Context-Aware Online Client Selection for Hierarchical Federated Learning

Zhe Qu, *Student Member, IEEE*, Rui Duan, *Student Member, IEEE*, Lixing Chen, *Member, IEEE*, Jie Xu, *Senior Member, IEEE*, Zhuo Lu, *Senior Member, IEEE*, and Yao Liu, *Senior Member, IEEE*

Abstract—Federated Learning (FL) has been considered as an appealing framework to tackle data privacy issues of mobile devices compared to conventional Machine Learning (ML). Using Edge Servers (ESs) as intermediaries to perform model aggregation in proximity can reduce the transmission overhead, and it enables great potential in low-latency FL, where the hierarchical architecture of FL (HFL) has been attracted more attention. Designing a proper client selection policy can significantly improve training performance, and it has been widely investigated in conventional FL studies. However, to the best of our knowledge, systematic client selection policies have not yet been fully studied for HFL. In addition, client selection for HFL faces more challenges than conventional FL (e.g., the time-varying connection of client-ES pairs and the limited budget of the Network Operator (NO)). In this article, we investigate a client selection problem for HFL, where the NO learns the number of successful participating clients to improve training performance (i.e., select as many clients in each round) as well as under the limited budget on each ES. An online policy, called Context-aware Online Client Selection (COCS), is developed based on Contextual Combinatorial Multi-Armed Bandit (CC-MAB). COCS observes the side-information (context) of local computing and transmission of client-ES pairs and makes client selection decisions to maximize NO's utility given a limited budget. Theoretically, COCS achieves a sublinear regret compared to an Oracle policy on both strongly convex and non-convex HFL. Simulation results also support the efficiency of the proposed COCS policy on real-world datasets.

Index Terms—Hierarchical federated learning, client selection, contextual combinatorial multi-armed bandit

1 Introduction

Federated Learning (FL) [1], [2], [3] has become an attractive ML framework to address the growing concerns of transmitting private data from distributed clients (e.g., mobile devices) to a central cloud server by leveraging the ever-increasing storage and computing capabilities of the client devices. In each FL round, clients train local models using their local data and the cloud server aggregates local model updates to form a global model. Because only local model information is exchanged in FL rather than the local data, FL preserves the data privacy of the clients and hence has found applications in a wide range of problems, such as next-word prediction [4] and image classification [5].

A main bottleneck that limits the performance of FL is the delay variability among individual clients due to their local training and model data transfer via the wireless network. In standard FL, the cloud server has to wait until receiving the training updates from all the clients before processing any next step. Therefore, straggler clients who have unfavorable wireless links or low computation capabilities may dramatically slow down the whole FL process [6], [7]. This is the so-called "straggler effect". Various approaches have been proposed to mitigate the "straggler effect". For example, model quantization [8] and gradient sparsification [9] schemes aim to directly reduce the transferred data size and the model training complexity, thereby reducing all clients' training and transmission delay. Asynchronous FL [10], [11] allows clients to train and upload training data in an asynchronous manner, and hence the cloud server does not have to wait for the slow clients to process the next step. Another mainstream and proven effective approach to address the straggler problem is client selection, which reduces the probability of straggler clients participating in FL by judiciously selecting clients in every FL round. However, these mechanisms mainly focus on the traditional FL and mitigate the straggler effect based on designing new mechanisms, which is not easy to solve the straggler effect from FL wireless networks (e.g., large distance of clients-Cloud Server (CS) and unstable connection). Thanks to the hierarchical architecture, some existing studies [12], [13], [14] propose Hierarchical FL (HFL) including multiple Edge Servers (ESs) which reside between the single CS and the large number of clients. Instead of communicating to the CS, clients in HFL only need to download/upload the training model updates to the nearest ESs. This significantly

- *Zhe Qu, Rui Duan, and Zhuo Lu are with the Department of Electrical Engineering, University of South Florida, Tampa, FL 33620 USA. E-mail: {zhequ, ruiduan, zhuolu}@usf.edu.*
- *Lixing Chen is with the Institute of Cyber Science and Technology, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai 200240, China. E-mail: lxchen@sjtu.edu.cn.*
- *Jie Xu is with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146 USA. E-mail: jiexu@miami.edu.*
- *Yao Liu is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA. E-mail: yliu@cse.usf.edu.*

Manuscript received 3 December 2021; revised 20 June 2022; accepted 24 June 2022. Date of publication 28 June 2022; date of current version 23 August 2022. The work of Zhe Qu, Rui Duan, Zhuo Lu, and Yao Liu was supported by NSF under Grant CNS-2044516. The work of Jie Xu was supported by NSF under Grants ECCS-2033681, ECCS-2029858, and CNS-2044991. (Corresponding author: Zhe Qu.) Recommended for acceptance by Y. Yang. Digital Object Identifier no. 10.1109/TPDS.2022.3186960

reduces the communication time of the slowest client located far from the CS and provides more stable connection to save training time. HFL is able to achieve a faster convergence speed than the traditional FL architecture both theoretically [12], [13] and empirically [14].

Although several learning algorithms have been designed for HFL [12], [13], [14], simplified assumptions have been made that all clients participate in each round of model parameter aggregation. This is weak for the straggler effect since each ES should wait for the slowest client in each edge aggregation round. Hence, it is necessary to design a new client selection mechanism for HFL. However, it is not straightforward to apply existing client selection solutions [7], [15], [16] to HFL due to several unique challenges that HFL faces. First, since the service area of an ES is much more restricted than CS and contains overlapping areas, the accessible clients of each ES are time-varying. This time-varying characteristic makes the client behavior of opportunistic communication more complicated, and Network Operator (NO) must carefully select the client to the corresponding ES in the overlapping area. Second, since the advantage of HFL is to deal with the straggler problem, how to design an efficient client selection policy is more important than traditional FL. Third, the client selection decision needs to be determined based on many uncertainties in the HFL network conditions (e.g., the traffic pattern of client-ES pair and available computation resources of clients), which affect training performance in previously unknown ways. Therefore, a learning-based client selection policy is preferred to a solely optimization-based policy.

In this paper, we investigate the client selection problem for HFL and propose a new learning-based policy, called Context-aware Online Client Selection (COCS). COCS is developed based on a novel Multi-Armed Bandit (MAB) framework called Contextual Combinatorial MAB (CC-MAB) [17], [18]. COCS is contextual because it allows clients to use their computational information (e.g., available computation resources), and the client-ES pairs transmission information (e.g., bandwidth and distance). COCS is combinatorial because NO selects a subset of client-ES pairs and attempts to maximize the training utilities (i.e., select as many as clients in each round) by optimizing the client selection decision. To the best of our knowledge, COCS policy is the first client selection decision for HFL. In summary, we highlight the contributions of this paper as follows:

1) We formulate a client selection problem for HFL, where NO needs to select clients to ESs clients to process the local training to make more clients received by ESs before deadline under limited budget. Client selection policy of HFL has a three-fold problem: (i) estimate the local model updates successfully received by ESs with cold-starts, (ii) decide whether a client should be selected to a certain ES due to time-varying connection conditions, and (iii) optimize how to pay computation resources on clients to maximize the utility under limited budgets.
2) Due to the a priori uncertain knowledge of participated clients, the client selection problem is formulated as a CC-MAB problem. An online learning policy COCS is developed, which leverages the contextual information such as downloading channel state and local computing time over aggregation round for making a decision. For the strongly convex HFL, we analyze the utility loss of COCS, termed regret, compared to the Oracle solution that knows the exacted information of participated clients. A sublinear regret bound is derived for the proposed COCS policy, which implies that COCS can produce asymptotically optimal client selection decisions for HFL.
3) For non-convex HFL, the utility function of the convergence speed is quadratically related to the number of participated clients. By assuming that the information of each client-ES pair is perfectly known by NO, we show that the client selection problem is a submodular maximization problem with M knapsack and one matroid constraints, where M is the number of ESs. We use the Fast Lazy Greedy (FLGreedy) algorithm [19] to approximate the optimal solution with a performance guarantee. To this end, the analysis shows that the COCS policy also achieves a sublinear regret.

The rest of this paper is organized as follows: Section 2 overviews the related works. The system model and client selection problem of HFL are presented in Section 3. We design the COCS policy for strongly convex HFL and provide an analytical performance in Section 4. Section 5 presents the COCS policy for non-convex HFL, which is applied by the approximated oracle solutions. Simulation results are shown in Section 6, followed by the conclusion in Section 7.

2 Related Work

Client selection can efficiently deal with the straggler problems and significantly improve the performance of FL in terms of convergence speed and training latency. For example, [15] designs a deep reinforcement learning algorithm (the local model updates and the global model are considered as states) to select clients. [20] uses gradient information to select clients. If the inner product between the client's local and global gradient is negative, it will be excluded. In [21], they develop a system model to estimate the total number of aggregation rounds and design a greedy algorithm to jointly optimize client selection and bandwidth allocation. Some biased client selection policies have been developed to improve the convergence results of conventional FL, [22] presents the biased client sampling can achieve communication and computation efficiency and [23] uses clustered client sampling to reduce the variance of local and global models. To improve the time-to-accuracy performance of training, [24] proposes an Oort algorithm that can guide the cloud server to select clients, and PyramidFL [25] fully exploits both data and system heterogeneity in a fine-grained manner. These mechanisms mainly focus on conventional FL, which differs from our scenario (client selection for HFL).

HFL has been considered to be a more practical FL framework for the current MEC system, since the hierarchical architecture makes FL communication more efficient and significantly reduces the impact of straggler [12]. Later, some studies improve the performance of HFL from different perspectives or use it in some other applications. For example, [13], [14] propose a detailed convergence analysis

of HFL, showing that the convergence speed of HFL achieves a linear speedup of conventional FL. Recently, FL has attracted the more interest, especially for ML on IoT devices. [26] designs a hierarchical blockchain framework for knowledge sharing on smart vehicles, which learns the environmental data through ML methods and share the knowledge with others. [27] uses HFL to better adapt to personalized modeling tasks and protect private information.

The MAB problem has been extensively studied to address the key trade-off between exploration and exploitation making under uncertain environment [28], and it has been used in FL for designing the client scheduling or selection [29], [30], [31]. For example, [17] considers an MAB problem for the edge service provisioning and [18] studies the optimal sniffer channel assignment for small cell cognitive radio networks. It has also been widely used in FL for designing the client scheduling or selection [29], [30], [31]. [29] designs a client scheduling problem and provides a MAB-based framework for FL training without knowing the wireless channel state information and the dynamic usage of local computing resources. In order to minimize the latency, [30] models fair-guaranteed client selection as a Lyapunov optimization problem and presents a policy based on CC-MAB to estimate the model transmission time. A multi-agent MAB algorithm is developed to minimize the FL training latency over wireless channels, constrained by training performance as well as each client's differential privacy requirement in [31]. In this paper, the COCS policy is proposed to select clients for HFL. In traditional FL, CS connects all clients and the available set of selecting clients does not change in each aggregation round. However, in HFL, due to the dynamic connection conditions of the client-ES pair and the limited available computing capacities of clients in each edge aggregation round, we cannot assume that each ES can make a selection decision for the same client set, which indicates that the COCS policy must face two constraints for deciding which clients can be selected and how to rent the computational resources. These two constraint can be divided into two different categories: knapsack and matroid constraints rather than single constraint MAB problems [17], [18], which brings more challenges.

