Heterogeneous Federated Learning Frameworks for Balancing Job Completion Time and Model Accuracy

Ruobei Wang, Ruiting Zhou, *Member, IEEE*, Jieling Yu, Bo Li, *Fellow, IEEE*, and Yuqing Li, *Member, IEEE*

***Abstract*—Federated Learning (FL) is a secure distributed learning paradigm, which enables potentially a large number of devices to collaboratively train a global model based on their local dataset. FL exhibits two distinctive features in job requirement and client participation, where FL jobs may have different training criteria, and clients possess diverse device capabilities and data characteristics. In order to capture such heterogeneities, this paper proposes a new FL framework, *Hca*, which aims to strike a balance between the job completion time and model accuracy. Specifically, *Hca* builds upon a number of innovations in the following three phases: 1) *pre-estimation*: we first derive the optimal set of parameters used in training in terms of the number of training rounds, the number of iterations and the number of participating clients in each round; 2) *client selection*: we design a novel device selection algorithm, which selects the most effective clients for participation based on both client historical contributions and data effectiveness; and 3) *model aggregation*: we improve the classic FedAvg algorithm by integrating the utility in consecutive rounds as a weighted factor into aggregation computation. We further improve our model to consider a more realistic scenario, where multiple performance factors are difficult to quantify and coupled with each other, under the privacy protection requirement. We propose *HcaRL*, a deep reinforcement learning (DRL) based smart client selection and model aggregation algorithm, to solve the problem in the above scenario. To evaluate the performance and effectiveness of *Hca* and *HcaRL*, we conduct theoretical analysis and testbed experiments over an FL platform FAVOR. Extensive results show that *Hca* can improve the job completion time by up to 34% and the model accuracy by up to 9.1%, and can reduce the number of communication rounds required in FL by up to 75% compared with two state-of-the-art FL frameworks. In addition, *HcaRL* has a further performance improvement over *Hca*, including a 5% increase in model accuracy and a 23.6% reduction in job completion time.**

***Index Terms*—Federated learning, client selection, time and accuracy balancing.**

Received 23 February 2024; revised 10 February 2025; accepted 14 August 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor G. Joshi. Date of publication 8 September 2025; date of current version 30 December 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62432008 and Grant 62232004; in part by Hong Kong Research Grants Council under Grant RIF R6021-20, Grant TRS T43-513/23N-2, Grant CRF C7004-22G, Grant CRF C1029-22G, Grant CRF C6015-23G, Grant GRF 16207922, Grant GRF 16207423, Grant GRF 16203824, and Grant NSFC/RGC Joint Scheme CRS_HKUST601/24; in part by the National Key Research and Development Program of China under Grant 2024YFB2907100; in part by Shenzhen Science and Technology Program under Grant KJZD20240903100814018; in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization; and in part by the Big Data Computing Center of Southeast University. *(Corresponding author: Ruiting Zhou.)*

Ruobei Wang and Ruiting Zhou are with the School of Computer Science and Engineering, Southeast University, Nanjing 210096, China, and also with the School of Computer Science and Engineering, Wuhan University, Wuhan 430079, China (e-mail: wrb.math.cs.whu.edu.cn; ruitingzhou@seu.edu.cn).

Jieling Yu and Bo Li are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: jyucm@connect.ust.hk; bli@cs.ust.hk).

Yuqing Li is with the School of Computer Science and Engineering, Wuhan University, Wuhan 430079, China (e-mail: li.yuqing@whu.edu.cn).

Digital Object Identifier 10.1109/TON.2025.3603254

I. INTRODUCTION

FEDERATED learning (FL), a privacy-preserving distributed learning paradigm, enables potentially a large number of devices to collectively train a global machine learning (ML) model using the devices' own local dataset [1], [2], [3]. With FL, an FL server in a cloud is used to coordinate participating clients synchronously in multiple communication rounds. In each round, a small subset of these clients is selected to train an ML model using their local data. After a number of training iterations are performed, clients will send their local model updates to the FL server, which aggregates these updates using an aggregation algorithm (*e.g.,* FedAvg [4]) to update the global model parameters and send them back to clients. This process iterates until a pre-specified model accuracy or the maximum training time (*i.e.,* job completion time) is reached [5]. While the training process of FL is similar to that of conventional ML, FL executions have distinctive features in job requirements and client participation. Improving FL model accuracy is generally achieved by increasing computational loads, which consume a larger amount of energy and need more time. But some FL jobs require making decisions in a short time, such as Gboard - Google keyboard, which makes typing suggestion while typing [6]. The trade-off between time and accuracy brings new challenges and opportunities in FL training [7], [8]. To start with, we need to properly select three critical control parameters including the total number of training rounds, the number of iterations and the number of participating clients in each round [7], [9], [10]. This turns out to be challenging in that there exist intrinsic correlations among these parameters, and their collective impacts on the job completion time or model accuracy are largely unknown. On the other hand, exhaustive search for the optimal values is practically infeasible. Second, FL potentially involves a large number of clients, each with heterogeneous device capability and data characteristics. For

instance, the clients with high computing capability and more relevant datasets can contribute more to the quality of the model training. Consequently, how to select the most effective set of clients for participation during each training round becomes critically important.

Existing works in FL have focused on improving model accuracy [11] or reducing energy cost [12], usually with a random client selection algorithm. Considerable efforts have been made to improve the effectiveness of client selection [13], [14]. However, they typically only consider device performances such as local loss or bias, while ignoring a crucial factor - the quality of the data. In FL, a major challenge is that data distribution among clients is unbalanced and not independent and identically distributed (non-IID). Irrelevant or even harmful data can significantly affect the FL training performance. Proper client selection can mitigate this problem. We will further discuss this in detail in Sec. II. To comprehensively capture the heterogeneity of FL, in this paper, we propose a new FL framework, *Hca*, for optimizing the FL training performance. First, we assign weighted indices to the model loss and job completion time to capture the different job training preferences. The value of the index is defined by FL jobs. Minimizing the sum of two weighted factors can strike a balance between job completion time and model accuracy in the training process. Second, to characterize the heterogeneity in device capability and data characteristics, we define two criteria in the client selection process: *historical effectiveness* which is related to the weighted index and captures the estimated convergence rate; and *data effectiveness*, measuring the matching degree between client's dataset and the target job's dataset, as well as the similarity between two participating clients' datasets. The objective is to select the most effective clients with higher values in these two criteria.

To strike a balance between job completion time and model accuracy, we consider the FL training process in three stages: i) *pre-estimate stage*, which aims to derive the three control parameters in polynomial time; ii) *client selection stage*, which focuses on selecting the most effective clients based on historical and data effectiveness. This stage utilizes the three control parameters obtained in the first stage to enable faster and better FL training while satisfying job preference; iii) *model aggregation stage*, which improves the classic *FedAvg* aggregation algorithm to further enhance the FL model training accuracy. The combined optimizations in the three stages not only meet the basic requirements in job completion time and model accuracy, but also maximize the training performance. The problem turns out to be particularly challenging from two aspects, i) neither the job completion time nor the model loss presents an explicit expression related to the three control parameters; ii) the client section problem is a 0-1 quadratic program (QP), which is proven to be NP-hard.

In this context, we propose a new FL framework, *Hca*, to solve the three-stage problem. We first reformulate the trade-off objective in the pre-estimate stage into a function capturing the three control parameters through analysis on the upper bounds of the training time and model accuracy. The reformulated problem is strong multi-convex and can be solved by the block coordinate decent (BCD) algorithm [15]. During the client section stage, we first linearize the 0-1 QP into an integer linear program (ILP). The relaxed version of the ILP can be solved directly to obtain a fractional solution. We then design a randomized pairwise rounding algorithm to round the fractional solution to the required integer solution. Noticeably that our proposed pair-by-pair rounding technique guarantees the feasibility of the solution with bounded integral gap. Finally, in the model aggregation stage, we select most relevant local updates and integrate the model loss reduction between consecutive rounds as a weighted factor into the aggregation calculation, to further enhance the FL model accuracy.

We extend to study a realistic scenario, where the practical requirement of privacy protection and the coupling of various performance factors are considered. We exploit deep reinforcement learning (DRL) technique to jointly solve the client selection and aggregation optimization problems. In this scenario, instead of expressing the selection criteria in the model, we use DRL techniques to analyze the effects of various complex coupling factors while avoiding the risk of privacy leakage caused by data characterization. A new algorithm, *HcaRL*, is proposed to solve the extended problem. *HcaRL* can jointly optimize the client selection and model aggregation in the each round in FL training.

We conduct extensive experiments to evaluate the performance of *Hca* and *HcaRL*. The experiments train a CNN model on three common datasets over the FL platform FAVOR [16]. The highlights of the results are: i) the weighted factor α plays a trade-off between the time and accuracy requirements; ii) *Hca* reduces the job completion time by up to 34% and improves the model accuracy by up to 9.1% at the same time when compared with FAIR [17] and FedAvg [4]; iii) *Hca* reduces the number of training rounds required by up to 67% and 75%, when compared to the above two benchmarks; iv) *HcaRL* improves the model accuracy by up to 5% and reduces the job completion time by up to 23.6% compared with *Hca*, which indicates that *HcaRL* outperforms *Hca* on all counts, not to mention the other benchmarks; v) *HcaRL* can achieve better performance even on data distribution with high non-IID degree.

The rest of the paper is organized as follows. We present the related work in Sec. II. We introduce system model in Sec. III and describe the design of *Hca* in Sec. IV. Sec. V proposes a smart client selection and model aggregation algorithm, *HcaRL*, based on DRL techniques. We evaluate *Hca* and *HcaRL* in Sec. VI and conclude the paper in Sec. VII.

