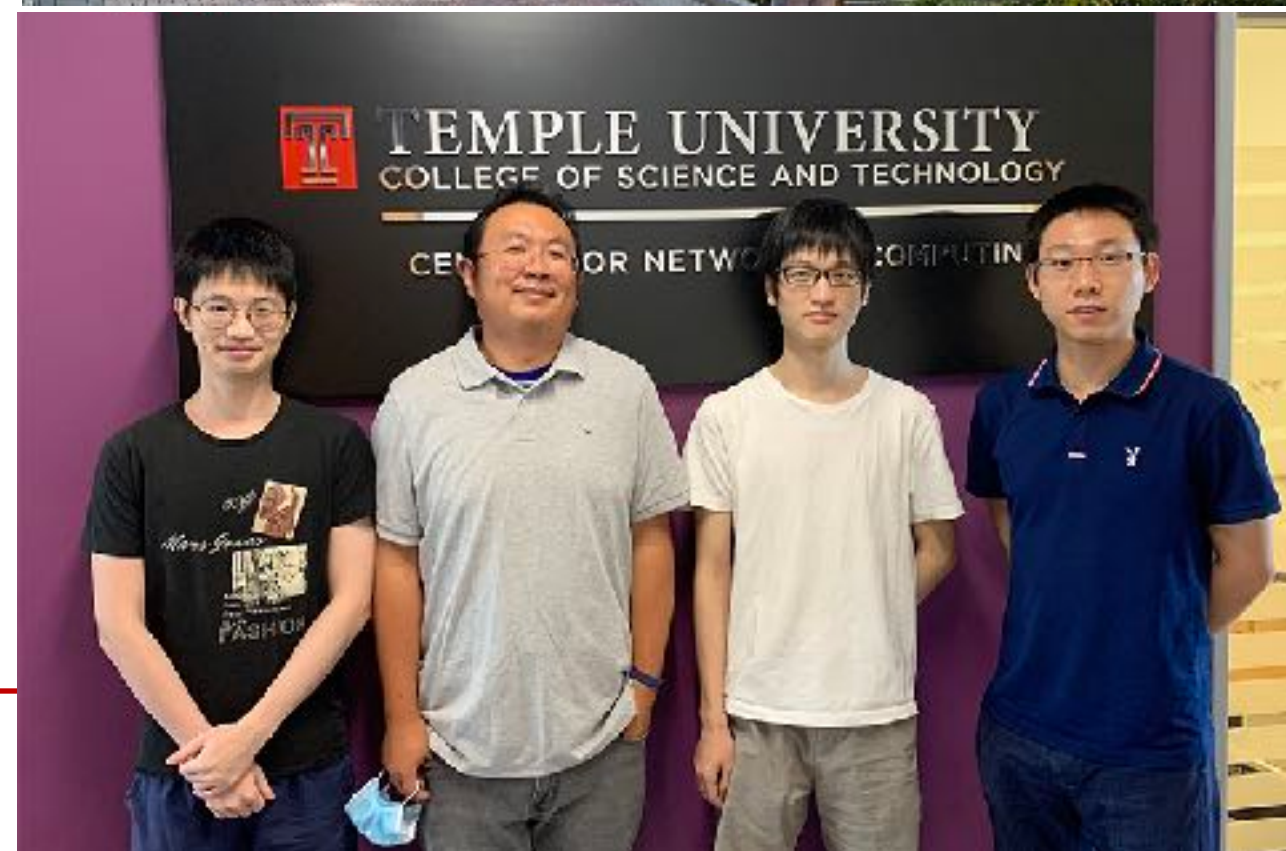
Collaborators: Xinghua Shi (Temple)

Zhu Han, Wei Fan (U Houston)

Yanming Gong, Yuanxiong Guo (UTSA)

Kejiang Ye (SIAT CAS), Cheng-Zhong Xu (U Macau)

1. X. Wei, J. Liu, Y. Wang, "Joint Participant Selection and Learning Scheduling for Multi-Model Federated Edge Learning", in 19th IEEE Int'l Conf. on Mobile Ad-Hoc and Smart Systems (MASS), 2022. Extended journal version appeared in JCST 2023.
2. X. Wei, K. Ye, X. Shi, C.-Z. Xu, Y. Wang. "Joint Participant and Learning Topology Selection in Federated Edge Learning", under review.
3. X. Wei, L. Fan, Y. Guo, Y. Gong, Z. Han, Y. Wang. "Quantum Assisted Scheduling Algorithm for Federated Learning in Distributed Networks", in 32nd Int'l Conf. on Computer Communications and Networks (ICCCN), 2023.



Questions? Comments?



Yu Wang

Temple University
Department of Computer and
Information Sciences
[https://cis.temple.edu/~yu/
wangyu@temple.edu](https://cis.temple.edu/~yu/wangyu@temple.edu)