3 System Model and Problem Formulation

3.1 Preliminary of Hierarchical Federated Learning

The Network Operator (NO) leverages a typical edge-cloud architecture to set a Federated Learning (FL) service, where it is named as Hierarchical FL (HFL) [12], [13], [14] in Fig. 1. Unlike the conventional FL [1], [2], [3] only including clients and a Cloud Server (CS), HFL consists of a set of mobile devices/clients, indexed by $\mathcal{N} = \{1, 2, \ldots, N\}$, a set of Edge Servers (ES), indexed by $\mathcal{M} = \{1, 2, \ldots, M\}$ and a Cloud Server (CS). Let $\mathcal{N}_m^t = \{1, 2, \ldots, N_m^t\}$ denote the set of clients, which can communicate with the ES m in edge aggregation round t. Note that the communication area of different edge server may be overlapped (i.e., $\sum_{m=1}^{M} N_m^t \geq N$). The client $n \in \mathcal{N}$ is able to communicate to a subset of ESs $\mathcal{C}_n^t \subseteq \mathcal{M}$ in round t. In particular, we assume that each client is equipped a single antenna such that it only communicates with one ES $m \in \mathcal{C}_n^t$ even it is located in the overlapped area in each round. Let $\boldsymbol{w}$ denote the parameters of the global model. The goal of the

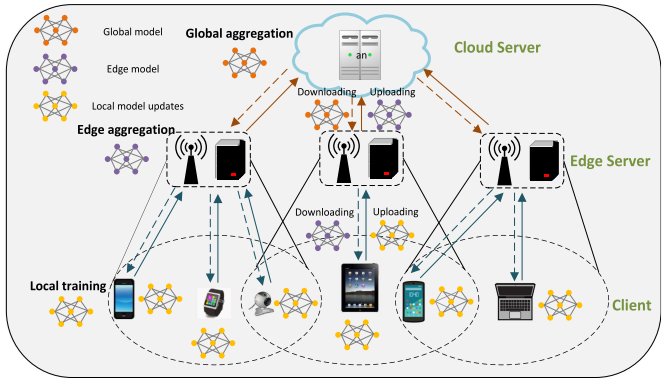


Fig. 1. The architecture of HFL.

FL service is to find the optimal parameters of global model $\boldsymbol{w}$, which minimizes the average loss function $f(\boldsymbol{w})$ under the HFL network as follows:

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) := \frac{1}{M} \sum_{m \in \mathcal{M}} \frac{1}{S_m} \sum_{n \in \boldsymbol{s}_m} F_n(\boldsymbol{w}), \tag{1}$$

where $\boldsymbol{s}_m$ is the selected client set by the ES m with the number S_m in each edge aggregation round, $F_n(\boldsymbol{w}) \triangleq \sum_{\xi_n \sim \mathcal{D}_n} \ell(\boldsymbol{w}; \xi_n)$ is the loss function associated with the local dataset $\mathcal{D}_n$ on client n, and $\ell(\boldsymbol{w}; \xi_n)$ is the loss of data sample ξ_n. The objective of the loss function $F_n(\cdot)$ can be convex (e.g., logistic regression) or non-convex (e.g., Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM)). The training steps of HFL can be summarized as follows:

(i) First, each ES m randomly selects a subset of clients $\boldsymbol{s}_m^t \subseteq \mathcal{N}_m^t$ in its coverage area. Even if a client is in the overlapping area, is is only allowed to communicate with one ES in one round. We assume that the HFL network contains a backhaul link to transmit the selected clients to avoid that some clients are selected on multiple ESs. Each client n selected by ES m downloads the edge model $\boldsymbol{w}_m^t$ and sets it to be the local model $\boldsymbol{w}_n^t = \boldsymbol{w}_m^t, \forall n \in \boldsymbol{s}_m^t$.

(ii) Then, each client n takes E epochs to update its own local model by Stochastic Gradient Descent (SGD) from its dataset $\mathcal{D}_n$ as follows:

$$\boldsymbol{w}_n^{t+e+1} = \boldsymbol{w}_n^{t+e} - \eta_t g(\boldsymbol{w}_n^{t+e}; \xi_n^{t+e}), \tag{2}$$

where $e = 0, 1, \ldots, E-1$, η_t is the learning rate, and $g(\boldsymbol{w}_n^{t+e}; \xi_n^{t+e})$ is the stochastic gradient of $F_n(\boldsymbol{w})$ (i.e., $\mathbb{E}_{\xi_n^{t+e} \sim \mathcal{D}_n}[g(\boldsymbol{w}_n^{t+e}; \xi_n^{t+e})] = \nabla F_n(\boldsymbol{w}_n^{t+e})$).

(iii) After E local training epochs, client $n \in \mathcal{S}_m^t$ uploads the local model updates $\Delta_n^t \triangleq \boldsymbol{w}_n^{t+E-1} - \boldsymbol{w}_n^t$ to the ES m. Instead of aggregating all local models on CS at the end of round t [1], [2], [3] of conventional FL, local model updates are averaged within ES m to be edge model $\boldsymbol{w}_m^{t+1}$, called *edge aggregation*, which is given as follows:

$$\boldsymbol{w}_m^{t+1} = \boldsymbol{w}_m^t + \frac{1}{S_m^t} \sum_{n \in \boldsymbol{s}_m^t} \Delta_n^t, \tag{3}$$

(iv) After T_{ES} rounds of edge aggregation, global model $\boldsymbol{w}^t$ is computed by $\boldsymbol{w}^t = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{w}_m^t$, $\forall t = \{T_{\mathrm{ES}}, 2T_{\mathrm{ES}}, \ldots\}$ from all M ESs, called *global aggregation*. Then, each ES m downloads the global model to be its edge model $\boldsymbol{w}_m^t = \boldsymbol{w}^t$.

Repeating the above four steps with a sufficiently large round T, NO will achieve the global model $\boldsymbol{w} = \boldsymbol{w}^T$ and stop the training process. HFL has been demonstrated that it achieves a linear speedup of convergence to conventional FL algorithms [13], [14].

3.2 Cost of Client Selection

Since clients usually do not belong to NO, clients are required to charge NO for the amount of requested computation resources for collecting dataset and processing the local training to achieve the learning goal. At the beginning of each edge aggregation round, each client reveals its available computation resources y_n^t to NO, which includes CPU frequency, RAM and storage, etc., in order to process the current local training updates.

Each client sets a price for its computation resources. Let $c_n(y_n^t)$ denote the price charged by client n, where $c_n(\cdot)$ is a non-decreasing mapping function related to the price for computation resources y_n^t. Due to the limited rental budget $\tilde{B}$ of NO, for any edge aggregation round t, the client selection decision NO must satisfy the budget constraint $\sum_{m\in\mathcal{M}}\sum_{n\in\boldsymbol{s}_m^t} c_n(y_n^t) \le \tilde{B}$.

3.3 Deadline Based HFL

In summary, an edge aggregation consists of four stages: *Download Transmission* (DT), *Local Computation* (LC), *Upload Transmission* (UT) and *Edge Computation* (EC).

In DT stage, the selected client $n \in \boldsymbol{s}_m^t$ downloads the current edge model from the ES m. Followed by Shannon's equation, the channel state of DT $c_{\mathrm{DT},n}^t$ is calculated by:

$$c_{\mathrm{DT},n}^t = \log_2(1 + P_n^t g_{\mathrm{DT},n}^t / N_0), \tag{4}$$

where P_n^t is the transmission power, $g_{\mathrm{DT},n}^t$ is the downlink wireless channel gain and N_0 is the noise power. Let a_{DT} denote the downloading data size (i.e., size of edge model $\boldsymbol{w}_m^t$) and the allocated bandwidth is b_n^t in the edge aggregation round t. Therefore, thus the DT time for client n is $\tau_{\mathrm{DT},n}^t = a_{\mathrm{DT}}/(b_n^t c_{\mathrm{DT},n}^t)$.

Once the client n receives $\boldsymbol{w}_m^t$, training comes to the LC stage (i.e., it updates the local model using its own dataset $\mathcal{D}_n$ according to (2)). The LC time of each client is determined by the local computation resources y_n^t in the current round t. Given the computation resources $y_n^t > 0$, the LC time can be obtained as $\tau_{\mathrm{LC},n}^t(y_n^t) = q/y_n^t$, where q is the computation workload, which is based on the complexity of learning model and data.

When the LC is finished, client n uploads its local model updates Δ_n^{t+1} to the ES m. Similar to the channel state definition of DT in (4), the channel state of UT is $c_{\mathrm{UT},n}^t = \log_2(1 + P_n^t g_{\mathrm{UT},n}^t/N_0)$ and UT time is $\tau_{\mathrm{UT},n}^t = a_{\mathrm{UT}}/(b_n^t c_{\mathrm{UT},n}^t)$, where $g_{\mathrm{UT},n}^t$ is the uplink channel gain and a_{UT} is the uploading data size (i.e., size of Δ_n^{t+1}).

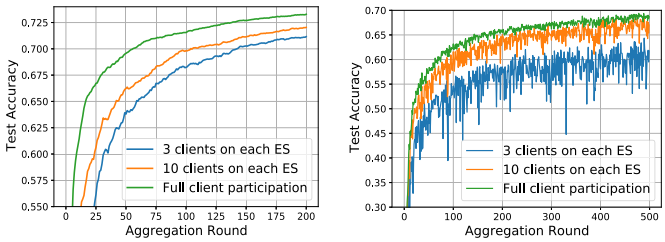
Finally, if the local model updates of all selected clients are received by ESs, the edge models should be computed in (3). The EC time is $\tau_{\mathrm{EC},m}^t = q_m/y_m$, where q_m is the edge model workload and y^m is the process capacity of the ES m. Note that $\tau_{\mathrm{EC},m}$ should be different for every ES m. However, since the EC stage only takes the average calculation of the received local model updates according to (3) and the capacity of ES q_m is always very large compared to clients', this does not waste much running time compared to the other three stages. The difference of $\tau_{\mathrm{EC},m}$ across all ESs is small and negligible, and hence NO does not need to consider the influence of $\tau_{\mathrm{EC},m}$. As such, the training time of client n is defined as follows:

$$\begin{aligned}\tau_n^t(y_n^t) &= \tau_{\mathrm{DT},n}^t + \tau_{\mathrm{LC},n}^t(y_n^t) + \tau_{\mathrm{UT},n}^t \\ &= \frac{a_{\mathrm{DT},n}^t}{b_n^t c_{\mathrm{DT},n}^t} + \frac{q}{y_n^t} + \frac{a_{\mathrm{UT},n}^t}{b_n^t c_{\mathrm{UT},n}^t}, \quad \forall n, t.\end{aligned} \tag{5}$$

Due to some physical limitations (e.g., low computation capability and unstable communication), some clients may incur huge training latency in one edge aggregation round. Therefore, the deadline-based FL [32], [33], [34] is more realistic to deal with straggler clients. Specifically, ESs drop the clients whose the local model updates cannot be received before the deadline $\tau_{\mathrm{dead},m}$ (i.e., client n such that $\tau_n^t > \tau_{\mathrm{dead},m}$). In this paper, we consider deadline-based HFL. Therefore, the edge aggregation can be reformulated:

$$\boldsymbol{w}_m^t = \begin{cases} \frac{1}{\sum_{n\in\boldsymbol{s}_m^t} X_n^t} \sum_{n\in\boldsymbol{s}_m^t} X_n^t \boldsymbol{w}_n^t, & \text{if } \sum_{n\in\boldsymbol{s}_m^t} X_n^t \ge Z \\ \frac{1}{Z}\sum_{\tau_n^t \le \tau^Z} \boldsymbol{w}_n^t, & \text{else} \end{cases} \tag{6}$$

where X_n^t is a binary random variable representing whether client n's model update can be received before the deadline (i.e., if $\tau_n^t \le \tau_{\mathrm{dead},m}$, $X_n^t = 1$; otherwise, $X_n^t = 0$), and τ^Z is the training time of the Zth fastest client. In Fig. 2, we can see that if each ES only receives 3 local model updates, the training performance has large degradation and variance. In order to guarantee a minimum level of training performance, we require that at least Z local model updates must be received for edge aggregation. Therefore, in case less than Z clients' updates are received before the deadline, the system has to wait for some additional time $\tau^Z - \tau_{\mathrm{dead},m}$. For practical values of $\tau^Z - \tau_{\mathrm{dead},m}$, the probability of having less than Z client updates received before the deadline is small. For analysis convenience, we assume that at least Z client updates can be received before the deadline in every edge aggregation round. In addition, we assume that the deadline of all ESs are set the same, $\tau_{\mathrm{dead},m} = \tau_{\mathrm{dead}}, \forall m$. The extension to heterogeneous deadlines is straightforward.



(a) MNIST dataset under logistic regression.

(b) CIFAR-10 dataset under CNN.

Fig. 2. HFL training performance with different number of participating clients in each edge aggregation round.

3.4 Utility Function of Client Selection of HFL

Some existing HFL studies [13], [14], [35] have demonstrated that the convergence speed depends on the number of participating clients in each edge aggregation round for both strongly convex and non-convex HFL (i.e., the more

clients participated, the faster convergence speed). In order to support the theoretical results, we show the training performance on our simulated HFL network with $M = 3$ and $N = 80$ in Fig. 2, and it is observed that more participating clients on ESs can improve the performance in both the strongly convex and non-convex HFL settings.