II. Related Work

FL Framework Optimization. Since McMahan et al. [4] proposed the first FL framework and demonstrated its effectiveness, many efforts have been devoted to improving the performance of FL. Some works [18], [19], [20] investigated the theoretical convergence guarantees in heterogeneous settings, while others were proposed to improve the structure of FL framework [21], [22], [23], [24]. Briggs et al. [25] proposed a hierarchical clustering approach to categorize clients by the similarity of their local updates. Wang et al. [26] designed a hierarchical aggregation approach by clustering clients and explored the optimal cluster structure

with resource constraints. Wang et al. [27] filtered out the irrelevant updates by ameliorating aggregation method. Leroy et al. [28] designed an Adam-based per-coordinate averaging strategy for global aggregation. Wang et al. [10] proposed a control algorithm to minimize the loss function under a given resource budget. Luo et al. [9] analyzed how to optimally select the essential control variables to minimize the total cost while ensuring convergence. Huba et al. [29] proposed an asynchronous FL system and experimentally demonstrate that asynchronous FL is significantly faster than synchronous FL when training across millions of devices. Nguyen et al. [30] proposed a new buffered asynchronous aggregation method, FedBuff, which combines the best properties of synchronous and asynchronous FL. Liu et al. [31] proposed an adaptive asynchronous strategy to minimize the impact of stragglers based on DRL. Compared with [31], the adaptive optimization of our paper covers a wider range of parameters. Therefore, our paper shows more comprehensive and efficient adaptability. Reddi et al. [32] propose an adaptive federated optimization framework that enhances convergence speed and performance in federated learning by introducing adaptive optimizers such as ADAGRAD, ADAM, and YOGI. Their work focuses on algorithmic improvements through adaptive optimization, whereas we address system heterogeneities to balance job completion time and model accuracy. Nguyen et al. [33] proposed FedDRL, which adaptively determines aggregation weights for each client using DRL. None of the existing approaches jointly considered the client selection and heterogeneous requirements of different FL jobs. Our proposal, *Hca* is different in that it aims to jointly optimize the training time and model accuracy through proper selection of training parameters and model aggregation.

FL Client Selection. Nishio and Yonetani [34] presented a client selection method based on the hardware and wireless resources. Ribero andd Vikalo [35] proposed a client selection strategy by utilizing the progression of clients' weights from an Ornstein-Uhlenbeck process. Wang et al. [16] selected clients by leveraging DRL technique to speed up convergence. Chen et al. [36] proposed a client selection scheme by minimizing the variance of the stochastic gradient. Cho et al. [37] proposed the Power-of-Choice client-selection framework to balance convergence speed and solution bias. While both address the topic of client selection, our approach focuses more on designing a client selection method that balances historical contributions and data effectiveness, specifically tailored for heterogeneous FL jobs. Deng et al. [17] constructed a quality-aware selection scheme by learning quality estimation. Such client selection strategies largely rely on the client performance such as local losses or bias, while completely ignoring the effect of clients' data properties. Another relevant work considered data property [38], and designed a method to select high-quality clients and data samples. Lai et al. [39] propose a client selection algorithms to improve the time-to-accuracy performance of FL training. Yu et al. [40] propose to dynamically adjust and optimize the trade-off between maximizing the number of selected customers and minimizing the total customer energy consumption. Luo et al. [41] present an adaptive client sampling algorithm to mitigate system and statistical heterogeneities in federated learning, aiming to reduce convergence time. Our proposed frameworks not only reduce the convergence time but also improve the model accuracy through a multi-stage approach. Li et al. [42] propose PyramidFL, a fine-grained client selection algorithm that takes full advantage of data and system heterogeneity within selected clients to more effectively analyze their utility. Sun et al. [43] propose a hardware-aware model selection algorithm that achieves good performance in training efficiency and model performance. Different from all the above works, *Hca* designs a comprehensive client selection criterion by incorporating both client's device performance and data properties with proven theoretical guarantee.

TABLE I
LIST OF NOTATIONS

K	total number of training rounds
τ	number of local iterations in each round
C	maximum completion time
ε	maximum loss requirement
α	weight of loss requirement
C_{tot}^K	total training time after K rounds
N	number of available clients before training
$F(\boldsymbol{w}^k)$	the loss of global model after k rounds' training
$F(\boldsymbol{w}_i^k)$	the loss of client i 's local model after round k
$F(\boldsymbol{w}^*)$	the minimum loss, a fixed value
$\mathbb{I}^k$	the set of available clients in round k
$\mathbb{I}'^k$	the set of initially screened clients in round k
N_k	number of clients in $\mathbb{I}'^k$
M	number of participating clients
x_i^k	whether or not select client i in round k
U_i^k	the historical effectiveness of client i in round k
$\widehat{q}_i^k$	the estimated utility of client i in round k
q_i^k	accuracy factor, the reduction of model loss
$\widehat{C}_{tot}^{k,i}$	the estimated training time of client i in round k
γ_i^k	the irrelevance of client i in round k
γ_0	the upper bound of the irrelevance
$s^k(i,j)$	the similarity between i and j 's data in round k
w_i^k	the aggregation weight of client i in round k

III. SYSTEM MODEL

A. System Overview

Heterogeneous Job Preferences. We assume that an FL job comes with its maximum completion time requirement C and accuracy demand ε, where ε is the required difference between the model loss and the minimum loss (which is determined by the property of the loss function). As discussed, *Hca*, aims to balance an FL job's completion time and model accuracy and selects the most efficient participating clients for model aggregation, while satisfying two training requirements. Considering FL jobs have different preferences on the completion time and model accuracy, we define a weight index α for a FL job, to capture the trade-off between the two values, where $\alpha \in [0,1]$. The closer α is to 1, the higher value the FL job puts on the accuracy than the completion time. α affects both the control parameter estimation process before FL training and the client selection process in each training round.

Training Process and Decision Overview. To effectively achieve the unique requirement of time and accuracy trade-off, we consider that a training process consists of three stages. *First*, upon the arrival of an FL job, *Hca* computes three control parameters based on the job preference and requirements: i) K, the total number of training rounds; ii) τ, the number of iterations in each round; and iii) M, the number of participating clients per round. In each training round, we use the control parameters derived in the pre-estimation stage to complete client selection, training, and aggregation.[1] *Second*, in each round, *Hca* evaluates the effectiveness of available clients based on their device capabilities and data characteristics, and selects the most effective clients from N available clients. Let a set of binary variables $\{x_i^k \in \{0,1\} | i \in \mathbb{I}^k, k \in K\}$ denote the selection, where $\mathbb{I}^k$ is the set of available clients in round k. If client i is selected to participate in the training process in round k, $x_i^k = 1$, otherwise $x_i^k = 0$. *Third*, *Hca* performs model aggregation. Stage 2 and stage 3 are carried out under the premise of the control parameters determined in stage 1. The first two stages will be described in details in the next two subsections. Notations are listed in Table I.

B. Pre-Estimation Stage

Problem Formulation. To achieve the trade-off between the two preferences, we introduce a weight index α and use a weighted sum approach to combine the two objectives [45]. The weight index α can be adjusted by sliding from 0 to 1, to customize the task optimization objective with specific requirement preference. The weighted sum method has been widely used to achieve multi-objective optimization in various fields of research [46], [47], [48]. The difference between the ranges of the two objectives needs to be normalized for consistency. In our implementation, we unified the ranges of the model loss and completion time values using the min-max normalization approach based on their actual value ranges obtained from pretraining. We formulate the control parameter pre-estimation problem as an optimization problem whose objective is to minimize the weighted completion time and model loss while satisfying job training requirements. We call this problem as PEP (Pre-estimation Problem). Let C_{tot}^K be the total training time after K rounds, and $F(\boldsymbol{w}^K)$ be the model loss after K rounds. $F(\boldsymbol{w}^*)$ is the minimum loss, which is a fixed value. The PEP for each incoming job can be formulated as follows:

[1]Although our derivation of the convergence upper bound in the pre-estimation stage is based on the basic *FedAvg* algorithm, recent works [19], [44] explicitly state that this derivation process can be naturally extended to other federated optimization methods. Moreover, the client selection and model aggregation algorithms proposed in this paper will not affect the correctness of the derivation process. It should be noted that random selection and average aggregation provide the lower bound of the convergence performance for client selection. In order to ensure the convergence of our proposed algorithm, we improve *FedAvg* from different aspects, such as considering data relevance in the selection phase and model relevance in the aggregation phase. These improvements are designed to mitigate the effects of deviating models and maintain convergence performance similar to or better than *FedAvg*. Therefore, it can be naturally deduced that our algorithm have a better convergence performance than *FedAvg*.

P1:

$$\min_{K,M,\tau} \quad (1-\alpha)\mathbb{E}[C_{tot}^K] + \alpha\mathbb{E}[F(\boldsymbol{w}^K)] \tag{1}$$

$$s.t.\ \mathbb{E}[C_{tot}^K] \le C, \tag{1a}$$

$$\mathbb{E}[F(\boldsymbol{w}^K)] - F(\boldsymbol{w}^*) \le \varepsilon, \tag{1b}$$

$$K, M, \tau \in \mathbb{Z}^+. \tag{1c}$$

Constraint (1a) guarantees that the expected training time is less than the time requirement. Constraint (1b) ensures that the expected loss after training K rounds is less than the loss requirement.

Challenges. Solving **P1** directly is intractable, because neither the expectation of training time $\mathbb{E}[C_{tot}^K]$ nor the model loss $\mathbb{E}[F(\boldsymbol{w}^K)]$ is an explicit expression related to the decision variables K, τ and M.

