Context-Aware Online Client Selection for Hierarchical Federated Learning

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..

Keywords: 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 socalled "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 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 (CCMAB) [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.

2 Related works

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 Method

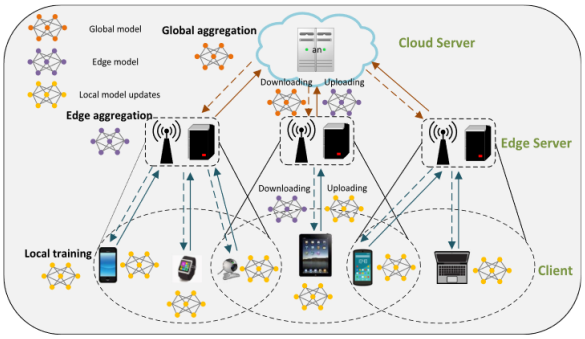

Figure 1. The Hierarchical Federated learning architecture of FL

3.1 Algorithm Overview

The Hierarchical federated learning framework is like Fig.1. Based on it, the paper design COCS policy for each edge server to choose clients. 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 $\boldsymbol{\phi}^t = \left\{\boldsymbol{\phi}^t_{n,m}\right\}_{n,m\in\mathcal{N}^t_m}, \phi^t_{n,m} \in \Phi$.. (ii) NO determines its selection decision s^t based on the observed context information fft in the current round t and the knowledge learned from the previous $t-1$ rounds. (iii) The selection decision sst is applied. If $s^t = 0, \forall n \in s^t_m$, 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 $s\Delta^t_n$ 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 ft n;m of client-ES pair $\phi^t_{n,m}$. Note that in this paper, we consider how to properly allocate the computational resources to improve the training performance in FL.

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 $C_{n,m}(l) = 0, \mathcal{E}_{n,m}(l) = \emptyset, \hat{\beta}_{n,m} = 0, \forall n \in \mathcal{N} \forall m \in \mathcal{M}, \forall l \in \mathcal{L}_T$.

for $t = 1, \ldots, T$ **do**
  Observes the context of clients on ESs ϕ^t;
  Determine $\mathcal{C}^{ue,t}_n$ and $\mathcal{N}^{ue,t}_n$, and estimate the participated clients $\hat{X}^t$ based on the contexts ϕ^t;
  if $\mathcal{N}^{ue,t}_n \neq \emptyset$ **then** ▷ Exploration
    if $\mathcal{N}^{ue,t}_n \neq \emptyset, \forall n$ **then**
      Determine s^t by solving (14);
    else
      Get $\tilde{s}^t$ by solving (15);
      Select the clients in $N^{ed,t}$ by solving (17) and determine the client selection decision s^t;
    end if
  else ▷ Exploration
    Determine the client selection decision s^t by solving (18);
  end if
  for $n \in \mathcal{N}$ **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: $\hat{p}_{n,m}(l) = \frac{\hat{p}_{n,m}(l)C_{n,m}(l)+X}{C_{n,m}(l)+1}$;
    Update counters: $C_{n,m}(l) = C_{n,m}(l) + 1$;
  end for
end for

3.2 Analysis of Algorithm

Initialization Phase. Given parameter h_T , the proposed policy first creates a partition denoted by $\boldsymbol{\mathcal{L}}_T$ 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 $C_{n,m}^t(l)$ 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 $n \in N_n^t$ m is selected to an ES m;2)the ES m successfully receives the client n before tdead (i.e., $\tau_n^t \leq \tau_{\text{dead}}$); 3) the context of client-ES pair (n, m) belongs to l (i.e., $\phi_{n,m}^t \in l$). The counter $C_{n,m}^t(l)$ 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 $\mathcal{E}_{n,m}^t(l)$ 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 $\mathcal{E}_{n,m}^t(l)$ is useful for making the future decision whether coming into exploration or exploitation phase. Based on the observed participation indicators in $\mathcal{E}_{n,m}^t(l)$,the estimated participated probability for a selection event $V_{n,m,l}$ is computed by:

$$\hat{p}_{n,m}^t(l) = \frac{1}{C_{n,m}^t(l)} \sum_{X \in \mathcal{E}_{n,m}^t(l)} X \qquad (12)$$