For now, we consider strongly strongly convex HFL, where the convergence speed is linearly dependent on the number of participating clients. The client selection policy for non-convex HFL will be developed in Section 5. As in [32], [33], [34], not all the selected clients in $\boldsymbol{s}_m^t$ may reach the EC stage (i.e., $\sum_{n\in\boldsymbol{s}_m^t} X_n^t \leq S_m^t$) due to straggler drop-out. To achieve a targeted convergence criteria, NO thus needs to run more FL rounds, thereby incurring a higher training cost. Therefore, it is necessary to develop an efficient client selection policy to improve the convergence speed for HFL, where more clients can participate in every round without dropping out. Let $\boldsymbol{X}_m^t = \{X_{n,m}^t\}_{\forall n\in\mathcal{N}_m^t}$, then the utility of the client selection decision on ES m is defined as:

$$\mu(\boldsymbol{s}_m^t; \boldsymbol{X}_m^t) = \sum_{n\in\boldsymbol{s}_m^t} X_{n,m}^t. \tag{7}$$

Further, let $\boldsymbol{s}^t = \{\boldsymbol{s}_1^t, \boldsymbol{s}_2^t, \ldots, \boldsymbol{s}_M^t\}$ denote the client selection decision of the overall system and $\boldsymbol{X}^t = \{\boldsymbol{X}_1^t, \boldsymbol{X}_2^t, \ldots, \boldsymbol{X}_M^t\}$. Therefore, the utility function of the whole HFL network is defined as:

$$\mu(\boldsymbol{s}^t; \boldsymbol{X}^t) = \frac{1}{M}\sum_{n\in\mathcal{M}}\sum_{n\in\boldsymbol{s}_m^t} X_{n,m}^t. \tag{8}$$

3.5 Client Selection Problem Formulation

The client selection problem for NO is a sequential decision-making problem. The goal of NO is to make selection decision $\boldsymbol{s}^t, \forall t$ to maximize the cumulative utility for a total of T aggregation rounds. If an ES selects very few clients, its training performance may be degraded and the computation resources of computation resources of ESs may be wasted. Since $\mathcal{N}_m^t$ is time variant in each edge aggregation due to the clients movement, we assume that the location of client n is uniformly distributed in the HFL network. Moreover, as T_{ES} is usually set larger than 1, NO cannot allocate the total budget $\tilde{B}$ to M ESs in each edge aggregation round. Therefore, we consider that NO equally divides the budget among the ESs (i.e., for each m, its budget is $B = \tilde{B}/M$). Assuming that NO knows a priori whether a selected client can return its model updates to the corresponding edge server in time, namely $\boldsymbol{X}^t, \forall t$, then the client selection problem is formulated:

$$\mathbf{P1:}\quad \max_{\{\boldsymbol{s}^t\}_{t=1}^T} \quad \sum_{t=1}^{T} \mu(\boldsymbol{s}^t; \hat{\boldsymbol{X}}^t) \tag{9a}$$

$$\text{s.t.} \quad \sum_{n\in\boldsymbol{s}_m^t} c_n(y_n^t) \leq B, \quad \forall m \in \mathcal{M} \tag{9b}$$

$$\boldsymbol{s}_m^t \subseteq \mathcal{N}_m^t, \quad \forall m, t \tag{9c}$$

$$\boldsymbol{s}_m^t \cap \boldsymbol{s}_{m'}^t = \emptyset, \quad m, m' \in \mathcal{M}, \forall t. \tag{9d}$$

The following challenges should be addressed to solve the client selection problem in HFL networks: (i) For maximizing the expected training utility of HFL, it is necessary to precisely estimate the selected clients in each edge aggregation round. In addition, since NO does not have enough experience to determine the selected clients at the first several rounds (i.e., cold start), collecting the historical data for estimation is important for this policy. (ii) With the successful participated clients estimation, how to optimize the selection decision on each ES under the limited budget should be carefully considered, because high variance of number of participated clients on each ES degrades the training performance. Therefore, we equally separate the total budget to each ES (constraint (9b)). (iii) Due to the movement of each client and the overlapping areas across all ESs, the available connecting ESs can be considered as time-varying (constraint (9c)), we must decide the client-ES pairs especially for the clients in overlapping areas, which brings more difficulties to make an efficient client selection decision. Note that constraint (9d) can guarantee that each client only can be selected to communicate with at most one ES. (iv) Since the selection decisions are based on the estimated participated clients $\hat{\boldsymbol{X}}^t$, the accuracy of participated clients estimation will directly influence the training utility of NO. After finishing the EC stage, ESs can process the client selection for next round (i.e., processing in DT, LC, and UT stages). The advantage is that if the client selection can be finished before τ^{dead}, it does not waste extra training time. The following section will propose an policy based on the Multi-Armed Bandits (MAB) in order to address the mentioned challenges.

4 Context-Aware Online Client Selection Policy for Strongly Convex HFL

In this section, we formulate our client selection problem of HFL as a Contextual Combinatorial Multi-Armed Bandit (CC-MAB). The combinatorial property is because NO pays computation resources from multiple clients for maximizing the training utility. The contextual property is because NO leverages contexts associated with clients to infer their participated probabilities. In this paper, whether successfully participating in the corresponding ES depends on many side factors, which are collaboratively referred to as *context*. We use contextual information to help infer the number of participated clients.

In CC-MAB, NO observes the context of clients at the beginning of each edge aggregation round before making the client selection decision. Recall that the participated probability of a client-ES pair depends on $r_{\mathrm{DT},n}^t, y_n^t$ and $r_{\mathrm{UT},n}^t$ in (5). At each edge aggregation round t, ES can measure the the channel state $c_{\mathrm{DT},n}^t$ by inferring the received signal strength of the received local model updates [36], [37]. Based on $c_{\mathrm{DT},n}^t$ and bandwidth b_n^t, ESs can compute the DT rate $r_{\mathrm{DT},n}^t$. Since the movement speed of clients is slower than the transmission speed of wireless signals, while NO cannot know the UT rate $r_{\mathrm{UT},n}^t$, it is not difficult to be inferred by $r_{\mathrm{DT},n}^t$ (suppose that clients do not locate in the same area in each edge aggregation round). Therefore, we set $c_{\mathrm{DT},n}^t$ as *context* and use the information to help significantly improve the participated probabilities of client-ES pairs.

Let $\boldsymbol{\phi}_{n,m}^t \in \Phi$ be the context of client-ES pair (n, m) in edge aggregation round t. Without loss of generality, we normalize $\boldsymbol{\phi}_{n,m}^t$ in a bounded space $\Phi = [0, 1]^2$ using min-max feature scaling. Let $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{n,m}^t\}_{\forall m,n\in\mathcal{N}_m^t}$ denote the context of all clients on ESs. The context of all clients on ESs are collected in $\boldsymbol{\phi}^t = \{\boldsymbol{\phi}_{n,m}^t\}_{\forall m,n\in\mathcal{N}_m^t}$. Whether successful participation of client n on ES m is a random variable parameterized by the context $\boldsymbol{\phi}_{n,m}^t$.

We simplify the notation of selected clients and define the context-aware $X^t_{n,m}(\boldsymbol{\phi}^t_{n,m})$. Specifically, $X_{n,m}$ is a mapping function for each client-ES pair (n,m), since the training time of clients is usually location-dependent (e.g., the distance between clients and ES, communication environment, and other processing tasks on a client). We further define $p_{n,m}(\boldsymbol{\phi}^t_{n,m}) \triangleq \mathbb{E}[X_{n,m}(\boldsymbol{\phi}^t_{n,m})]$ as the expected value (i.e., the participated probability $X^t_{n,m} \sim \text{Bernoulli}(p^t_{n,m})$) of $X_{n,m}(\boldsymbol{\phi}^t_{n,m})$.

4.1 Oracle Solution and Regret

Similar to the existing CC-MAB studies [17], [18], before providing our policy design, we first give an Oracle benchmark solution to the client selection problem of HFL by assuming that the NO knows the context-aware successful participated probability $p^t_{n,m}(\boldsymbol{\phi}^t_{n,m})$, $\forall m, n \in \mathcal{N}^t_m$. In this ideal setting, the utility function $\mu(\boldsymbol{s}^t; \boldsymbol{p}^t)$ is perfectly known by NO, and thus we can get the optimal value of the client selection problem. The long-term selection problem **P1** can be decomposed into T independent subproblems in each edge aggregation round:

$$\textbf{P2:} \quad \max_{\boldsymbol{s}^t} \quad \mu(\boldsymbol{s}^t; \boldsymbol{p}^t) \tag{10a}$$

$$\text{s.t.} \quad \sum_{n \in \boldsymbol{s}^t_m} c_n(y^t_n) \le B, \quad \forall m \in \mathcal{M} \tag{10b}$$

$$\boldsymbol{s}^t_m \subseteq \mathcal{N}^t_m, \quad \forall m, t \tag{10c}$$

$$\boldsymbol{s}^t_m \cap \boldsymbol{s}^t_{m'} = \emptyset, \quad m, m' \in \mathcal{M}, \forall t. \tag{10d}$$

P2 is a combinatorial optimization problem with M *Knapsack* and a *Matroid* constraints. The combinatorial property is because NO should choose a proper client selection decision to optimize participated probabilities on all ESs in order to achieve higher convergence speed. Knapsack constraints are from the constraint (10b), which bounds computation resources payment on each ES.

To prove that (10c) is a matroid constraint, we first state the definition of matroid. A matroid $\mathcal{E} = ((X), (I))$ is a system with independents sets, in which $\mathcal{X}$ is a finite set (named the ground set) and $\mathcal{I}$ represents the set of independent subsets of $\mathcal{X}$. It has the three properties: (1) $\emptyset \in \mathcal{I}$ and $\mathcal{X}$ has at least one subset of $\mathcal{X}$; (2) For each $A \subset B \subset \mathcal{X}$, if $A \in \mathcal{I}$, then $B \in \mathcal{I}$; (3) If $A, B \in \mathcal{I}$, and $|A| < |B|$, then $\exists \in B \setminus A$ such that $A\{x\} \in \mathcal{I}$. In the subproblem **P2**, let $\mathcal{X} = \cup_{m \in \mathcal{M}} \mathcal{N}^t_m$ denote the ground set of matroid $\mathcal{E} = (\mathcal{X}, \mathcal{I})$, and $\mathcal{I} = \{I_1, I_2, \ldots\}$ consists of subsets of $\mathcal{X}$ (i.e., $I_1 \subseteq \mathcal{X}, I_2 \subseteq \mathcal{X}, \ldots$), where all $I \in \mathcal{I}$ includes at most one client from $\mathcal{N}^t_m$ for each $m \in \mathcal{M}$. We can write I as $I = \cup_{m \in \mathcal{M}} \boldsymbol{s}^t_m$, s.t. $\boldsymbol{s}^t_m \in \mathcal{N}^t_m, \forall m$. In this paper, $\mathcal{I}$ is the set of all feasible client selection decisions. Therefore, it can be verified that (10c) is a matroid constraint [18].

Based on our analysis, it is easy to observe that **P2** is NP-hard, and hence it can be solved by *brute-force*, if the size of the HFL network is moderate. If the HFL network is too large, NO can use some commercial software to obtain the optimal solution (e.g., CPLEX [38]). For simplicity, we define the optimal Oracle solution for each **P2** in edge aggregation round t is $\boldsymbol{s}^{\text{opt},t}$. However, in practice, obtaining the prior knowledge of participated clients is infeasible, thus NO has to make a selection decision $\boldsymbol{s}^t$ based on the estimated participated clients $\hat{\boldsymbol{X}}^t$ in each edge aggregation round. Intuitively, NO should design an online client selection policy to choose $\boldsymbol{s}^t$ based on the estimation $\hat{\boldsymbol{X}}^t$. The performance of an online client selection policy is calculated by utility loss compared with the Oracle solution, called *regret*. Suppose that we have a selection sequence $\{\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^T\}$ given by a policy, the expected regret is

$$\mathbb{E}[R(T)] = \sum\nolimits_{t=1}^{T} (\mathbb{E}[\mu(\boldsymbol{s}^{\text{opt},t}; \boldsymbol{X}^t)] - \mathbb{E}[\mu(\boldsymbol{s}^t; \boldsymbol{X}^t)]). \tag{11}$$

The expectation is concerning with respect to the decisions made by the client selection decision policy and the participated clients over contexts.