C. One-Round Client Selection Stage

Selection Criteria. For each training round k, given the collection of available clients $\mathbb{I}^k$, *Hca* selects at least M clients for this round to maximize the quality of the aggregated global model. *Hca* evaluates the effectiveness of client i for the target job from two aspects: i) *historical effectiveness*, which applies to clients that have participated in the target job before. It evaluates the effectiveness of each available client based on staleness mechanism [49] using training time and quality from previous training rounds. The weight parameter α is integrated into this evaluation function to capture the heterogeneity of job preference; and ii) *data effectiveness*, which measures the quality and the fitness of client i's dataset for the target job. Data effectiveness includes the data *relevance* (γ_i^k) between client i and the target job, as well as the data *similarity* ($s^k(i,j)$) between the two participating clients i and j.

Historical Effectiveness. In particular, the historical effectiveness of client i in round k (U_i^k) is defined as:

$$\widehat{U}_i^k = \begin{cases} \dfrac{\alpha \widehat{q}_i^k}{(1-\alpha)\widehat{C}_{tot}^{k,i}}, & \widehat{C}_{tot}^{k,i} \neq 0, \alpha \neq 1 \\ \widehat{q}_i^k & \widehat{C}_{tot}^{k,i} \neq 0, \alpha = 1 \\ -\widehat{C}_{tot}^{k,i} & \widehat{C}_{tot}^{k,i} \neq 0, \alpha = 0 \\ 0, & otherwise \end{cases} \tag{2}$$

where $\widehat{q}_i^k$ and $\widehat{C}_{tot}^{k,i}$ are the estimated utility and estimated completion time in round k, both of which are estimated from historical training information. Note that $\widehat{C}_{tot}^{k,i}$ includes client i's both computation time and communication time. $\widehat{U}_i^k$ is a concept similar to the training speed, representing the estimated model quality per unit of time. We multiply the weight factor α and $(1-\alpha)$ in front of $\widehat{q}_i^k$ and $\widehat{C}_{tot}^{k,i}$, in order to meet the time and accuracy trade-off. We use the reduction of the model loss to define the accuracy factor in round t, *i.e.,* $q_i^t = F(\boldsymbol{w}^{t-1}) - F(\boldsymbol{w}_i^t)$. It should be noted that the quality value of clients that did not participate in round t is 0. The FL framework records the q_i^t values of all participating clients in each training round. The staleness mechanism is used to obtain a more accurate estimated quality $\widehat{q}_i^k$. $\widehat{q}_i^k$ is calculated based on the weighted average of the quality of each previous round of training $\{q_i^1, q_i^2, \ldots, q_i^{k-1}\}$, where the weight of q_i^t

in round t decreases with the difference from the current round number $k-t$. Specifically, the weight of q_i^t is determined by an exponential function β^{k-t}. The constant base β is introduced, with a value range of $(0,1)$. β determines the decay rate of time, which controls the change in weight between different rounds. Smaller β values will lead to faster decay, so that the quality of earlier rounds has less impact on the current estimated quality, and vice versa. So $\widehat{q}_i^k$ is defined as:

$$\widehat{q}_i^k = \begin{cases} \frac{\sum_{t=1}^{k-1} q_i^t \beta^{k-t}}{\sum_{t=1}^{k-1} \beta^{k-t}}, & \exists k,\ s.t.\ q_i^k > 0 \\ 0, & otherwise \end{cases} \tag{3}$$

Similarly, the estimated completion time of round k $\widehat{C}_{tot}^{k,i}$ is also obtained from the historical information:

$$\widehat{C}_{tot}^{k,i} = \begin{cases} \frac{\sum_{t=1}^{k-1} C_{tot}^{t,i} \beta^{k-t}}{\sum_{t=1}^{k-1} \beta^{k-t}}, & \exists C_{tot}^{k,i} \\ 0, & otherwise \end{cases} \tag{4}$$

where $C_{tot}^{t,i}$ is the completion time of round t.

Data Effectiveness – Relevance: Besides the historical information, to avoid the negative effect of unbalanced and non-IID data distribution on participating clients, we also hope to obtain information from data as a selection criterion under the premise of protecting privacy. First, we define the relevance γ_i^k to measure the matching degree between client i's dataset and the target job's dataset. The target job's dataset refers to any dataset that can identify the target job. For example, in the digit classification task, a dataset containing ten types of digit pictures from 0 to 9 can be used as the target job's dataset. Its main purpose is to obtain all classification labels D_i^k to calculate the correlation γ_i^k. To protect the privacy of clients and the job, *Hca* adopts private set intersection (PSI) protocol [50], which is a widely used lightweights multi-party secure computing protocol. In PSI protocol, both parties involved in the calculation collaborate to compute the intersection of these sets, without disclosing any information about their respective sets (except for the intersection part) during the calculation process. This ensures privacy protection while calculating correlations, eliminating the need for the clients to directly share tags with the FL server and thereby mitigating the risk of data leakage. Let Y_i^k denote the set of client i 's labels in round k, and let Y denote the target labels. The dataset of client i in round k is D_i^k. $D_i^{'k}$ represents the dataset that overlaps with the target labels, *i.e.,* $D_i^{'k} = \{(x,y)|y \in Y_i^k \cap Y\}$. We define γ_i^k as follows:

$$\gamma_i^k = \frac{|D_i^{'k}|}{|D_i^k|}, \tag{5}$$

Obviously, the larger γ_i^k is, the more suitable client i is for the training.

Data Effectiveness – Similarity: Clients with similar data appearing in the same training round may waste training resources, as we want to get as much data information as possible from the available clients for training. Therefore, we use a privacy-preserving method to measure the data similarity between clients, and select clients whose data information is as unique as possible to participate in each training round. *Hca* measures the similarity $s^k(i,j)$ of two clients' datasets D_i and D_j by a privacy-preserving method [38].[2] In particular, client i locally generates content embedding vectors $\phi_i^k = \{\phi_{i,m}^k | m \in [S_i^k]\}$, where each embedding vector $\phi_{i,m}^k \in \mathbb{R}^{L_\phi}$. Then a projection matrix $w \in \mathbb{R}^{l_\phi \times L_\phi}$ is selected to encode the L_ϕ-dimensional vector into l_ϕ-dimensional vector $h(\phi_{i,m}^k)$, where $l_\phi < L_\phi$ [51]. Each client computes the projection vector $h(\phi_{i,m}^k) = sgn(w \cdot \phi_{i,m}^k)$, where $sgn(\cdot)$ denotes signum function. Then the sketch of dataset D_i^k is $H_i^k = \sum_{m \in [S_i^k]} h(\phi_{i,m}^k)$. To protect the privacy of each sample, a randomized response mechanism is applied to generate a noisy sketch $\widehat{H}_i^k$ to replace H_i^k. Given the noisy content sketch of each client, the similarities between two clients' datasets D_i^k and D_j^k is defined as:

$$s^k(i,j) = \frac{\widehat{H}_i^k \cdot \widehat{H}_j^k}{|\widehat{H}_i^k||\widehat{H}_j^k|}. \tag{6}$$

If $s^k(i,j)$ is too large, the datasets of clients i and j are too similar, hence the training efficiency by selecting these two clients is low.

Problem Formulation. The goal in the second stage is to maximize the quality of clients and the data diversity[3] at the beginning of round k.

P2:

$$\max_{\mathbf{x}^k} \quad \sum_{i \in \mathbb{I}^k} U_i^k x_i^k - \sum_{i,j \in \mathbb{I}^k} s^k(i,j) x_i^k x_j^k \tag{7}$$

$$s.t. \quad M \le \sum_{i \in \mathbb{I}^k} x_i^k \le R, \tag{7a}$$

$$\widehat{C}_{tot}^{k,i} x_i^k \le \frac{C}{K}, \forall i \in \mathbb{I}^k, \tag{7b}$$

$$\gamma_i^k x_i^k \ge \gamma_0, \forall i \in \mathbb{I}^k, \tag{7c}$$

$$x_i^k \in \{0,1\}, \forall i \in \mathbb{I}^k. \tag{7d}$$

Constraint (7a) indicates that in round K, the central server selects at least M and at most R participating clients, where $R = \min\{2M, \frac{M+N}{2}\}$. Since clients may drop during training, more than M clients are selected before each round, and the top M training models are selected for the current round's aggregation. Note that constraint (7b) ensures that the estimated completion time does not exceed the average one-round maximum completion time according to the estimation result of **P1**. Constraint (7c) filters out mismatched clients by an upper bound γ_0.

Discussion on Privacy Trade-offs and Overheads. We use the PSI protocol to measure client data relevance to target task data. Despite its privacy benefits, PSI's computational and communication overheads may affect practical applications. However, since our focus is on optimizing FL performance, including model accuracy and job completion time, we use PSI in the client selection phase to ensure data relevance and minimize privacy risks. We don't extensively consider PSI's overheads for the following reasons:

[2]This method sketches each client's dataset by a low-dimensional vector based on JL-transformation [51] and protects the privacy of each sample using a random response mechanism. The high efficiency and low computation cost of this method has been proven in [38].

[3]Same as **P1**, two items in the objective function are normalized.

Acceptability of Computational Overhead. Although the computational complexity of the PSI protocol is proportional to the set size, some studies have shown that the computational overhead of the PSI protocol is acceptable in practical applications. For example, Pinkas et al. [50] proposed a scalable PSI protocol based on OT extension, which demonstrates good computational efficiency on large-scale datasets. Additionally, our client selection process is executed only once at the beginning of each training round, so the impact of computational overhead on overall performance is minimal.