Hypercube Identification Phase. If the local model updates Dt n of client $n \in N_m^t$can be successfully received by an ES m in edge aggregation round t, we obtain that lt n;m is the hypercube for the context ft n;m, the estimated participated probability of client n on ES m is $\hat{X}_{n,m}^t = \{\hat{p}_{n,m}^t(l_{n,m}^t)\}$. Let $\hat{\boldsymbol{X}}^t = \{\hat{X}_{n,m}^t\}_{\forall m, n \in \mathcal{N}_m^t}$ 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 $\mathcal{L}_m^{\text{ue}}(\boldsymbol{\phi}^t)$ for the ES m in edge aggregation round t as follows:

$$\mathcal{L}_m^{\text{uc},t} \triangleq \{l \in \mathcal{L}_T | \quad \begin{array}{c} \exists \phi_{n,m}^t \in \boldsymbol{\phi}^t, \phi_{n,m}^t \in l, \tau_n^t \leq \tau_{\text{dead}} \\ \text{and} C_{n,m}^t(l) \leq K(t) \end{array} \Big\} \quad (13)$$

Also, let $\mathcal{N}_m^{\text{ue},t}(\boldsymbol{\phi}^t) \triangleq \{n \in \mathcal{N}_m^t | l_{n,m}^t \in \mathcal{L}_m^{\text{ue},t}(\boldsymbol{\phi}^t)\}$ 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 $\mathcal{N}_m^{\text{ue},t}(\boldsymbol{\phi})^t$, COCS can identify that then either enters an exploration phase or an exploitation phase.

Exploration Phase. First, let $\mathcal{N}_m^{\text{ue},t}(\boldsymbol{\phi^t}) \triangleq \{n \in \mathcal{N}_m^t | \mathcal{N}_m^{\text{ue},t} \neq \emptyset\}$ denote an ES m has under-explored clients, and $\mathcal{N}_m^{\text{ed},t}(\boldsymbol{\phi}^t) \triangleq \mathcal{N} \setminus \mathcal{N}^{\text{ue},t}(\boldsymbol{\phi}^t)$ denote the ES m does not have underexplored clients. If the ES m has a non-empty $N_m^{ue,t}$, 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_n^t. Thus, COCS policy aims to select as many clients that have under-explored ESs sequentially solved by the following optimization:

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

where $|s^t|$ is the size of the collection $\boldsymbol{s}^t = \{\boldsymbol{s}_1^t, \boldsymbol{s}_2^t, \ldots, \boldsymbol{s}_M^t\}$. We use the **cost-effective greedy** to achieve this optimization .

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

selects ESs that have under-explored clients $m \in \mathcal{N}_m^{ue,t}$ by solving the following optimization:

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

$$\text{s.t.} \quad \sum_{n \in \widetilde{\boldsymbol{s}}_m^t} c_n(y_n^t) \leq B, \quad \forall n \in \mathcal{N}_m^{\mathbf{uc},t}, \forall m \in \mathcal{M} \tag{15b}$$

$$s_m^t \in \mathcal{N}_m^t \cup \{null\}, \quad \forall n \in \mathcal{N}_m^{\mathbf{ue},\iota} \tag{15c}$$

$$\tilde{\boldsymbol{s}}_m^t \cap \tilde{\boldsymbol{s}}_{m'}^t = \emptyset, \qquad m, m' \in \mathcal{M}, \forall t. \tag{15d}$$

where $\mathbf{s}^t$ is client selection decision on ES m that has underexplored clients and $|\tilde{s}^t|$ is the size of the collection $\tilde{s}^t = \{\tilde{\boldsymbol{s}}_1^t, \tilde{\boldsymbol{s}}_2^t, \ldots, \tilde{\boldsymbol{s}}_M^t\}$.