Algorithm 1. Context-Aware Online Client Selection (COCS)

Input: T, h_T, $K(t)$.
Initialization: Create partition $\mathcal{L}_T$ on context space Φ; set $C_{n,m}(l) = 0, \mathcal{E}_{n,m}(l) = \emptyset\ \hat{\beta}_{n,m}(l) = 0, \forall n \in \mathcal{N}\ \forall m \in \mathcal{M}, \forall l \in \mathcal{L}_T$.

```
for t = 1, ..., T do
  Observes the context of clients on ESs φ^t;
  Determine C_n^{ue,t} and N_n^{ue,t}, and estimate the participated clients X̂^t based on the contexts φ^t;
  if N_n^{ue,t} ≠ ∅ then                                   ▷Exploration
    if N_n^{ue,t} ≠ ∅, ∀n then
      Determine s^t by solving (14);
    else
      Get s̃^t by solving (15);
      Select the clients in N^{ed,t} by solving (17) and determine the client selection decision s^t;
    end if
  else                                                     ▷Exploitation
    Determine the client selection decision s^t by solving (18);
  end if
  for n ∈ N do                                             ▷Update
      Identify each ES m successfully receives the client n before τ^dead and context hypercube l that belongs to φ^t_{n,S_m};
      Observe whether the selected client n successfully participates on ES m as X_{n,m};
      Update estimations: p̂_{n,m}(l) = (p̂_{n,m}(l) C_{n,m}(l) + X) / (C_{n,m}(l) + 1);
      Update counters: C_{n,m}(l) = C_{n,m}(l) + 1;
  end for
end for
```

4.2 Context-Aware Online Client Selection Policy

Now, we will present our online client selection decision policy name Context-aware Online Client Selection (COCS). The COCS policy is designed based on CC-MAB. In edge aggregation round t, the process of COCS of NO is operated sequentially as follows: (i) NO observes the contexts of all client-ES pairs $\boldsymbol{\phi}^t = \{\phi^t_{n,m}\}_{n,m \in \mathcal{N}^t_m}, \phi^t_{n,m} \in \Phi$. (ii) NO determines its selection decision $\boldsymbol{s}^t$ based on the observed context information $\boldsymbol{\phi}^t$ in the current round t and the knowledge learned from the previous $t-1$ rounds. (iii) The selection decision $\boldsymbol{s}^t$ is applied. If $s^t_n / = 0, \forall n \in \boldsymbol{s}^t_m$, the clients located in the coverage area of ES m can be selected by ES m for training in round t. (iv) At the end of each edge aggregation round, the local model updates Δ^t_n from which clients are observed by all ESs, which is then used to update the estimated participated clients $\hat{X}_{n,m}(\phi^t_{n,m})$ from the observed context $\phi^t_{n,m}$ of client-ES pair (n,m). Note that in this paper, we consider how to properly allocate the computational resources to improve the training performance in FL.

Instead of computational resources, our policy can easily extend to other scenario such as spectrum resources [39], [40].

The pseudocode of COCS policy is presented in Algorithm 1. It has two parameters $K(t)$ and h_T to be designed, where $K(t)$ is a deterministic and monotonically increasing function used to identify the under-explored context, and h_T decides how we partition the context space. The COCS policy is stated as follows:

Initialization Phase. Given parameter h_T, the proposed policy first creates a partition denoted by $\mathcal{L}_T$ for the context space $\Phi = [0,1]^2$, which splits Φ into $(h_T)^2$ sets. Each set is a 2-dimensional hypercube with size $\frac{1}{h_T} \times \cdots \times \frac{1}{h_T}$. Note that h_T is an important input parameter to guarantee policy performance. For each hypercube $l \in \mathcal{L}_T$, the NO keeps a counter $C^t_{n,m}(l)$ for each client $n \in \mathcal{N}$ and each ES $m \in \mathcal{M}$. For the tuple (n,m,l) of a counter $C^t_{n,m}(l)$ for each client-ES pair (n,m), we define a selection event $V_{n,m,l}$ that represents a selection decision satisfying the following three conditions: 1) the client $n \in \mathcal{N}^t_m$ is selected to an ES m; 2) the ES m successfully receives the client n before τ_{dead} (i.e., $\tau^t_n \le \tau_{\text{dead}}$); 3) the context of client-ES pair (n,m) belongs to l (i.e., $\phi^t_{n,m} \in l$). The counter $C^t_{n,m}(l)$ stores the number of times that the event $V_{n,m,l}$ occurs until edge aggregation round t. Each ES m also saves an experience $\mathcal{E}^t_{n,m}(l)$ for each client n and each hypercube l, which contains the observed participated clients indicators when a selection event $V_{n,m,l}$ occurs. The experience $\mathcal{E}^t_{n,m}(l)$ is useful for making the future decision whether coming into exploration or exploitation phase. Based on the observed participation indicators in $\mathcal{E}^t_{n,m}(l)$, the estimated participated probability for a selection event $V_{n,m,l}$ is computed by:

$$\hat{p}^t_{n,m}(l) = \frac{1}{C^t_{n,m}(l)} \sum_{X \in \mathcal{E}^t_{n,m}(l)} X. \tag{12}$$

In each edge aggregation round t, the COCS policy has the following phases:

Hypercube Identification Phase. If the local model updates Δ^t_n of client $n \in \mathcal{N}^t_m$ can be successfully received by an ES m in edge aggregation round t, we obtain that $l^t_{n,m}$ is the hypercube for the context $\phi^t_{n,m}$, the estimated participated probability of client n on ES m is $\hat{X}^t_{n,m} = \hat{p}^t_{n,m}(l^t_{n,m})$. Let $\hat{\boldsymbol{X}}^t = \{\hat{X}^t_{n,m}\}_{\forall m, n \in \mathcal{N}^t_m}$ denote the collection of all the estimated participated probabilities. For making a client selection decision, COCS policy needs to check whether these hypercubes have been explored sufficiently in order to ensure the enough accuracy of the estimated participated probability for each client-ES pair (n,m). Therefore, we define *under-explored* hypercubes $\mathcal{L}^{\text{ue}}_m(\boldsymbol{\phi}^t)$ for the ES m in edge aggregation round t as follows:

$$\mathcal{L}^{\text{ue},t}_m \triangleq \left\{ l \in \mathcal{L}_T \middle| \begin{array}{l} \exists \boldsymbol{\phi}^t_{n,m} \in \boldsymbol{\phi}^t, \phi^t_{n,m} \in l, \tau^t_n \le \tau_{\text{dead}} \\ \text{and } C^t_{n,m}(l) \le K(t) \end{array} \right\}. \tag{13}$$

Also, let $\mathcal{N}^{\text{ue},t}_m(\boldsymbol{\phi}^t) \triangleq \{n \in \mathcal{N}^t_m | l^t_{n,m} \in \mathcal{L}^{\text{ue},t}_m(\boldsymbol{\phi}^t)\}$ denote the collection of the under-explored client n for each ES m. The challenge of COCS policy is how to decide the current estimated participated clients are accurate enough to guide the client selection decision in each edge aggregation round, which is referred as *exploitation* or more training results need to be collected for a certain hypercube, which is referred to as *exploration*. COCS policy aims to balance the exploration and exploitation phases in order to maximize the utility of NO up to a finite round T. Based on the $\mathcal{N}^{\text{ue},t}_m(\boldsymbol{\phi})^t$, COCS can identify that then either enters an exploration phase or an exploitation phase.

Exploration Phase. First, let $\mathcal{N}^{\text{ue},t}_m(\boldsymbol{\phi}^t) \triangleq \{n \in \mathcal{N}^t_m | \mathcal{N}^{\text{ue},t}_m \neq \emptyset\}$ denote an ES m has under-explored clients, and $\mathcal{N}^{\text{ed},t}_m(\boldsymbol{\phi}^t) \triangleq \mathcal{N} \setminus \mathcal{N}^{\text{ue},t}(\boldsymbol{\phi}^t)$ denote the ES m does not have under-explored clients. If the ES m has a non-empty $\mathcal{N}^{\text{ue},t}_m$, then COCS enters the exploration phase. We may have two cases in exploration phase:

(i) All the clients have under-explored ESs. NO hopes to receive more local training updates Δ^t_n. Thus, COCS policy aims to select as many clients that have under-explored ESs sequentially solved by the following optimization:

$$\max_{\boldsymbol{s}^t} \ |\boldsymbol{s}^t| \quad \text{s.t. (10b), (10c), (10d),} \tag{14}$$

where $|\boldsymbol{s}^t|$ is the size of the collection $\boldsymbol{s}^t = \{\boldsymbol{s}^t_1, \boldsymbol{s}^t_2, \ldots, \boldsymbol{s}^t_M\}$.

(ii) Part of ESs have under-explored clients $\exists \mathcal{N}^{\text{ue},t}_m \neq \emptyset$. We divide this case into two stages: NO first selects ESs that have under-explored clients $m \in \mathcal{N}^{\text{ue},t}_m$ by solving the following optimization:

$$\max_{\tilde{\boldsymbol{s}}^t} \quad |\tilde{\boldsymbol{s}}^t| \tag{15a}$$

$$\text{s.t.} \quad \sum_{n \in \tilde{\boldsymbol{s}}^t_m} c_n(y^t_n) \le B, \quad \forall n \in \mathcal{N}^{\text{ue},t}_m, \forall m \in \mathcal{M} \tag{15b}$$

$$s^t_m \in \mathcal{N}^t_m \cup \{null\}, \quad \forall n \in \mathcal{N}^{\text{ue},t}_m \tag{15c}$$

$$\tilde{\boldsymbol{s}}^t_m \cap \tilde{\boldsymbol{s}}^t_{m'} = \emptyset, \quad m, m' \in \mathcal{M}, \forall t. \tag{15d}$$

where $\tilde{\boldsymbol{s}}^t$ is client selection decision on ES m that has under-explored clients and $|\tilde{\boldsymbol{s}}^t|$ is the size of the collection $\tilde{\boldsymbol{s}}^t = \{\tilde{\boldsymbol{s}}^t_1, \tilde{\boldsymbol{s}}^t_2, \ldots, \tilde{\boldsymbol{s}}^t_M\}$. Second, ESs aim to select the explored clients $\forall n \in \mathcal{N}^{\text{ed},t}_m$. Here, we assume that there exists ESs that $B - \sum_{n \in \tilde{\boldsymbol{s}}^t_m} c_n(y^t_n) \ge c^{\min,t}, m \in \mathcal{M}$, where $c^{\min,t} = \min_{n \in \mathcal{N}^{\text{ed},t}_m} c_n(y^t_n), \forall m$. Therefore, ESs can select the clients $n \in \mathcal{N}^{\text{ed},t}_m$ with the following constraint:

$$\sum_{n \in \boldsymbol{s}^t_m \setminus \tilde{\boldsymbol{s}}^t_m} c_n(y^t_n) \le B - \sum_{n \in \tilde{\boldsymbol{s}}^t_m} c_n(y^t_n), \ \forall n \in \mathcal{N}^{\text{ed},t}_m. \tag{16}$$

If not, NO does not need to select clients in $\mathcal{N}^{\text{ed},t}_m$ due to no budget left. Under this condition, the client selection decisions are jointly optimized the following optimization:

$$\max_{\boldsymbol{s}^t} \ \mu(\boldsymbol{s}^t; \hat{\boldsymbol{X}}^t) \tag{17a}$$

$$\text{s.t. } B - \sum_{n \in \tilde{\boldsymbol{s}}^t_m} c_n(y^t_n) \ge c^{\min,t}, \quad m \in \mathcal{M}, \tag{17b}$$

$$\text{(15b), (15c), (16).} \tag{17c}$$

Exploitation Phase. If the set of under-explored clients is empty (i.e., $\mathcal{N}^{\text{ue},t}_m = \emptyset, \forall m$), then COCS policy enters the exploitation phase. The optimal client selection decision $\boldsymbol{s}^t$ is derived by solving **P2** from the current estimated participated clients $\hat{\boldsymbol{X}}^t$:

$$\max_{\boldsymbol{s}^t} \ \mu(\boldsymbol{s}^t; \hat{\boldsymbol{X}}^t) \quad \text{s.t. (10b), (10c).} \tag{18}$$

Update Phase. After selecting the client-ES pair in each round t, the proposed COCS policy observes whether the local model updates of selected clients can be received before the deadline τ^{dead}; then, it updates $\hat{p}^t_{n,m}(l)$ and $C^t_{n,m}(l)$ of each hypercube $l \in \mathcal{L}_T$.