Optimization of Communication Overhead. The communication overhead of the PSI protocol mainly comes from the interaction between clients and the server. However, by designing the protocol appropriately, the communication overhead can be significantly reduced. For example, Chen et al. [36] proposed a lightweight PSI protocol that optimizes the communication mechanism to bring the communication overhead within an acceptable range. Moreover, our system assumes reliable network connections between clients and the server, so communication overhead does not significantly impact the training process.

Reasonableness of Privacy Trade-offs. Although the PSI protocol offers significant privacy protection, our model does not need to extensively consider privacy trade-offs. This is because the PSI protocol already provides sufficient privacy protection to prevent client data leakage. For example, Li et al. [1] pointed out that combining PSI with differential privacy (DP) can further enhance privacy protection, but such enhancement is not necessary in our application scenario. Our goal is to optimize the overall performance of FL rather than pursue extreme privacy protection.

In summary, while the computational and communication overheads and privacy trade-offs of the PSI protocol are important considerations, in the context of this paper, these factors have minimal impact on overall performance. Therefore, we choose to use the PSI protocol in the client selection phase to ensure data relevance while minimizing potential privacy risks.

Challenges. Notice that **P2** is a 0-1 quadratic programming (QP). The heaviest k-subgraph problem (HSP) which is NP-hard [52] can be reduced to **P2** by ignoring constraints (7b) and (7c).

IV. Design of Hca

A. Design Overview

As shown in Fig. 1, *Hca* consists of three stages.

i. Pre-estimate stage. To meet the target job's weighted time and accuracy demand, we explore the specific mathematical expression of $\mathbb{E}[C_{tot}^K]$ and $\mathbb{E}[F(\boldsymbol{w}^K)]$ in **P1**, and rewrite **P1** to a complex non-convex optimization program **P3** expressed by three decision variables K, M and τ. Next, we relax the integral constraints in **P3** and prove that the relaxed **P3** is a multi-convex problem. We then leverage an efficient BCD method in Alg. 1 to solve it and obtain the value of K, M and τ.
ii. Client selection stage.
iii. Model aggregation stage.

We next introduce three stages in Sec. IV-B, Sec. IV-C and Sec. IV-D respectively.

B. Solving Pre-Estimation Problem

Problem Reformulation. We reformulate the objective function of **P1** into a function of the decision variables K, M and τ. The total training time C_{tot}^K consists of computation time and communication time. Let C_1^k denote the computation time of one iteration in round k, and C_2^k denote the communication time between participating clients and the central server in round k, then we have $C_{tot}^K = \sum_{k=1}^{K}(\tau C_1^k + C_2^k)$. Our intuition is that larger M leads to larger probability of suffering from straggler effect and risk of communication delay. Here, we adopt an approximate representation of $\mathbb{E}[C_{tot}^K]$ from [9][4]:

$$\mathbb{E}[C_{tot}^K] \approx K\left(\left(\frac{M-1}{N-1}\triangle C_1 + c_1\right)\tau + \overline{C}_2\right), \tag{8}$$

where c_1 is the smallest of C_1^k, $\triangle C_1$ is the range of C_1^k, *i.e.*, $\triangle C_1 = \max_{i,j\in\mathbb{I}^k}\{|C_{1i}^k - C_{1j}^k|\}$ and $\overline{C}_2$ is the average of all C_2^k, both of which can be obtained by historical information. As for the expression of $\mathbb{E}[F(\boldsymbol{w}^K)]$, we express it by the convergence upper bound [19].[5]

In order to obtain the specific expression of $\mathbb{E}[F(\boldsymbol{w}^K)]$ in **P1** with respect to the independent variables $\{K, M, \tau\}$, we introduce Lemma 1.

Lemma 1: The convergence upper bound after K rounds in the case of non-IID data distribution is given by

$$\mathbb{E}[F(\boldsymbol{w}^K)] - F(\boldsymbol{w}^*) \le \frac{A_1}{K\tau} + \frac{A_2\tau}{K}\left(1 + \frac{N-M}{M(N-1)}\right), \tag{9}$$

where A_1 and A_2 are constants and determined by model and system properties.

Proof: Please see Appendix A for the proof of Lemma 1.■

Then **P1** can be reformulated as following:

P3:

$$\min_{\tau,K,M} \quad \alpha\left(\frac{A_1}{K\tau} + \frac{A_2\tau}{K}\left(1 + \frac{N-M}{M(N-1)}\right) + F(\boldsymbol{w}^*)\right) + (1-\alpha)\left(K\left(\left(\frac{M-1}{N-1}\triangle C_1 + c_1\right)\tau + \overline{C}_2\right)\right) \tag{10}$$

$$s.t. \quad K\left(\left(\frac{M-1}{N-1}\triangle C_1 + c_1\right)\tau + \overline{C}_2\right) \le C, \tag{10a}$$

$$\frac{A_1}{K\tau} + \frac{A_2\tau}{K}\left(1 + \frac{N-M}{M(N-1)}\right) \le \varepsilon, \tag{10b}$$

$$K, M, \tau \in \mathbb{Z}^+, M \le N. \tag{10c}$$

P3 is a complex integer optimization problem. To solve it, we first relax constraint (10c) to $K, M, \tau \ge 1$ and $M \le N$. We denote the relaxed version of **P3** by $\tilde{\textbf{P3}}$. We next analyze the multi-convex property of $\tilde{\textbf{P3}}$, and propose Lemma 2.

[4]This approximation has been adopted and proven its effectiveness in [9], we omit the theoretical proof and will demonstrate that the solution of **P3** achieves a near-optimal performance of the original problem **P1** in Sec. VI-B.

[5]The convergence upper bound is a common and effective approach to estimating the model accuracy in literature [19].

Lemma 2: $\tilde{\mathbf{P3}}$ is non-convex but strong multi-convex.

Proof: Please see Appendix B for the proof of Lemma 2.■

Algorithm 1 Pre-Estimation With BCD

Input: $\alpha, C, \varepsilon, N, A_1, A_2, c_1, \triangle C_1, \overline{C}_2, \epsilon_0$, loss
Output: K^{**}, M^{**} and τ^{**}
1: choose a feasible solution $D_0 \leftarrow (K_0, M_0, \tau_0)$;
2: **for** $k = 1, 2, \cdots$ **do**
3: Substitude M_{k-1}, τ_{k-1} for M^*, τ^* in **sub1** to get K_k;
4: Substitude K_{k-1}, τ_{k-1} for K^*, τ^* in **sub2** to get M_k;
5: Substitude K_{k-1}, M_{k-1} for K^*, M^* in **sub3** to get τ_k;
6: **if** $||D_k - D_{k-1}|| < \epsilon_0$ **then**
7: $(K^*, M^*, \tau^*) \leftarrow (K_k, M_k, \tau_k)$;
8: Break;
9: **else**
10: $k \leftarrow k + 1$;
11: **end if**
12: **end for**
13: Substitute the eight rounding combinations of (K^*, M^*, τ^*) : $((\lfloor K^* \rfloor, \lfloor M^* \rfloor, \lfloor \tau^* \rfloor), (\lfloor K^* \rfloor, \lfloor M^* \rfloor, \lceil \tau^* \rceil), (\lfloor K^* \rfloor, \lceil M^* \rceil, \lfloor \tau^* \rfloor), (\lfloor K^* \rfloor, \lceil M^* \rceil, \lceil \tau^* \rceil), (\lceil K^* \rceil, \lfloor M^* \rfloor, \lfloor \tau^* \rfloor), (\lceil K^* \rceil, \lfloor M^* \rfloor, \lceil \tau^* \rceil), (\lceil K^* \rceil, \lceil M^* \rceil, \lfloor \tau^* \rfloor), (\lceil K^* \rceil, \lceil M^* \rceil, \lceil \tau^* \rceil))$ into the objective function of **P3** and choose the combination that is feasible and has the minimum value as (K^{**}, M^{**}, τ^{**});
14: Return (K^{**}, M^{**}, τ^{**}).

Algorithm Design. We leverage a widely used method, block coordinate decent (BCD) [15] algorithm, to solve $\tilde{\mathbf{P3}}$. BCD method has been proven to have good convergence for multi-convex problem in polynomial time [15]. The main idea of BCD is to cyclically solve the convex subproblems over each control variable while fixing the remaining variables. After obtaining the fractional solution, we pick out the best feasible integer solution by rounding fractional variables. The pre-estimation stage is presented in Alg. 1. Lines 1-12 use BCD method to solve $\tilde{\mathbf{P3}}$. Line 13 is the rounding process to restore the integer constraints in **P3**. We define the three kinds of subproblems are as follows:

$$\mathbf{sub1}: \mathbf{K}^* = \arg\min_{\mathbf{K}} \mathbf{f}(\mathbf{K}; \mathbf{M}^*, \tau^*) \text{with } (\mathbf{10a}), (\mathbf{10b}), \mathbf{K} \geq \mathbf{1}$$

$$\mathbf{sub2}:$$

$$\mathbf{M}^* = \arg\min_{\mathbf{K}} \mathbf{f}(\mathbf{M}; \mathbf{K}^*, \tau^*) \text{with } (\mathbf{10a}), (\mathbf{10b}), \mathbf{M} \leq \mathbf{N}, \mathbf{M} \geq \mathbf{1}$$

$$\mathbf{sub3}: \tau^* = \arg\min_{\mathbf{K}} \mathbf{f}(\tau; \mathbf{K}^*, \mathbf{M}^*) \text{with } (\mathbf{10a}), (\mathbf{10b}), \tau \geq \mathbf{1}$$

C. Solving One-Round Client Selection Problem

D. Model Aggregation

The details of Sec. IV-C and Sec. IV-D are described in our previous work [53].

Theorem 1: Hca can meet the job completion time and model accuracy requirements of the FL job.