4.3 Performance Analysis

To present an upper performance bound of the COCS policy in terms of *regret*, we make the following assumption that the participated clients are similar when their contexts are similar. This natural assumption is formalized by the following Hölder condition [17], [18], which is defined as follows:

Assumption 1. *(Hölder Condition). If a real function f on D-dimensional euclidean space satisfies Hödel condition, there exists $L > 0, \alpha > 0$ such that for any $\boldsymbol{\phi}, \boldsymbol{\phi}' \in \Phi$, it holds that $|f_n(\boldsymbol{\phi}) - f_n(\boldsymbol{\phi}')| \leq L\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|^{\alpha}$ for an arbitrary client $n \in \mathcal{N}$, where $\|\cdot\|$ is the euclidean norm.*

By providing the design of the input parameters $K(t)$ and h_T, we show that COCS policy achieves a sublinear $R(T) = O(T^{\gamma})$ with $\gamma < 1$, which guarantees that COCS has an asymptotically optimal performance. This means that the online client selection decision via COCS policy converges to the Oracle solution. Because any edge aggregation round is either in the exploration or exploitation phase, the regret can be divided into two parts $R(T) = R_{\text{explore}}(T) + R_{\text{exploit}}(T)$, where $R_{\text{explore}}(T)$ and $R_{\text{exploit}}(T)$ are the regrets due to exploration and exploitation phases, respectively. The total regret bound is achieved by separately bounding these two parts. Therefore, we present two lemmas for bounding exploration and exploitation regrets.

Lemma 1. *(Bound of $\mathbb{E}[R_{\text{explore}}(T)]$.) Given the input parameters $K(t) = t^z \log(t)$ and $h_T = \lceil T^{\gamma} \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{2}$, the regret $\mathbb{E}[E_{\text{explore}}(T)]$ is bounded by:*

$$\mathbb{E}[E_{\text{explore}}(T)] \leq \frac{4N^2MB}{c^{\min}}(T^{z+2\gamma}\log(T) + T^{2\gamma}),$$

where $c^{\min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$.

Proof. See in online Appendix A, available in the online supplemental material [41]. □

Lemma 1 shows that the order of $R_{\text{explore}}(T)$ is determined by the control function $K(T)$ and the number of hypercubes $(h_T)^D$ in partition $\mathcal{L}_T$.

Lemma 2. *(Bound of $\mathbb{E}[R_{\text{exploit}}(T)]$.) Given $K(t) = t^z \log(t)$ and $h_T = \lceil T^{\gamma} \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{2}$, if the Hölder condition holds true and the additional condition $2H(t) + \frac{2NMB}{c^{\min}} L 2^{\frac{\alpha}{2}} h_T^{\alpha} \leq At^{\theta}$ is satisfied with $H(t) > \frac{NMB}{c^{\min}} t^{-\frac{z}{2}}$, $A > 0$, $\theta < 0$, for all t, then $\mathbb{E}[R_{\text{exploit}}(T)]$ is bounded by:*

$$\mathbb{E}[R_{\text{exploit}}(T)] \leq \frac{NMB}{c^{\min}}\left(\sum\nolimits_{k=1}^{B/c^{\min}} \binom{N}{k}\right)\frac{\pi^2}{3} + \frac{3NMB}{c^{\min}} L 2^{\frac{\alpha}{2}} T^{1-\gamma\alpha} + \frac{A}{1+\theta} T^{1+\theta},$$

where $c^{\min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$.

Proof. See in online Appendix B, available in the online supplemental material [41]. □

Lemma 2 indicates that the regret of exploitation $\mathbb{E}[R_{\text{exploitation}}(T)]$ depends on the choice of z and γ with an additional condition being satisfied. Based on the above two Lemmas, we will have the following Theorem for the upper bound of the regret $\mathbb{E}[R(T)]$.

Theorem 1. *(Bound of $\mathbb{E}[R(T)]$.) Given the input parameters $K(t) = t^z \log(t)$ and $h_T = \lceil T^{\gamma} \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{2}$, if the Hölder condition holds true and the additional condition $2H(t) + \frac{2NMB}{c^{\min}} L 2^{\frac{\alpha}{2}} h_T^{\alpha} \leq At^{\theta}$ is satisfied with $H(t) > \frac{NMB}{c^{\min}} t^{-\frac{z}{2}}$, $A > 0$, $\theta < 0$, for all t, then the regret $\mathbb{E}[R(T)]$ can be bounded by:*

$$\mathbb{E}[R(T)] \leq \frac{4N^2MB}{c^{\min}}(T^{z+2\gamma}\log(T) + T^{2\gamma}) + \frac{NMB}{c^{\min}}\left(\sum\nolimits_{k=1}^{B/c^{\min}} \binom{N}{k}\right)\frac{\pi^2}{3} + \frac{3NMB}{c^{\min}} L 2^{\frac{\alpha}{2}} T^{1-\gamma\alpha} + \frac{A}{1+\theta} T^{1+\theta},$$

where $c^{\min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$.

Proof. See in online Appendix C, available in the online supplemental material [41]. □

The regret upper bound in Theorem 1 is given with properly choosing input parameters $K(t)$ and h_t. However, the values of z, γ, A and θ are not deterministic. Next, we will show that the regret upper bound of $\mathbb{E}[R(T)]$ in these parameters design.

Theorem 2. *(Regret upper bound). If we select $z = \frac{2\alpha}{3\alpha+2} \in (0,1)$, $\gamma = \frac{z}{2\alpha}$, $\theta = -\frac{z}{2}$, $A = \frac{2NMB}{c^{\min}} t^{-\frac{z}{2}} + \frac{2NMB}{c^{\min}} L 2^{\frac{\alpha}{2}}$, and COCS algorithm runs with these parameters, the regret $\mathbb{E}[R(T)]$ can be bounded by:*

$$\mathbb{E}[R(T)] \leq \frac{4N^2MB}{c^{\min}}(\log(T) T^{\frac{2\alpha+2}{3\alpha+2}} + T^{\frac{2}{3\alpha+2}}) + \frac{NMB}{c^{\min}}\left(\sum\nolimits_{k=1}^{B/c^{\min}} \binom{N}{k}\right)\frac{\pi^2}{3} + \left(3L2^{\frac{\alpha}{2}} + \frac{2 + L2^{\frac{\alpha}{2}}}{(2\alpha+2)/(3\alpha+2)}\right)\frac{NMB}{c^{\min}} T^{\frac{2\alpha+2}{3\alpha+2}},$$

where $c^{\min} = \min_{y_n^t, \forall n,t} c_n(y_n^t)$. The dominant order of the regret $\mathbb{E}[R(T)]$ is $O(\frac{4N^2MB}{c^{\min}} T^{\frac{2\alpha+2}{3\alpha+2}} \log(T))$.

Proof. See in online Appendix C, available in the online supplemental material [41]. □

The dominant order of regret upper bound indicates the COCS policy in Theorem 2 is sublinear. In addition, the regret bound is valid for any total rounds T, and it can be used to characterize the convergence speed of HFL.

4.4 Complexity Analysis

The space complexity of COCS policy is determined by the number of counters $C_{n,m}^t(l)$ and experiences $\mathcal{E}_{n,m}^t(l)$ maintained for hypercubes. Because the counter is an integer for each hypercube, the space complexity is determined by the number of hypercubes. The experience $\mathcal{E}_{n,m}^t(l)$ is a set of observed successfully participating clients records up to round t, which requires a higher memory. However, it is unnecessary to store all historical records, since most estimators can be updated recursively. Therefore, the NO only needs to keep the current participated clients estimation for a hypercube. If COCS is run with the parameters in Theorem 2, the number of hypercubes is $(h_T)^2 = \lceil T^{\frac{1}{3\alpha+2}} \rceil^2$, and thus the required space is sublinear in total rounds T. This

means that when $T \to \infty$, COCS will require infinite memory. In the practical implementations, NO only needs to keep the counters and experiences of hypercubes to which at least one of the observed contexts occurs. Therefore, the practical space requirement of some counters and experiences is much smaller than the theoretical requirement.

5 COCS Policy for Non-Convex HFL

In this section, we will discuss the solutions for the client selection problems for non-convex HFL (e.g., neural network), where the convergence speed is quadratically related to the number of participated clients in each edge aggregation round [13], [14]. Similar to the strongly convex HFL in (7), the utility function of non-convex HFL is

$$\mu_{\text{non}}(\boldsymbol{s}^t; \boldsymbol{X}^t) = \sqrt{\frac{1}{M} \sum\nolimits_{m \in \mathcal{M}} \sum\nolimits_{n \in \boldsymbol{s}_m^t} X_{n,m}^t}. \tag{19}$$

Therefore, the client selection problem of non-convex HFL in edge aggregation round t is formulated as follows:

$$\textbf{P3}: \quad \max\nolimits_{\boldsymbol{s}^t} \quad \mu_{\text{non}}(\boldsymbol{s}^t; \boldsymbol{X}^t) \quad \text{s.t. (10b), (10c), (10d).} \tag{20}$$

The problem **P3** is also a combinatorial optimization problem. While *brute-force* search can always find the optimal solution, the complexity can be high due to the non-linear property in (20). In order to address this problem, we aim to design an efficient polynomial runtime approximation algorithm to solve **P3** in the next subsection. We will show the performance guarantee of the approximation algorithm.

5.1 Approximated Oracle Solutions

To solve the problem **P3**, we first show that **P3** is a monotone submodular maximization problem with M knapsack and a matroid constraints. Below gives the definition of the monotone submodular maximization [42]:

Definition 1. *(Monotone Submodular Maximization.) A set function $F: 2^I \to \mathcal{R}$ is monotone increasing if $\forall A \subseteq B \subseteq I$, $F(A) \leq F(B)$. In addition, the function $F(\cdot)$ is submodular if $\forall A \subseteq B \subseteq I$ and $e \in I \setminus B$, $F(A \cup \{e\}) - F(A) \geq F(B \cup \{e\}) - F(B)$.*

Theorem 3. *P3 is a monotone submodular maximization with M knapsack and a matroid constraints problem.*

Proof. See in online Appendix D, available in the online supplemental material [41]. □

To facilitate the solution for non-convex HFL in **P3**, approximation algorithms are efficiently obtained approximate solutions in polynomial runtime. Some existing studies are focusing on solving the submodular maximization with knapsack and matroid constraints [19], [43], and they proposed the approximation guarantee to the optimal solution. In this paper, we use the Fast Lazy Greedy (FLGreedy) algorithm in [19] to achieve the approximated oracle solution with $\frac{1}{(1+\epsilon)(2+2M)}$-approximation guarantee, where ϵ is an error parameter in the FLGreedy algorithm. If ϵ is small, FLGreedy can achieve high approximation guarantee but have high computational complexity.

The FLGreedy algorithm acquires the client selection decision of the client sequentially, which starts with the all-$null$ decisions. In each edge aggregation round, it selects a client to an ES that gives the largest incremental learning utility. Because each client can only be selected at most one ES and each iteration decides the client selection decision for one client, the algorithm terminates in at most M iterations. Based on the results for submodular maximization with a knapsack and a matroid constraints in [19], the FLGreedy algorithm guarantees to yield a $\frac{1}{(1+\epsilon)(2+2M)}$-approximation for **P3**:

Lemma 3. *In an arbitrary edge aggregation round t, let $\boldsymbol{s}^{*,t}$ be the client selection decision solved by FLGreedy algorithm and $\boldsymbol{s}^{\text{opt},t}$ be the optimal client selection decision for the problem **P3**, we will have $\mu(\boldsymbol{s}^{*,t}; \boldsymbol{X}^t) \geq \frac{1}{(1+\epsilon)(2+2M)} \mu(\boldsymbol{s}^{\text{opt},t}; \boldsymbol{X}^t)$.*

Proof. The proof follows [19] and hence is omitted. □

We use FLGreedy to approximate the optimal client selection decision with oracle information on participated clients. Note that the actual performance of the FLGreedy algorithm is usually much better than the $\frac{1}{(1+\epsilon)(2+2M)}$ approximation ratio in practice.

5.2 Performance Analysis of COCS Policy for Non-Convex HFL

The regret in (11) is used when the optimal oracle solutions can be derivable. Because the FLGreedy algorithm can efficiently approximate the optimal oracle solution for **P3** instead of obtaining the optimal oracle solution. As such, we leverage the definition of δ-regret, which is usually used in MAB based on approximation algorithms [44]. For a δ-approximation algorithm (i.e., the solution $\boldsymbol{s}^t$ solved by the approximation algorithm satisfies $\mu(\boldsymbol{s}^t; \boldsymbol{X}^t) \geq \frac{1}{\delta} \mu(\boldsymbol{s}^{\text{opt},t}; \boldsymbol{X}^t)$), for problem **P3**, the δ-regret is

$$R^{\delta}(T) = \sum_{t=1}^{T} \frac{1}{\delta} \mu(\boldsymbol{s}^{\text{opt},t}; \boldsymbol{X}^t) - \sum_{t=1}^{T} \mu(\boldsymbol{s}^t; \boldsymbol{X}^t). \tag{21}$$

Because the FLGreedy algorithm for the problem **P3** has an approximation ratio of $\frac{1}{(1+\epsilon)(2+2M)}$, COCS policy obtains $\delta = \frac{1}{(1+\epsilon)(2+2M)}$. The definition of δ-regret essentially compares the utility of a policy with the lower bound of the approximated oracle solution.

Next, we aim to present the input parameters $K(t)$ and h_T and propose a regret upper bound for COCS policy. The regret analysis is also proved based on the Hölder condition in Assumption 1. Given the input parameters $K(t)$ and h_T in Theorem 4, we achieve a sublinear regret upper bound of COCS policy for **P3** as follows:

Theorem 4. *(δ-Regret Upper Bound.) If $K(t) = t^{\frac{2\alpha+2}{3\alpha+2}} \log(t)$, $h_T = \lceil T^{\frac{1}{3\alpha+2}} \rceil$, Hölder condition holds true and a δ-approximation is applied for optimization, then the dominate order of δ-regret $\mathbb{E}[R^{\delta}(T)]$ is $O(\frac{4N^2MB}{\delta c^{\min}} T^{\frac{2\alpha+2}{3\alpha+2}} \log(T))$.*

Proof. See in online Appendix E, available in the online supplemental material [41]. □

The regret upper bound given in Theorem 4 implies that COCS policy performs well enough if the subproblem in each edge aggregation round can only be derived approximately.

A sublinear δ-regret can be achieved based on the performance guarantee of δ-approximation algorithms.

6 Simulations

6.1 Setup

Datasets and Training Models. We set up the simulation with PyTorch and the computation is conducted by a workstation with 2 NVIDIA RTX 2080 GPUs. We have prepared two datasets for evaluating our proposed COCS algorithm. Specifically, MNIST dataset [45] under a logistic regression, which is widely used for strongly convex FL studies [2], [46]. For the non-convex HFL, we use CIFAR-10 dataset [47] by adopting a CNN with two 5×5 convolution layers, followed with 2×2 max polling, two fully-connected layers with 384 and 192 units, and finally a softmax output layer. The NLP dataset is built from Complete Works of William Shakespeare Shakespeare dataset [1]. We use a two-layer LSTM classifier with 100 hidden units and an 8D embedding layer. To this end, there is a densely-connected layer. For each simulation, we distribute the dataset among $N = 80$ clients in a general non-iid fashion such that each clients only contains samples of only two labels.

Contexts. For the context generation, in each edge aggregation round, we assume the allocated bandwidth of all clients is sampling from a uniform distribution between $\mathcal{U} \sim [0.3, 1]$ for MNIST dataset and $\mathcal{U} \sim [2, 4]$ for CIFAR-10 dataset, since the data and model sizes are different for each dataset. Likewise, the available computation capacity of all clients is also sampling from $\mathcal{U} \sim [2, 4]$ for MNIST dataset, $\mathcal{U} \sim [8, 15]$ for CIFAR-10 dataset, and $\mathcal{U} \sim [1, 2]$ for Shakespeare dataset. The distance $d^t_{n,m}$ between client and ES is from $\mathcal{U} \sim [0, 2]$km. For SAFA, the distance d^t_n between clients and CS is from $\mathcal{U} \sim [0, 10]$km.

Parameters of HFL Networks. Our simulated HFL network includes 3 ESs and 50 clients, where the radius of each ES is 2km. Within the coverage area, there are several clients randomly distributed and communicated by the corresponding ES through a wireless channel in each edge aggregation round. In the edge aggregation round t, the downlink and uplink channel gain are decomposed of both small-scale fading and large-scale fading, where the small-scale fading is set as Rayleigh distribution with uniform variance and the large-scale fading are calculated by the path-loss with random shadowing $g^t_{\mathrm{DT},n} = g^t_{\mathrm{UT},n} = 37.6 \log(d^t_{n,m}) + 128.1$, where d represent the distance of client-ES pair (n, m). We set parameter of our simulated HFL network for the two datasets shown in Table 1.

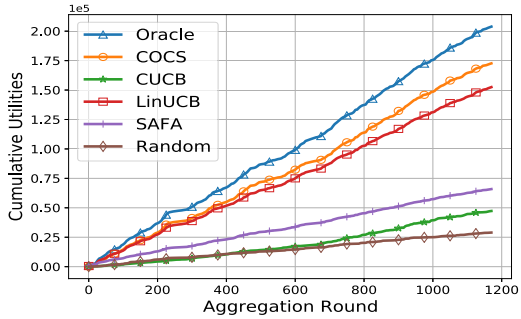
TABLE 1
HFL Network Parameters (If the Parameter has Three Values, the First One is on MNIST Dataset, Second is on CIFAR-10 Dataset, and Third is on Shakespeare Dataset)

Parameter	Value
Number of clients, N	80
Number of ESs, M	3
Size of local model updates, s	0.18, 18.7, 31.3 (Mbits)
Computation workload, q^n	2.41, 28.3, 11.6 (Mbytes)
Transmission power, P^t_n	23dBm
Deadline, τ^{dead}	4, 20, 30 (sec)
Pricing function, $b_n(f_n)$	$\mathcal{U} \sim [0.5, 2]$ per Mbytes
Budget on each ES B	3.5, 40, 18
α in Hölder condition	1
h_T in COCS	5
Local training epochs, E	2, 5, 5
Global aggregation, T_{ES}	5
Learning rate, η	0.001, 0.1, 0.01

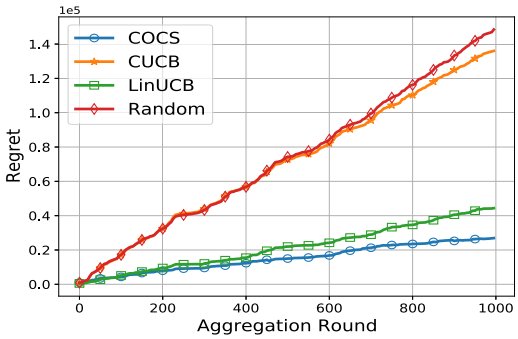
6.2 Comparison Benchmarks

We compare the COCS policy to the five benchmarks:

1) *Oracle:* the Oracle algorithm knows precisely whether one client can be received by the corresponding ES before the deadline τ^{dead} with any observed context. In each aggregation round, it makes a client selection decision to maximize the utility in (10a): brute-force for the strongly convex HFL and GreedyLS for non-convex HFL.
2) *Combinatorial UCB (CUCB):* CUCB is designed based on a classical MAB policy UCB [28]. It develops combinations of client selection decisions on all ESs to enumerate NO's decision $\boldsymbol{s}$. CUCB runs UCB with feasible NO selection decisions $\boldsymbol{s}$ and learns the expected utility for each $\boldsymbol{s}^t$ in edge aggregation round. Since CUCB does not fit for the time-varying arm set, we set the static computation and transmission resource for client-ES pairs.
3) *LinUCB:* LinUCB [48] is a contextual variant of running CUCB. LinUCB also aims to learn the expected utility for client selection decision $\boldsymbol{s}$, which assumes that the utility of an arm is a linear function of client-ES pairs' contexts.
4) *Random:* The Random algorithm selects a client to an accessible ES randomly in each edge aggregation round under these two constraints.



(a) Cumulative Utilities.



(b) Regret.

Fig. 3. Comparisons on cumulative utilities under logistic regression on MNIST dataset.

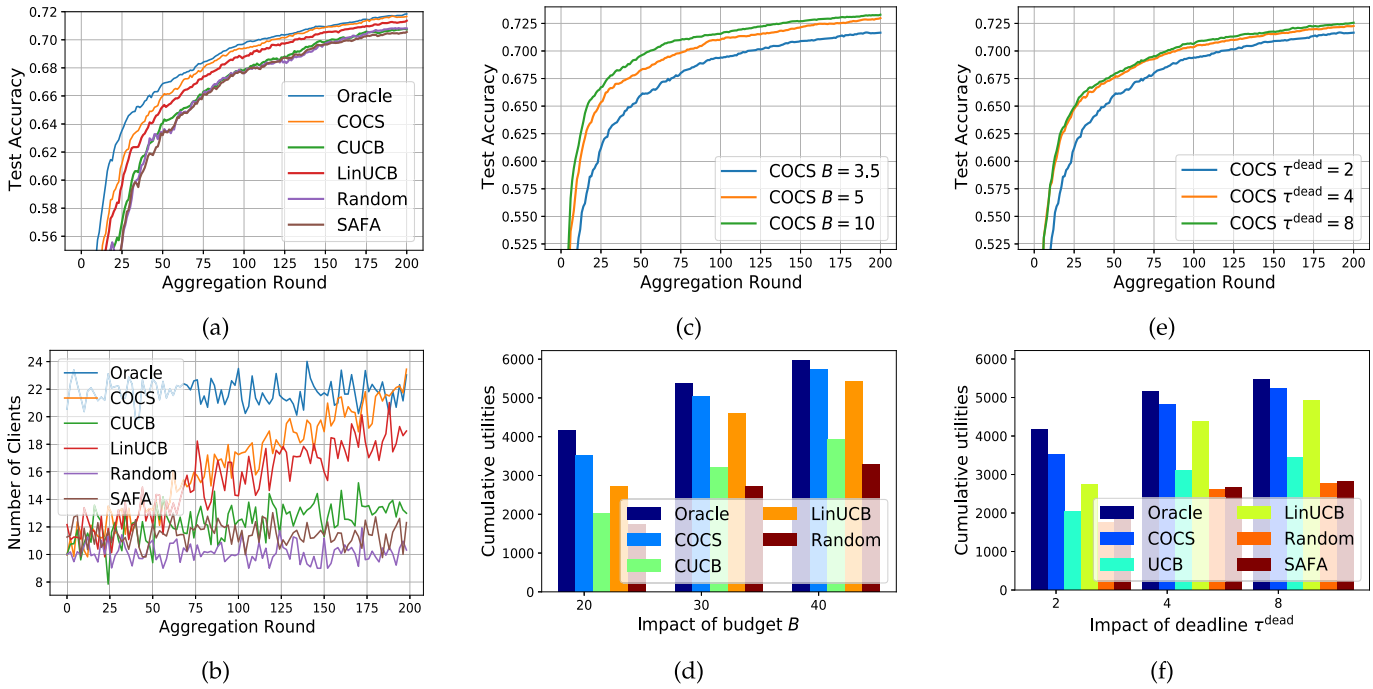


Fig. 4. (a) Training performance of 5 client selection policies based on logistic regression; (b) Temporal number of successful participated clients in each edge aggregation round (i.e., $\sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{S}_m^t} X_{n,m}^t = 1$); (c) Training performance of different budget B; (d) Cumulative utilities of different budget B; (e) Training performance of different deadline τ^{dead} and (f) Cumulative utilities of different deadline τ^{dead}.