Proof: Please see Appendix C.■

V. HCARL: SMART CLIENT SELECTION AND MODEL AGGREGATION BASED ON DEEP REINFORCEMENT LEARNING

In the previous sections, we proposed the *Hca* framework to optimize heterogeneous FL tasks. In practice, it is difficult to accurately model the FL training process using only traditional mathematical methods. **First**, the training of each round is affected by many factors, including heterogeneity in device capability, data distribution and so on. These factors are coupled and interact with each other. It is intractable to extract them into independent explicit constraints by using traditional mathematical methods. **Second**, given heterogeneous devices and unbalanced and non-IID data distribution, the aggregation weight of each local model should be carefully designed. The allocation of aggregation weights has a direct impact on the global model of each round. **Third**, although we manage to use privacy-preserving methods to obtain the data features, from the perspective of security, these methods always have the disadvantage of either inaccurate measurement or privacy leakage risk. Therefore, we consider client selection and aggregation weight allocation as joint decision variables and reformulate the client selection model with the objective of long-term weighted FL performance. And an optimization framework based on RL is proposed to solve it.

In this section, we discuss an improved FL training framework that utilizes DRL technique for the joint model, making optimal decisions in environments that are not fully known. We reformulate the selection-aggregation problem in Sec. V-A. Then we present our improved algorithm framework, *HcaRL*, in Sec. V-B.

A. Selection-Aggregation Model

Selection Criteria. Instead of only obtaining the information before the round k, we redefine $\widehat{q}_i^k$ and $\widehat{C}_{tot}^{k,i}$ to be the long-term training utility and completion time in FL system for all rounds, both of which can be expressed as the expectation of the cumulative discounted value. A discounting factor $\beta \in (0, 1]$ for different periods is introduced to calculate the cumulative value over training time, which is a basic technique in DRL that relates the value of quality and completion time to the time domain. Specifically, $\widehat{q}_i^k$ is defined as:

$$\widehat{q}_i^k = \sum_{t=1}^{T} \beta^{t-1} q_i^t x_i^k \tag{11}$$

Similarly, the estimated completion time of round k, $\widehat{C}_{tot}^{k,i}$ is also obtained from the historical information:

$$\widehat{C}_{tot}^{k,i} = \sum_{t=1}^{T} \beta^{t-1} C_{tot}^{t,i} x_i^k \tag{12}$$

where $C_{tot}^{t,i}$ is the completion time of round t.

We multiply the weight factor α and $(1 - \alpha)$ in front of $\widehat{q}_i$ and $\widehat{C}_{tot}^i$, in order to meet the time and accuracy trade-off. Our objective P can be expressed as:

$$P = \alpha \sum_{i=1}^{N} \widehat{q}_i - (1 - \alpha) \sum_{i=1}^{N} \widehat{C}_{tot}^i \tag{13}$$

Problem Reformulation. The objective in the second stage is to maximize the weighted sum of utility and the completion time at the beginning of round k. The decision variables are client selection set $\mathbf{x}^k = \{x_i^k \in \{0,1\} | i \in \mathbb{I}^k, k \in K\}$ and the aggregation weights set $\mathbf{w}^k = \{w_i^k \in [0,1] | i \in \mathbb{I}^k, k \in K\}$. The client selection and aggregation weight optimization problem aiming at maximizing the long-term weighted performance P can be reformulated as follows.

P6:

$$\max_{\mathbf{x}^k, \mathbf{w}^k} \quad P \tag{14}$$

$$s.t.\ M \leq \sum_{i=1}^{N} x_i^k \leq R, \tag{14a}$$

$$\widehat{C}_{tot}^{k,i} x_i^k \leq \frac{C}{K}, \forall i \in \mathbb{I}^k, \tag{14b}$$

$$x_i^k \in \{0,1\}, \forall i \in \mathbb{I}^k, \tag{14c}$$

$$w_i^k \in [0,1], \forall i \in \mathbb{I}^k. \tag{14d}$$

Constraint (14a)-(14c) are the same as constraints (7a)-(7c). Constraint (14d) indicates that the aggregation weight of client i in round k is between 0 and 1.

Challenges. The optimization objective is a function related to model loss, which changes with each round of training and cannot be measured by any explicit functional relationship.

B. Algorithm Design

To deal with the above challenges, we leverage a DRL approach based on the Deep Deterministic Policy Gradient (DDPG) algorithm. We formulate the original problem of model selection and aggregation at the central server of FL into a DRL paradigm.

1) Problem Transformation: To maximize our customized objective of the FL job, we transform **P6** to the Markov decision process (MDP), which is described by the tuple $\{s^k, a^k, r^k, s'^k\}$. Here s^k denotes a set of states and a^k denotes a set of actions. The reward r^k at global round k is computed by the reward function $\mathcal{R}$: $s^k \times a^k \to \mathbb{R}$ and future rewards are discounted by the factor $\beta \in [0,1]$. After the actions are conducted jointly by all the clients, the FL environment interacts with the agent, and then it returns the corresponding reward r^k to the agent. After every τ local updates, the FL environment turns to the next state s'^k, which is determined by the state transition probability $p(s^{k+1}|s^k, a^k)$ that comes from the state transition function $\mathcal{P}$: $s^k \times a^k \times s^k \to [0,1]$. Then, the agent observes the next state s'^k and stores the tuple $\{s^k, a^k, r^k, s'^k\}$ in its own replay buffer $\mathcal{R}_b$. Accordingly, the agent updates its actor and critic networks by samples from the replay buffer $\mathcal{R}_b$. After finishing the training process of DRL network, we use the trained DRL network to perform the dynamic aggregation weight allocation and client selection. The details of four essential elements are defined as following.

1) Agent. We consider the situation where one central server is responsible for the client selection and the aggregation procedure in each training round k. For such, we take the central server as the RL-based decision-making agent whose outputs are the impact factors of all the candidate clients indicating the contributions of locally trained models to the aggregated global one. The agent can observe the state information s^k from the external FL environment. Given the state information s^k, the agent determines clients' aggregation weights $\{w_i^k\}_{i \in \mathbb{I}^k}$ through its policy network. Then the client selection decision $\{x_i^k\}_{i \in \mathbb{I}^k}$ is jointly determined by w_i^k and the control parameter M, that is, the largest top-M clients of w_i^k are selected to participate in this training round, and the aggregation weights are re-allocated according to the weight proportion in w_i^k.

2) State. We hope that the state can reflect the relationship between the quality of the clients' locally trained model and the clients' data. To this end, we design the state of the FL environment consisting of: i) The losses of the global model on each client's dataset. The losses reflect correlations between clients' data and the global model's effectiveness for each client. ii) The weight of each client in the previous global round w^{k-1}. This parameter reflects the relationship between the decision variable and the resulting global model. iii) The current global round index k, which reflects the current training progress. Therefore, the state observed by the agent at global round k is represented by a vector $s^k = \{\{F(\omega_i^k), w_i^{k-1}\}_{i \in \mathbb{I}^k}, t\}$.

3) Action. The purpose of the agent is to get the weight of each client's influence on the aggregation in the upcoming kth training and the resulting client selection strategy. Given the input state, the agent decides the action at global round k, *i.e.*, the aggregation weights $\{w_i^k\}_{i \in \mathbb{I}^k}$ for all the clients. The selection variable x_i^k is decided accordingly. Specifically, after getting the aggregation weight w_i^k of each client x_i^k for the incoming round k, we descending rank the clients according the weights, and get the top M clients with the largest weights. Then their selection variables are set to 1, *i.e.,* $x_i^k = 1$, which means that they will participate in the training round k. Accordingly, for the other clients, the selection variables are set to zero, $x_i^k = 0$.

4) Reward. Given the action a^k, the agent receives a reward $r^k \in \mathbb{R}$ from the environment. The reward is supposed to increase when P^k increases. One penalty term is added in the reward to punish the agent for choosing actions that violate constraints (19b). Therefore, the reward $r^k \in \mathbb{R}$ at round k is defined as:

$$r^k = P^k - p^k \tag{15}$$

where P^k indicates the objective value defined by Eq. (13) in round k and p^k indicates the penalty that the training time of the current training round does not meet the time requirements, which is set as a function that increases linearly with the growth of the time beyond the specified time, we set $p^k = 0.5k$. Otherwise, p^k is set to zero.

2) Design of HcaRL: As can be seen from the above definition of the four elements of RL customized for the system model, it is challenging to model the state transition probability for FL training, especially under the influence of non-IID data. In addition, the high dimensional continuous state space and action space also make us unable to use the traditional deep Q-network (DQN) algorithm for discrete state and action space. Inspired by DDPG algorithm, *HcaFL* is

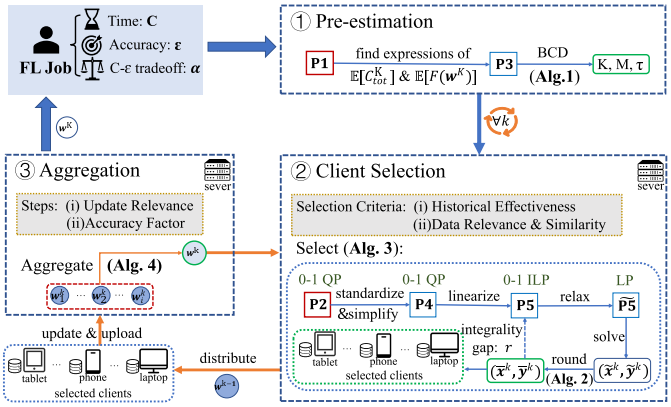

Fig. 1. Architecture of *Hca*.