TABLE 2
Final Accuracy and Edge Aggregation Round to the Targeted Accuracy (MNIST 70%, CIFAR-10 60%, and Shakespeare 45%) Under non-iid (iid) Dataset of Different Benchmarks

Policy	MNIST		CIFAR-10		Shakespeare	
	Final Accuracy	Round	Final Accuracy	Round	Final Accuracy	Round
Oracle	71.90 (74.85)	111 (70)	68.41 (73.81)	92 (58)	58.83 (61.24)	223 (180)
COCS	71.84 (74.57)	121 (74)	67.93 (73.12)	101 (63)	57.66 (60.39)	246 (192)
UCB	71.15 (73.53)	147 (86)	63.76 (68.16)	175 (120)	54.25 (57.19)	289 (219)
LinUCB	71.67 (74.26)	134 (79)	65.19 (72.30)	133 (82)	55.52 (59.87)	251 (201)
Random	70.81 (72.31)	161 (103)	62.25 (67.79)	207 (146)	47.30 (52.41)	339 (297)
SAFA	71.07 (73.40)	152 (93)	63.31 (66.93)	184 (129)	53.63 (56.18)	296 (230)

5) *SAFA:* SAFA [10] is an asynchronous FL algorithm. If some clients cannot be received before deadline, they will join to the next aggregation stage.

6.3 Performance Evaluation of Strongly Convex HFL

1) Comparison on Cumulative Utilities. Fig. 2 shows the cumulative utilities and regret obtained by the COCS policy and the other 4 benchmarks during 1,000 edge aggregation rounds under logistic regression on the MNIST dataset. For the cumulative utilities in Fig. 3a, it is observed that the Oracle policy achieves the highest cumulative utilities and provides an upper bound to the other benchmarks as expected. Among the others, COCS policy significantly outperforms the other benchmarks and has a closed cumulative utility performance to Oracle. The profit of the context of client-ES pairs can be shown by comparing the performance of context-aware policy (LinUCB) and context-unaware policies (CUCB and Random). More specifically, the results show that the cumulative utilities of CUCB are similar to the Random policy. The disadvantage of CUCB comes from the following two reasons: (1) an arm of CUCB is a combination of the selection decisions of all client-ES pairs, and hence CUCB obtains a large number of arms. This means that CUCB is difficult to enter the exploitation phase. (2) CUCB fails to capture the connection between context and clients. The cumulative utilities of LinUCB are based on the context, for which a CUCB arm is not effective to produce a good result due to the large arm set. The reason that the cumulative utilities of SAFA are low (only higher than random) is because the average distance between clients and CS is much larger than HFL. Therefore, due to the low value of τ^{dead}, most of clients cannot be received within one aggregation round. In Fig. 3b, we notably depict the regret generated by the 4 benchmarks. It is easy to observe that our proposed COCS policy incurs a sublinear regret.

2) Training Performance and Client Selection Results. We use two metrics to evaluate the training performance based on client selection policies: edge aggregation rounds to achieve targeted accuracy and final accuracy. In Fig. 4a, we present the training performance under logistic regression on the MNIST dataset from different client selection benchmarks.

TABLE 3
Final Accuracy and Edge Aggregation Round to the Targeted Accuracy (MNIST 70%, CIFAR-10 60%, and Shakespeare 45%) Under non-iid Dataset Without Z Constriant

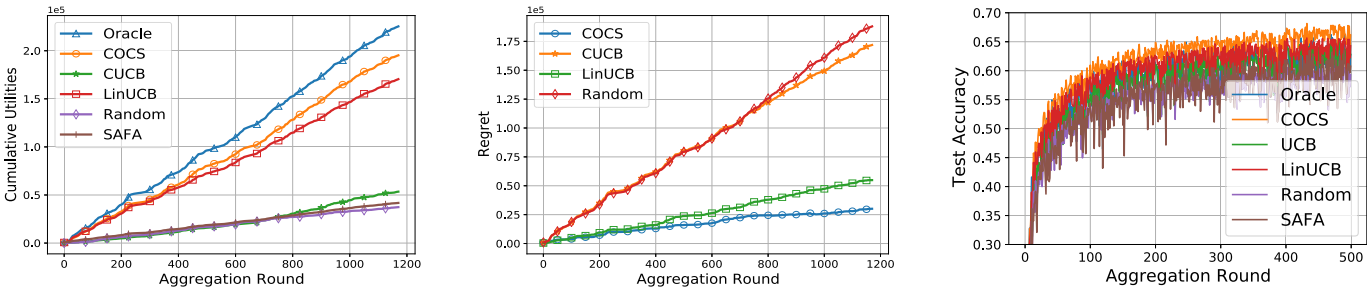
Policy	MNIST		CIFAR-10		Shakespeare	
	Acc.	Rou.	Acc.	Rou.	Acc.	Rou.
Oracle	71.90	111	68.41	92	58.76	227
COCS	71.20	126	66.10	109	57.13	251
UCB	69.98	181	61.54	196	53.11	311
LinUCB	70.97	142	64.43	146	55.03	259
Random	68.16	208	60.48	276	43.49	386

The Oracle policy, as expected, results in the fastest convergence speed and highest accuracy among all benchmarks. Although COCS performs slower than Oracle during the first several rounds due to exploration, it achieves similar testing accuracy to Oracle in 200th round. In particular, it easy to observe that COCS outperforms others. Due to the insufficient selection of clients all the rounds, Random selecting clients is considerably inferior to all other benchmarks. For clarifying the training performance, we present an auxiliary Table 2 to emphasize the results, in which the targeted accuracy on MNIST dataset is set 70%. As is shown, our proposed COCS policy only uses 121 rounds to achieve 70% test accuracy, which is 36, 13, 40, 38 rounds faster than CUCB, LinUCB, Random, and SAFA.

In Fig. 4b, we show that the temporal number of clients are selected in each round. The upper bound and lower bound of clients are Oracle and Random policies. Although COCS, LinUCB, and CUCB all have increasing levels due to obtaining historical experiences, it is easy to observe that the increase of CUCB is very slow, and COCS outperforms the other two benchmarks. The reason why the number of successful participated clients are few in the first several rounds via the MAB based policies is that most of selected clients cannot be received by ESs before deadline τ^{dead}.

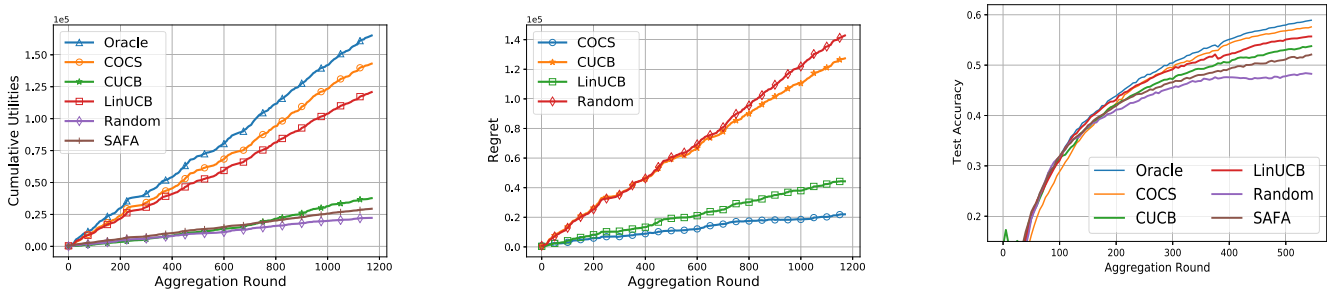
In addition, Table 3 also presents the training performance without minimum received clients constraint (i.e., even though ESs cannot receive $Z = 9$ clients before the deadline τ^{dead}, they must process the EC stage). We can see that there is less impact on Oracle, COCS and LinUCB policies, because they usually guarantee to successfully select more than 9 clients on each ES. However, due to fewer clients participating edge aggregation, the accuracy of UCB and Random degrade from 65.19%-61.54% and 62.25%-60.48% on the non-iid CIFAR-10 dataset.

3) Impact of Budget B. Fig. 4c shows that the training performance of COCS under different budgets B ($B = 3.5, 5$ and 10). It is easy to observe that COCS has better performance when NO increases the budget B. This is simply to explain because increasing the budget can select more clients in each round in order to increase the utility of HFL. In particular, when $B = 5$ NO only uses 77 rounds to achieve 70% test accuracy, which is much faster than $B = 3.5$. However, the training performance does not have significant improvement from $B = 5$ to 10. Fig. 4c presents the cumulative utilities of these five benchmarks after 200 edge aggregation rounds. Clearly, for all the benchmarks, the NO can achieve higher cumulative utilities with increasing budget. As can be observed, when the budget increases from 3.5 to 5, the benefit of client selection gradually increases. However, although the payment budget is set large, the redundant selection does not have significant improvement. The reason may be because some client-ES pairs performing poor transmission status cannot be successfully received by



(a) Cumulative utilities. (b) Regret. (c) Training performance.

Fig. 5. Performance evaluation under CNN on CIFAR-10 dataset.



(a) Cumulative Utilities. (b) Regret. (c) Training performance.

Fig. 6. Performance evaluation under LSTM on Shakespeare dataset.

TABLE 4
Final Accuracy and Edge Aggregation Round to the Targeted Accuracy (MNIST 70%, CIFAR-10 60%, and Shakespeare 45%) Under non-iid Dataset With the Optimal Budget B on Each ES

	MNIST		CIFAR-10		Shakespeare	
Policy	Acc.	Rou.	Acc.	Rou.	Acc.	Rou.
Oracle	71.96	109	68.07	95	58.75	227
COCS	71.88	118	67.16	108	57.52	254
UCB	71.26	140	63.49	180	54.02	294
LinUCB	71.73	131	65.03	137	55.39	256
Random	71.32	155	62.31	202	47.32	338

ESs before the deadline τ^{dead}, even if NO increases the budget of computation resources.

4) Impact of Deadline τ^{dead}. Fig. 4e depicts the training performance and Fig. 4f depicts the cumulative utilities under different deadlines $\tau^{\text{dead}} = 2$, 4 and 8. We can see that when NO increases the value of deadline, the number of clients increases gradually, which performs similarly to increasing budget B. However, if NO sets the deadline too large (i.e., $\tau^{\text{dead}} = 8$), training performance and cumulative utilities perform similarly to $\tau^{\text{dead}} = 4$. More specifically, cumulative utilities of COCS only increase from 5,125 to 5,378. It is easy to observe that the impact of increasing τ^{dead} is less than increasing budget B. We consider that the less budget can control the number of selected clients, which is more dominant than the impact of deadline τ^{dead} for the training performance both on convergence and accuracy.

6.4 Performance Evaluation of Non-Convex HFL

We show the performance of non-convex HFL under the CNN model on the CIFAR-10 dataset and LSTM model on the Shakespeare dataset, where the utility is quadratically related to the number of participated clients. We set the error parameter of FLGreedy algorithm $\epsilon = 0.3$. Fig. 5a depicts the cumulative utilities and Fig. 5b depicts regret. Similar to the performance of strongly convex HFL in Fig. 2, the Oracle policy performs the best cumulative utilities as expected, and COCS outperforms the other 3 benchmarks (e.g., 1.7× higher than CUCB policy). In particular, the difference of cumulative utilities between Oracle and COCS is smaller than the result in Fig. 2a, since this is an approximated Oracle solution in this case. It is also observed that COCS achieves a sublinear regret in Fig. 5b. Fig. 5c shows the test accuracy of different client selection benchmarks on the CIFAR-10 dataset. Since the training model and data size of CIFAR-10 is complicated enough, training performance of different benchmarks are clear to see. Oracle has the best performance among all benchmarks, which achieves 68.41% test accuracy. In Table 2, COCS policy can achieve 67.93% test accuracy, which is 4.17%, 1.74%, 5.68%, 4.52% higher than LinUCB, CUCB, Random, and SAFA policies.