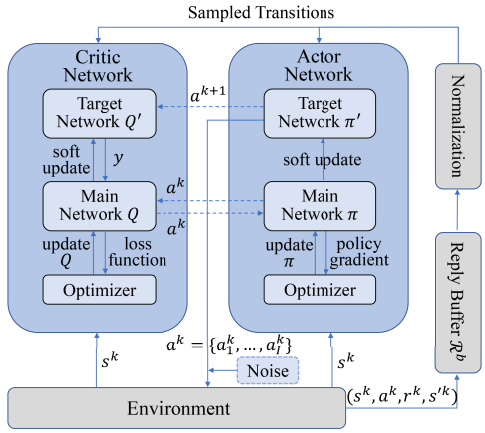

Fig. 2. Structure of *HcaRL*'s Training Network.

Algorithm 2 *HcaRL*, Aggregation Weight Allocation and Client Selection

Initialize the weights of critic network ζ and actor network θ, target networks $\zeta' \leftarrow \zeta$; $\theta' \leftarrow \theta$, replay buffer $\mathcal{R}_b$, hyper-parameters η, β.
for episode $1, 2, \ldots, EP$ **do**
  Initialize a random process $\mathcal{N}$ for exploration of action;
  Receive the initial state s^1;
  for $k = 1, 2, \ldots, K$ **do**
    The agent decides the aggregation weight $\{w_i^k\}_{i\in\mathbb{I}^k}$ based on the exploration and policy;
    Sort the clients by the w_i^k in descending order, for the top M clients, set $x_i^k \leftarrow 1$, for the remaining clients, set $x_i^k \leftarrow 0$;
  end for
  Using $\{w_i^k, x_i^k\}_{i\in\mathbb{I}^k}$ to perform one round FL procedures,
  i.e., execute action a^k;
  Finish global aggregation for round k, and receive the corresponding rewards r^k according to Eq. (15);
  Observe new state s'^k;
  Store $\{s^k, a^k, r^k, s'^k\}$ in replay buffer $\mathcal{R}_b$;
  Sample a random mini-batch of $\mathcal{V}$ experiences $\{s_j^k, a_j^k, r_j^k, s_j'^k\}$ from $\mathcal{R}_b$;
  Update critic by minimizing the loss $L(Q) = \frac{1}{|\mathcal{V}|}\sum_{\mathcal{V}}(Q(s^k, a^k) - (r^k + \beta Q'(s'^k, a'^k)))^2$;
  Update actor by maximizing the policy objective update
  network function according to Eq. (16);
  Update the corresponding parameters of target network:
  $\theta' \leftarrow \eta\theta + (1-\eta)\theta'$
  $\zeta' \leftarrow \eta\zeta + (1-\eta)\zeta'$
end for

designed to handle the above MDP problem. Fig. 2 illustrates DDPG-based workflow.

As shown in Fig. 2, the agent maintains two components: i) the actor network, a policy network that determines the next action, and ii) the critic network, a value network that assesses the goodness of the state caused by the action. An action decision is made by the actor for the current state in accordance with policy π. The actor's chosen action is assessed by the critic network using a state-action function $Q(\cdot)$. Both of the two networks utilize two deep neural networks (DNN) subnets with the same structure: the main network and the target network, to build the learning agent. The roles of the two subnets are: the main network interacts directly with the environment and it is updated depending on the feedback from the environment; the target network is a soft copy of the main network and is used as a reference point for the update of the main network. The main idea of DDPG is to learn the state-action function corresponding to the optimal policy π^*, which is accomplished by gradually training the networks for the actor and the critic until convergence. Transitions for the training stage are saved in an experience replay buffer of size $|\mathbb{R}_b|$.

Specifically, in the k-th transition $\{s^k, a^k, r^k, s'^k\}$, the actor network aims to maximize the expected cumulative reward:

$$J(\pi_\theta) = \int d^\pi(s^k) Q(s, \pi_\theta(s^k), \zeta)\, ds = \mathbb{E}_{s^k\sim d^\pi}[Q(s^k, \pi_\theta(s^k), \zeta)] \quad (16)$$

where ζ is the critic network, θ is the actor network. Eq. (16) calculates the expected cumulative reward of the policy by integrating over all possible states s^k according to the state distribution d^π induced by the policy π_θ, and computing the expected Q-values for actions chosen by the policy in each state.

The gradient update method of the actor is the gradient ascent method:

$$\theta = \theta + \nabla_\theta J(\pi_\theta) = \theta + \nabla_\theta Q(\pi_\theta) \quad (17)$$

where π_θ denotes the policy network, θ denotes its parameters, Q denotes the critic network, and ζ denotes its parameters. $d^\pi(s)$ indicates the probability distribution of the state under the policy.

In order for the Q value network to accurately assess the actor, the critic network's objective is to minimize the loss:

$$L(Q) = \mathbb{E}[(Q(s^k, a^k) - y)^2] \tag{18}$$

and

$$y = r^k + \beta Q'(s'^k, a'^k) \tag{19}$$

where $Q'(\cdot)$ is the state-action function for the target network. Since the loss function is continuously differentiable, $L(Q)$ can be modified using its gradient.

Algorithm Details. The client selection and aggregation weight optimization strategy based on the DDPG algorithm *HcaRL* is presented in Alg. 2. *HcaRL* works as follows.

- *Firstly,* the actor and the critic are initialized. In addition, an experience replay buffer $\mathcal{R}_b$ and some hyper-parameters are also initialized (line 1).
- *Secondly,* given the local state information $s^k = \{\{F(\omega_i^k), w_i^{k-1}\}_{i\in\mathbb{I}^k}, t\}$ from the FL environment, the agent determines clients' aggregation weight $\{w_i^k\}_{i\in\mathbb{I}^k}$ based on exploration noise and its own policy (line 6). Then the clients are sorted in the descending order of w_i^k. The top *M* clients are selected for the aggregation using our client selection strategy. If the client *i* is chosen, $x_i^k = 1$ and reallocate the aggregation weights for these chosen clients according to the weight proportion. Note that some tricks are used to improve the effectiveness of *HcaRL*. The input (*i.e.*, state of the agent) should be normalized to avoid over-fitting. After the actions are conducted jointly by all the clients, FL environment interacts with the agent, and then it returns the corresponding reward r^k to the agent. After local updates, the FL environment turns to the next state (line 10). The agent then observes the next state s'^k and stores the tuple $\{s^k, a^k, r^k, s'^k\}$ in its own replay buffer $\mathcal{R}_b$, which contains the experiences of historical tuples (line 11).
- *Thirdly,* the agent executes the following procedures to update its actor and critic networks throughout the entire episode. The agent samples the random mini-batch of $\mathcal{V}$ samples from $\mathcal{R}_b$ (line 12). Then, Q of the behavior critic is updated by minimizing the difference between $Q(s^k, a^k)$ and $r^k + \beta Q'(s'^k, a'^k)$ among $\mathcal{V}$ samples (line 14). Similarly, parameter of the actor network is updated based on gradient ascent computation (line 15).
- *Finally,* the target network parameters are changed using the idea of soft update when the agent eventually updates their behavior networks (lines 16-18).

3) FL Architecture With HcaRL: Next we summarize the whole process of the FL training framework that integrates *HcaRL* as each round's client selection and aggregation procedure.

When a FL job arrives with its own requirements including completion time *C*, model accuracy ε and requirement preference α, *HcaRL* first customizes the control parameters of the whole training process, including number of training rounds *K*, number of participating clients per round *M*, and number of local iterations per round τ according to Alg. 1. Then before each training round, the knowledge related to training is first fed into our DRL module, which determines the participating client subset and each participating client's aggregation weight. Following this decision, clients collaborate on the current round of training, and finally get a global model meeting the job requirement.

VI. Performance Evaluation

A. Experimental Setup

FL Platform and Model. The testbed experiments are carried out on the FL platform FAVOR [16]. We create 100 clients and each client with a PyTorch model is simulated as a thread running synchronously in a global iteration. We conduct experiments to train a classic CNN model with 5×5 convolutional layers. The output channels of the first and second layers are 16 and 32 respectively, and each layer has a 2×2 max pooling. The training task is multi-classification, and we use cross-entropy as the loss function. By default, we set $\alpha = 1$.

Datasets. We evaluate the performance of our frameworks using three widely-used datasets: MNIST, Fashion-MNIST (FMNIST), and CIFAR-10. These datasets are chosen for their diverse characteristics and suitability for federated learning tasks.

- MNIST: Contains 60,000 training images and 10,000 test images of handwritten digits (28×28 grayscale images).
- FMNIST: Contains 60,000 training images and 10,000 test images of fashion products (28×28 grayscale images).
- CIFAR-10: Contains 50,000 training images and 10,000 test images of colored images (32×32) from 10 classes.

Simulating Non-IID Data Distributions. To simulate non-IID data distributions across clients, we follow a specific method to allocate and update client datasets. This method ensures that each client's dataset reflects the characteristics of real-world data, which is often imbalanced and diverse. **i) Initial Data Allocation:** At the beginning of each round, each client is assigned an initial dataset from the main dataset (e.g., MNIST, FMNIST, or CIFAR-10). In the initial data allocation for each client, a major class is randomly selected from the dataset (e.g., digit "3" in MNIST), and 70% of the client's dataset is composed of randomly chosen samples from this major class using numpy.random.choice to ensure diversity. The remaining 30% of the dataset is filled with samples randomly selected from other classes to introduce data diversity and imbalance, maintaining the non-IID characteristic while ensuring each client's dataset is representative of the overall dataset. **ii) Dynamic Data Update:** After each training round, clients receive a new subset of data to simulate dynamic changes, allocated in the same 7:3 ratio between the major and minor classes as the initial setup. Specifically, 10% of the client's existing data is randomly selected and replaced with this new data, ensuring the dataset evolves over time to reflect real-world scenarios where client data changes between rounds. **iii) Pre-Communication Data Preparation:** Before each communication round, we calculate the relevance γ_i^k by Eq. (5), and the similarity by Eq. (6). We use cross-entropy to measure the loss. The values of job completion time and

TABLE II
THE VALUE OF THREE PARAMETERS FOR DIFFERENT α

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	1
K	2	4	5	7	10	21	23	27	33	60
M	12	12	12	5	12	12	12	12	12	10
τ	3	3	3	7	3	3	3	3	3	10

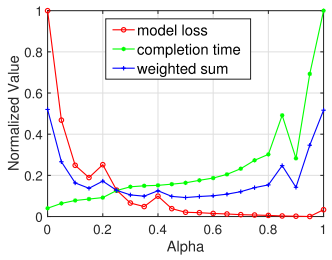

Fig. 3. Normalized completion time, model loss and their weighted sum.

loss are mapped to the range of 0-1 for normalized processing, according to the actual observed time and loss range.

Benchmarks. Taking into account many factors, such as problem relevance, timeliness and advancement, we selected three benchmark algorithms for performance comparison: *FedAvg*, *FAIR*, and *CMFL*. i) *FedAvg* is one of the fundamental and widely-used federated learning frameworks. By comparing our proposed frameworks with *FedAvg*, we can assess the overall performance improvement of it. *FAIR* and *CMFL*, on the other hand, focus on optimizing different stages and have achieved significant improvements. ii) *FAIR* emphasizes optimization during the client selection stage. By defining and measuring historical quality of clients, *FAIR* selects high-quality participating clients, effectively enhancing training performance. Comparison with *FAIR* allows us to measure *Hca*'s advancement in client selection. iii) *CMFL* prioritizes optimization by filtering irrelevant parameters during the aggregation stage. By comparing with *CMFL*, we can evaluate the performance improvement of *Hca* in the aggregation stage. We will compare these algorithms with our method from the perspectives of different stages and overall framework to comprehensively demonstrate the performance improvement brought by our algorithm.

B. Performance of Pre-Estimation

We first verify the efficiency of the first stage: the pre-estimation of three parameters K, M and τ. We train a CNN model by adopting *Hca* on non-IID MNIST. To make the time and loss have the same order of magnitude, we first normalize two values by scaling down their values respectively. We then vary the value of α and call Alg. 1 to obtain the value of K, M and τ, and continue the training by involving the second and third stage in *Hca*. Table II shows the value of three parameters for different α.

Impact of α**.** We verify the effect of α on job requirements. Specifically, we compare the model loss, the completion time and their weighted sum with different α. From Fig. 3, we can observe that when the value of α increases, the completion time (green line) decreases, while the model loss (blue line) increases, which indicates that α has the trade-off effect between the time and accuracy requirement.

TABLE III
HYPERPARAMETERS OF DDPG

Parameter	Value
Time step K	60
Minibatch size	32
Discounting factor β	0.9
soft update υ	0.01
Learning rate	0.001
Size of replay buffer $\mathcal{R}_b$	10000
Optimizer	Adam
Activation function	ReLU

C. Performance of Client Selection

D. Performance of Model Aggregation

E. Performance of Hca

The details of Sec. VI-C- VI-E are described in our previous work [53].

F. Performance of HcaRL

In this section, the total FL training framework employing *HcaRL* will be contrasted with *Hca* and other cutting-edge algorithms in terms of training completion time, model accuracy, and personalized requirement.

Settings. Same as above, the FL training system is comprised of 1 central server and 100 clients. The FL job is an image classification task, and a CNN model is trained on three datasets: MNIST, Fashion-MNIST, and CIFAR-10, which are distributed to clients in a non-IID manner. For all of the experiments, we employ the stochastic gradient descent (SGD) local solver with a learning rate of 0.01 and a local batch size of 10. For the structure of DRL, the input of the actor network is the dimension of the state, *i.e.*, $K + 1 = 10 + 1 = 11$. Through fully connected transformation and activation function transformation, the actor outputs 100 action values. The input of the critic network is the dimension of the state and the action, *i.e.*, $100 + 11 = 111$. These two components are combined and fed into the full connection layer after the features are extracted through the full connection layer. The network then outputs a scalar evaluation value via the full connection layer, which can be used to evaluate the actions given by the actor. Note that to ensure the DRL training model stably converge, the input should be normalized to avoid over-fitting. Other important simulation parameters are listed in Table III.

Benchmarks. To evaluate the performance of *HcaRL*, the following three benchmarks are compared.

- *Hca*: which is the FL framework we proposed in Sec. IV.
- FedAvg [4]: which randomly selects participant clients in each round and allocates aggregation weights according to the sizes of participant clients' dataset.

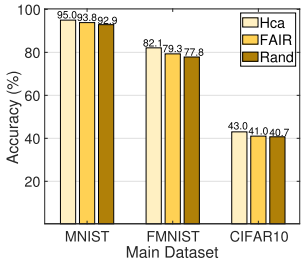

Fig. 4. Test accuracy with different selection strategies on different non-IID datasets.

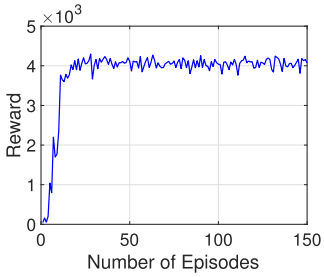

Fig. 5. The convergence curve of *HcaRL*'s Training.

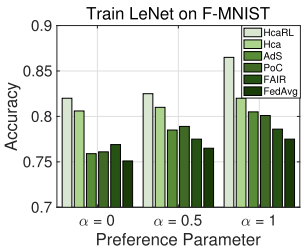

Fig. 6. Test accuracy for LeNet task on non-IID FMNIST.

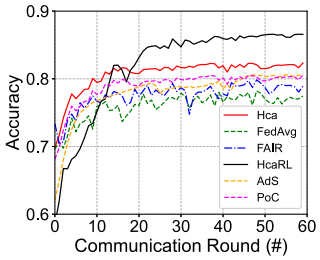

Fig. 7. Test accuracy curve for LeNet task on non-IID FMNIST.

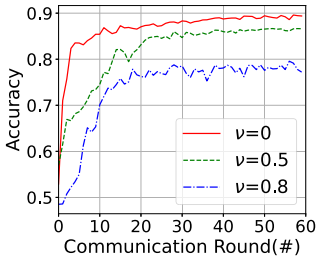

Fig. 8. Test accuracy curve of *HcaRL* for LeNet task on non-IID FMNIST.

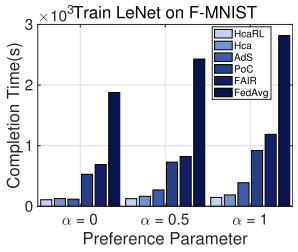

Fig. 9. Completion time for LeNet task on non-IID FMNIST.

- FAIR [17]: which selects participant clients with the best training quality greedily and aggregates the clients by the same way of FedAvg.
- AdS [41]: which optimizes the client sampling probabilities by considering both communication delay and data importance to minimize the convergence time.
- PoC [37]: which chooses the clients with the highest local loss to accelerate convergence and gradually reduce the size of the candidate client set.

Convergence of HcaRL. Fig. 5 depicts the convergence curve in terms of average reward during the training process of the *HcaRL*. One episode of the *HcaRL* begins with the initialization of a FL job and ends when the job achieves the target accuracy or reaches the required completion time. As shown in Fig. 5, the average reward value grows as the episode increases and eventually converges at around 40 episodes.

Performance of accuracy and convergence. We test the average accuracy achieved using different algorithms when three types of FL jobs arrive with different requirements (*i.e.*, α =0, 0.5, and 1). Fig. 6 shows the accuracy comparison results of LeNet model training on the Fashion-MNIST dataset. It can be seen that on the basis of *Hca*, *HcaRL* has a maximum accuracy improvement of 15% for each type of FL job. Fig. 7 shows the curve of test accuracy for each algorithm in the first 60 rounds of FL training, which reflects the model convergence of FL training. As shown in Fig. 7, we can see that the accuracy achieved by *HcaRL* is higher than that of the other three algorithms. Specifically, the training accuracy of *HcaRL* is 5% higher than that of *Hca* and better than the other three algorithms.

Impact of non-IID level. Fig. 8 indicates the test accuracy curve on data distribution of different non-IID degrees ($\nu = 0, 0.5$, and 0.8). In our experiment, the value of non-IID degree is implemented as follows: for example, ν =0.3 represents only 30% of the data on each client belongs to one label, and the remaining 70% of the data belongs to the other labels. It can be easily observed that the higher the non-IID degree of data distribution is, the higher the value of ν should be set. As shown in Fig. 8, the higher degree of non-IID in the data distribution has a greater negative impact on *HcaRL*, which is consistent with the common sense. In addition, the accuracy fluctuation range of different non-IID degrees is less than 0.1. It can be concluded that the non-IID degree has a small impact on *HcaRL*'s performance.

Performance of job completion time. As shown in Fig. 9, the completion time of *HcaRL* is shorter than that of the

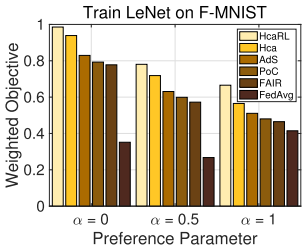

Fig. 10. Weighted objective for LeNet task on non-IID FMNIST.

other three algorithms for three types of FL jobs with different requirements. Specifically, *HcaRL* can reduce the job completion time by 23.6%, compared to *Hca*.