TABLE 5
Average Time Cost (sec) of Oracle and COCS Client Selection Policies on Each ES

Policy	MNIST	CIFAR-10	Shakespeare
Oracle	3.73	5.18	5.36
COCS	3.41	4.09	4.22

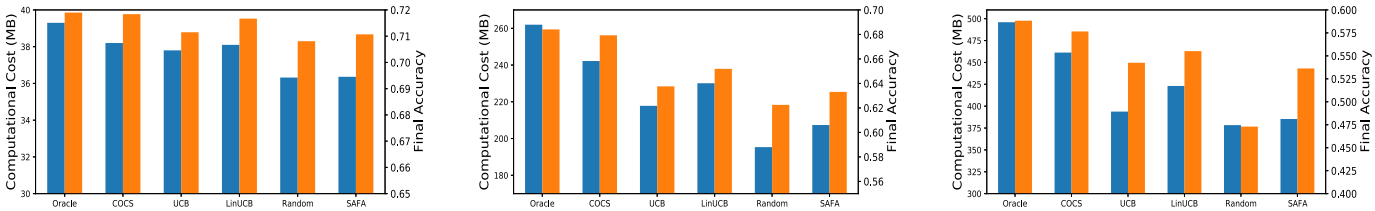
Fig. 6 shows the cumulative utilities, regret, and training performance on the Shakespeare dataset under LSTM. The results are similar for other datasets, where COCS consistently has the best performance except Oracle; and Random performs the worst. From the convergence performance perspective, COCS also outperforms the other three benchmarks. We can conclude that COCS policy can improve the training performance for HFL significantly both on strongly convex and non-convex settings.

6.5 Other Simulation Results

We present the training performance with optimal budget B on each ES in Table 4. It is observed that the training performance of equal budget B and optimal budget B does not have obvious difference (e.g., on Shakespeare dataset, the Oracle policy achieves 58.83% with equal B and 58.75% with optimal B). In addition, all the training performances on MNIST dataset improve, but decrease on CIFAR-10 and Shakespeare datasets. This matches our intuition in Section 3.5, which may be due to the larger variance of the number of selected clients on each ES. For example, on CIFAR-10 dataset, COCS policy obtains 57.66% with equal B and 57.52% with optimal B.

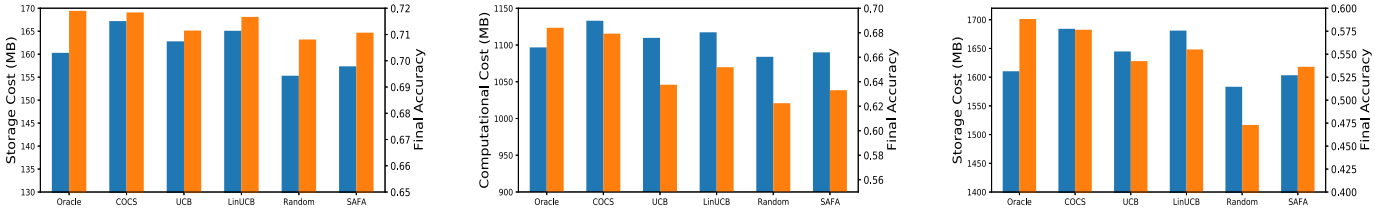
The time cost of Oracle and COCS policies is shown in Table 5. We can see that all the client selection can be finished before τ^{dead}, which means client selection does not have impact on the training time. Moreover, since the experience and counter in CC-MAB are both scalars, which indicates that the client selection does not depend on the complexity of dataset. For example, the time cost is 4.09 sec on CIFAR-10 and 4.22 sec on Shakespeare dataset.

To compare the efficiency of computation and storage, we show the trade-off between computational cost and final accuracy in Fig. 7, and trade-off between storage cost and final accuracy in Fig. 8. It is worth noting that both costs include training cost, edge aggregation cost and client selection cost. It means that if NO selects more clients, the cost should increase. For example, on the Shakespeare dataset in Fig. 7c, COCS selects 17.29 more clients than the Random policy, but



(a) MNIST dataset. (b) CIFAR-10 dataset. (c) Shakespeare dataset.

Fig. 7. Computational cost and testing accuracy on different dataset.



(a) MNIST dataset. (b) CNN dataset. (c) Shakespeare dataset.

Fig. 8. Storage cost and testing accuracy on different dataset.

only increases 57.36MB, which indicates that client selection does not waste much computational cost. In addition, the reason that Oracle policy incurs most computational cost is due to the brute force algorithm. Fig. 8 exhibits the similar performance to Fig. 7, the dominant storage is due to saving the local model updates, which are high-dimensional matrices. The calculation of CC-MAB is only for low dimensional data since the experience and counter are both scalars. For example, on CIFAR-10 dataset, saving these parameters only cost 3.58MB in total. In summary, we consider that making client selection does not have obvious impact on training cost.

7 Conclusion

In this paper, we investigated the client selection problem to improve the training performance for HFL. An online decision-making policy, called Context-aware Online Client Selection (COCS), was designed for NO to make proper client selection decisions for each client-ES pair. COCS was developed based on the CC-MAB framework, where NO observes the context (i.e., downloading channel state and local computation resources) of client-ES pairs and learns the participated probabilities to select clients and guide rental computation resources. Our proposed COCS policy departs from conventional optimization-based algorithms, which is able to work in HFL networks with uncertain information. More specifically, COCS addresses many practical challenges for HFL networks, and it is easy to implement and achieves a provably asymptotically optimal performance guarantee. Although our COCS policy has presented superior performance in extensive HFL experiments, there are still several future research questions. For example, considering dynamic partition of context space may improve the selection results, since it can generate more appropriate hypercubes. A theoretical improvement of convergence speed from COCS should also be considered.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," 2019, *arXiv:1907.02189*.

[3] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[4] X. Zhu, J. Wang, Z. Hong, and J. Xiao, "Empirical studies of institutional federated learning for natural language processing," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2020, pp. 625–634.

[5] P. Guo, P. Wang, J. Zhou, S. Jiang, and V. M. Patel, "Multi-institutional collaborations for improving deep learning-based magnetic resonance image reconstruction using federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2423–2432.

[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.

[7] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[8] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 500–514, 2020.

[9] H. Sun, X. Ma, and R. Q. Hu, "Adaptive federated learning with gradient compression in uplink NOMA," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16 325–16 329, Dec. 2020.

[10] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, May 2021.

[11] X. Li, Z. Qu, B. Tang, and Z. Lu, "Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients," 2021, *arXiv:2102.06329*.

[12] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[13] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Local averaging helps: Hierarchical federated learning and convergence analysis," 2020, *arXiv:2010.12998*.

[14] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Hierarchical quantized federated learning: Convergence analysis and system design," 2021, *arXiv:2103.14272*.

[15] 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.*, 2020, pp. 1698–1707.

[16] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1188–1200, Feb. 2021.

[17] L. Chen and J. Xu, "Budget-constrained edge service provisioning with demand estimation via bandit learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2364–2376, Oct. 2019.

[18] L. Chen, Z. Lu, P. Zhou, and J. Xu, "Learning optimal sniffer channel assignment for small cell cognitive radio networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 656–665.

[19] A. Badanidiyuru and J. Vondrák, "Fast algorithms for maximizing submodular functions," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2014, pp. 1497–1514.

[20] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, "Fast-convergent federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 201–218, Jan. 2021.

[21] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, "Communication-efficient edge AI: Algorithms and systems," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2167–2191, Oct.–Dec. 2020.

[22] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 10 351–10 375.

[23] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "Clustered sampling: Low-variance and improved representativity for clients selection in federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 3407–3416.

[24] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. 15th USENIX Symp. Oper. Syst. Des. Implementation*, 2021, pp. 19–35.

[25] C. Li, X. Zeng, M. Zhang, and Z. Cao, "PyramidFL: A fine-grained client selection framework for efficient federated learning," in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2022.

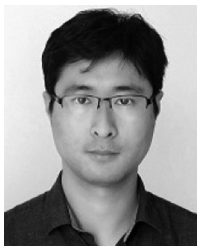
[26] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 3975–3986, Jul. 2021.

[27] J. Wu *et al.*, "Hierarchical personalized federated learning for user modeling," in *Proc. Web Conf.*, 2021, pp. 957–968.
[28] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[29] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.
[30] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1552–1564, Jul. 2021.
[31] K. Wei *et al.*, "Low-latency federated learning over wireless channels with differential privacy," 2021, *arXiv:2106.13039*.
[32] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.
[33] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1387–1395.
[34] C. Yang *et al.*, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proc. Web Conf.*, 2021, pp. 935–946.
[35] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.
[36] C. Avin, Y. Emek, E. Kantor, Z. Lotker, D. Peleg, and L. Roditty, "SINR diagrams: Towards algorithmically usable SINR models of wireless networks," in *Proc. 28th ACM Symp. Princ. Distrib. Comput.*, 2009, pp. 200–209.
[37] H. Choi and H. Moon, "Throughput of CDM-based random access with SINR capture," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15 046–15 056, Dec. 2020.
[38] I. I. Cplex, "V12. 1: User's manual for CPLEX," *Int. Bus. Mach. Corporation*, vol. 46, no. 53, 2009, Art. no. 157.
[39] W. Zhang *et al.*, "Optimizing federated learning in distributed industrial IoT: A multi-agent approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3688–3703, Dec. 2021.
[40] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[41] Z. Qu, R. Duan, L. Chen, J. Xu, Z. Lu, and Y. Liu, "Context-aware online client selection for hierarchical federated learning," 2021, *arXiv:2112.00925*.
[42] J. Lee, *A First Course in Combinatorial Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[43] C. Chekuri, J. Vondrák, and R. Zenklusen, "Submodular function maximization via the multilinear relaxation and contention resolution schemes," *SIAM J. Comput.*, vol. 43, no. 6, pp. 1831–1879, 2014.
[44] L. Chen, A. Krause, and A. Karbasi, "Interactive submodular bandit," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 141–152.
[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[46] S. Yang, B. Ren, X. Zhou, and L. Liu, "Parallel distributed logistic regression for vertical federated learning without third-party coordinator," 2019, *arXiv:1911.09824*.
[47] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.
[48] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 661–670.

Zhe Qu (Student Member, IEEE) received the BS degree from Xiamen University, Xiamen, China, in 2015, and the MS degree from the University of Delaware, Newark, Delaware, in 2017. He is currently working toward the PhD degree in systems and security in the Department of Electrical Engineering, University of South Florida, Tampa, Florida. His primary research interests include network and mobile system security, and machine learning for networks.

Rui Duan (Student Member, IEEE) received the BS degree from the University of Shanghai for Science and Technology, Shanghai, China, in 2015, and the MS degree from the University of South Florida, Tampa, Florida, in 2017. He is currently working toward the PhD degree in the Department of Electrical Engineering, University of South Florida, Tampa, Florida. His primary research interests include machine learning security, and security usability and measurement.

Lixing Chen (Member, IEEE) received the PhD degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, University of Miami, in 2020. He is currently an assistant professor with the Institute of Cyber Science and Technology, Shanghai Jiao Tong University, China. His primary research interests include mobile edge computing and machine learning for networks, and cybersecurity.



Jie Xu (Senior Member, IEEE) received the BS and MS degrees in electronic engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively, and the PhD degree in electrical engineering from the University of California, Los Angeles (UCLA), in 2015. He is currently an associate professor with the Department of Electrical and Computer Engineering, University of Miami. His research interests include mobile edge computing/intelligence, machine learning for networks, and network security. He received the NSF CAREER Award in 2021.

Zhuo Lu (Senior Member, IEEE) received the PhD degree from North Carolina State University, in 2013. He is currently an associate professor with the Department of Electrical Engineering, University of South Florida. He is also affiliated with the Florida Center for Cybersecurity and by courtesy with the Department of Computer Science and Engineering. His research has been mainly focused on both theoretical and system perspectives on communication, network, and security. His recent research is equally focused on machine learning and AI perspectives on networking and security. He received the NSF CISE CRII Award in 2016, the Best Paper Award from IEEE GlobalSIP in 2019, and the NSF CAREER Award in 2021.

Yao Liu (Senior Member, IEEE) received the PhD degree in computer science from North Carolina State University, in 2012. She is currently an associate professor with the Department of Computer Science and Engineering, University of South Florida. Her research is related to computer and network security, with an emphasis on designing and implementing defense approaches that protect mobile, network and computer technologies from being undermined by adversaries. Her research interests also lie in the security applications for cyber-physical systems, Internet of Things, and machine learning. She was an NSF CAREER Award recipient in 2016. She also received the ACM CCS Test-of-Time Award by ACM SIGSAC in 2019.