Performance of weighted objective. As shown in Fig. 10, the weighted objective value of *HcaRL* is the highest among the four algorithms for three types of FL jobs with different requirements. Specifically, *HcaRL* improves the normalized weighted objective value by $2.92\times$, over the baselines in the best-case scenario, and it increases the objective value by 17.8%, compared to *Hca*. We can conclude that *HcaRL* performs the best in all cases.

VII. Conclusion

In this paper, we propose *Hca*, a heterogeneous FL framework for balancing job completion time and model accuracy. Different from existing literature, our framework consists of three stages. First, we determine the number of training rounds, the number of iterations and the number of participating clients in each round, to satisfy FL job's requirement on job completion time and model accuracy. Second, a multi-criteria client selection framework is involved in selecting the most efficient clients for the FL job. At last, we tailor an improved model aggregation algorithm to further optimize the quality of the FL model. On the basis of *Hca*, we further propose a RL-based smart client selection and model aggregation algorithm, *HcaRL*. The extensive results from testbed experiments based on real-world data verify that *Hca* and *HcaRL* achieve near-optimal performance in both model accuracy and job completion time, compared with existing FL frameworks. However, the historical effectiveness metric used for client selection may not always reflect real-time changes in client capabilities or data distribution, limiting adaptability to dynamic environments. In the future work, we will develop real-time adaptation strategies to dynamically adjust to client and data changes during training, aiming to enhance the robustness and scalability of the framework. We will also incorporate additional objectives, such as energy efficiency or fairness, into the optimization process to address broader concerns in FL deployments.

References

[1] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," 2019, *arXiv:1909.05830*.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[3] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. PMLR AISTATS*, 2016, pp. 1273–1282.

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[6] A. Hard et al., "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[7] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2457–2471, Apr. 2021.

[8] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," 2020, *arXiv:2010.01264*.

[9] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

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

[11] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.

[12] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.

[13] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton, "Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation," 2021, *arXiv:2101.00787*.

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

[15] Y. Xu, "Block coordinate descent for regularized multi-convex optimization," Dept. Appl. Comput. Math., Rice Univ., Houston, TX, USA, Tech. Rep., 2013.

[16] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1698–1707.

[17] Y. Deng et al., "FAIR: Quality-aware federated learning with precise user incentive and model aggregation," in *Proc. IEEE Conf. Comput. Commun.*, Milwaukee, WI, USA: Quality, May 2021, pp. 1–10.

[18] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. MIT Press NIPS*, vol. 30, 2017, pp. 4427–4437.

[19] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. ICLR*, 2020.

[20] S. U. Stich, "Local SGD converges fast and communicates little," 2018, *arXiv:1805.09767*.

[21] Z. Hu, D. Li, K. Yang, Y. Xu, and B. Peng, "Optimizing data distributions based on Jensen–Shannon divergence for federated learning," *Tsinghua Sci. Technol.*, vol. 30, no. 2, pp. 670–681, Apr. 2025.

[22] W. Liu, X. Xu, J. Wu, and J. Jiang, "Federated meta reinforcement learning for personalized tasks," *Tsinghua Sci. Technol.*, vol. 29, no. 3, pp. 911–926, Jun. 2024.

[23] A. Xiong et al., "A multi-task based clustering personalized federated learning method," *Big Data Mining Analytics*, vol. 7, no. 4, pp. 1017–1030, Dec. 2024.

[24] R. Liu, S. Yu, L. Lan, J. Wang, K. Kant, and N. Calleja, "A remedy for heterogeneous data: Clustered federated learning with gradient trajectory," *Big Data Mining Analytics*, vol. 7, no. 4, pp. 1050–1064, Dec. 2024.

[25] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–9.

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

[27] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 954–964.

[28] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6341–6345.

[29] D. Huba et al., "Papaya: Practical, private, and scalable federated learning," in *Proc. Mach. Learn. Syst.*, vol. 4, 2021, pp. 814–832.
[30] J. Nguyen et al., "Federated learning with buffered asynchronous aggregation," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 3581–3607.
[31] J. Liu et al., "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 674–690, Feb. 2023.
[32] S. Reddi et al., "Adaptive federated optimization," 2020, *arXiv:2003.00295*.
[33] N. Hung Nguyen et al., "FedDRL: Deep reinforcement learning-based adaptive aggregation for non-IID data in federated learning," 2022, *arXiv:2208.02442*.
[34] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[35] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," 2020, *arXiv:2007.15197*.
[36] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," 2020, *arXiv:2010.13723*.
[37] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *Proc. AISTATS*, 2022, pp. 10351–10375.
[38] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.
[39] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. 15th USENIX Symp. Operat. Syst. Design Implement.*, 2021, pp. 19–35.
[40] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4385–4395, Mar. 2022.
[41] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *Proc. IEEE Conf. Comput. Commun.*, May 2022, pp. 1739–1748.
[42] C. Li, X. Zeng, M. Zhang, and Z. Cao, "PyramidFL: A fine-grained client selection framework for efficient federated learning," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2022, pp. 158–171.
[43] R. Sun et al., "FedMSA: A model selection and adaptation system for federated learning," *Sensors*, vol. 22, no. 19, p. 7244, Sep. 2022.
[44] A. C. Castillo J, E. C. Kaya, L. Ye, and A. Hashemi, "Equitable client selection in federated learning via truncated submodular maximization," in *Proc. IEEE 63rd Conf. Decis. Control (CDC)*, Dec. 2024, pp. 5496–5502.
[45] E. K. Burke, E. K. Burke, G. Kendall, and G. Kendall, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Cham, Switzerland: Springer, 2014.
[46] X. Zhang, X. Guo, and X. Zhang, "Bidding modes for renewable energy considering electricity-carbon integrated market mechanism based on multi-agent hybrid game," *Energy*, vol. 263, Jan. 2023, Art. no. 125616.
[47] D. Yang et al., "DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5162–5178, May 2024.
[48] V. Nasrollahi, G. Moslehi, and M. Reisi-Nafchi, "Minimizing the weighted sum of maximum earliness and maximum tardiness in a single-agent and two-agent form of a two-machine flow shop scheduling problem," *Oper. Res.*, vol. 22, no. 2, pp. 1403–1442, Apr. 2022.
[49] G. Damaskinos, R. Guerraoui, A.-M. Kermarrec, V. Nitu, R. Patra, and F. Taiani, "FLeet: Online federated learning via staleness awareness and performance prediction," in *Proc. 21st Int. Middleware Conf.*, Dec. 2020, pp. 163–177.
[50] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Trans. Privacy Secur.*, vol. 21, no. 2, pp. 1–35, 2018.
[51] C. Biswas, D. Ganguly, D. Roy, and U. Bhattacharya, "Privacy preserving approximate K-means clustering," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1321–1330.
[52] D. G. Corneil and Y. Perl, "Clustering and domination in perfect graphs," *Discrete Appl. Math.*, vol. 9, no. 1, pp. 27–39, Sep. 1984.
[53] R. Zhou, R. Wang, J. Yu, B. Li, and Y. Li, "Heterogeneous federated learning for balancing job completion time and model accuracy," in *Proc. IEEE 28th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Jan. 2023, pp. 562–569.

Ruobei Wang received the B.S. degree from the School of Mathematics and Statistics, Wuhan University, China, in 2020. She is currently pursuing the master's degree from the School of Cyber Science and Engineering, Wuhan University. Her research interests include edge computing, federated learning, online learning, and network optimization.

Ruiting Zhou (Member, IEEE) received the Ph.D. degree from the Department of Computer Science, University of Calgary, Canada, in 2018. She is currently a Professor with the School of Computer Science and Engineering, Southeast University. She has published research papers in top-tier computer science conferences and journals, including IEEE INFOCOM, ACM MobiHoc, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and IEEE TRANSACTIONS ON MOBILE COMPUTING. Her research interests include cloud computing, machine learning, and mobile network optimization. She serves as the TPC Chair for INFOCOM Workshop-ICCN from 2019 to 2024. She also serves as a reviewer for international conferences and journals, such as IEEE INFOCOM, IEEE ICDCS, IEEE/ACM IWQoS, IEEE SECON, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE TRANSACTIONS ON MOBILE COMPUTING.

Jieling Yu received the B.E. and master's degrees from the School of Cyber Science and Engineering, Wuhan University, China, in 2021 and 2024, respectively. She is currently pursuing the Ph.D. degree with The Hong Kong University of Science and Technology. Her research interests include edge computing, federated learning, online learning, and network optimization.

Bo Li (Fellow, IEEE) received the B.Eng. degree (summa cum laude) in computer science from Tsinghua University, Beijing, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts at Amherst. He was a Cheung Kong Visiting Chair Professor at Shanghai Jiao Tong University between 2010 and 2016 and the Chief Technical Advisor of ChinaCache Corporation (NASDAQ: CCIH), a leading CDN provider. He is currently a Chair Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. He made pioneering contributions in multimedia communications and the internet video broadcast, in particular coolstreaming system, which was credited as the first large-scale peer-to-peer live video streaming system in the world. It attracted significant attention from both industry and academia and received the Test-of-Time Best Paper Award from IEEE INFOCOM in 2015.

Yuqing Li (Member, IEEE) received the B.S. degree in communication engineering from Xidian University, Xi'an, China, in 2014, and the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019. She was a Researcher at the Huawei Hong Kong Research Center from 2020 to 2022 and a Post-Doctoral Fellow at The Hong Kong University of Science and Technology from 2019 to 2020. She is currently an Associate Professor at the School of Cyber Science and Engineering, Wuhan University, China. Her research interests include data-intensive and machine learning systems, cloud and edge computing, and data privacy and security.