Second, ESs aim to select the explored clients $\forall n \in \mathcal{N}_m^{\text{ed},t} s$. Here, we assume that there exists ESs that $B - \sum_{n \in \bar{\boldsymbol{s}}_m^t} c_n(y_n^t) \geq c^{\min,t}, m \in \mathcal{M}$, where $c^{\min,t} = \min_{n \in \mathcal{N}_m^{\text{ed},t}} c_n(y_n^t), \forall m$. Therefore, ESs can select the clients $n \in \mathcal{N}_m^{\text{ed},t}$ with the following constraint:

$$\sum_{n \in \boldsymbol{s}_m^t \setminus \bar{\boldsymbol{s}}_m^t} c_n(y_n^t) \leq B - \sum_{n \in \bar{\boldsymbol{s}}_m^t} c_n(y_n^t), \forall n \in \mathcal{N}_m^{\text{ed},t}. \tag{16}$$

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

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

$$\text{s.t.} B - \sum_{n \in \widetilde{\boldsymbol{s}}_m^t} c_n(y_n^t) \geq c^{\min,t}, m \in \mathcal{M}, \tag{17b}$$

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

We treat this optimization as a **0-1 knapsack problem** and use **dynamic programming** to solve for the client to be selected.

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

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

In a similar way to the solution of (17a), we consider this optimization problem as a **0-1 knapsack problem** and solve it using **dynamic programming**.

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 t_dead; then, it updates $\hat{p}_{n,m}^t(l)$ and $C_{n,m}^t(l)$ of each hypercube e$l \in \mathcal{L}_T$..

3.3 Complexity of the Algorithm

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.

4 Implementation details

4.1 Comparing with the released source codes

There is **no source code** for this work, so all work needs to be reproduced based on the contents of the paper.

In contrast to the work described in the original paper, this replication provides the model training code for HFL and the code for the COCS strategy and other benchmarks.

4.2 Main contributions

I reproduce the Hierarchical Federated Learning framework and COCS strategy of the original paper with the following main work:

- Construct the code for the **hierarchical federated learning framework**, including dataset partitioning, client assignment, simulated communication at the cloud-side end, wireless communication state simulation, and federation learning.
- Understand how the client context space is constructed and implement it.
- Use **cost-effective greedy algorithm** to realize the exploration phase of COCS.
- Use **dynamic programming algorithms** to complete the development phase of COCS.
- Construct CUCB and Randomized Selection algorithm as Benchmark. compare the selection efficiency and test accuracy of COCS algorithm with other two Benchmarks.

4.3 Experimental environment setup

Datasets and Training Models. We set up the simulation with PyTorch and the computation is conducted by a workstation with 1 NVIDIA RTX 2060 GPU. We have prepared a dataset for evaluating our proposed COCS algorithm. Specifically, MNIST dataset [45] under a logistic regression, which is widely used for strongly convex FL studies [2], [2]. 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 $\mathcal{U} \sim [2, 4]$.

Likewise, the available computation capacity of all clients is also sampling from $\mathcal{U} \sim [2, 4]$ for MNIST dataset. The distance d_n^t between client and ES is from $\mathcal{U} \sim [0, 2]km$.

Parameters of HFL Networks. Our simulated HFL network includes 3 ESs and 80 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 $g_{\mathrm{DT},n}^t = g_{\mathrm{UT},n}^t = 37.6\log\left(d_{n,m}^t\right) + 128.1$,, where d represent the distance of client-ES pair (n, m).We set parameter of our simulation.

Comparison Benchmarks. We compare the COCS policy to the two benchmarks:

- 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 s. CUCB runs UCB with feasible NO selection decisions s and learns the expected utility for each 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.

- Random: The Random algorithm selects a subset of clients from ESs' communications coverage randomly in each edge aggregation round under these two constraints.

5 Results and analysis

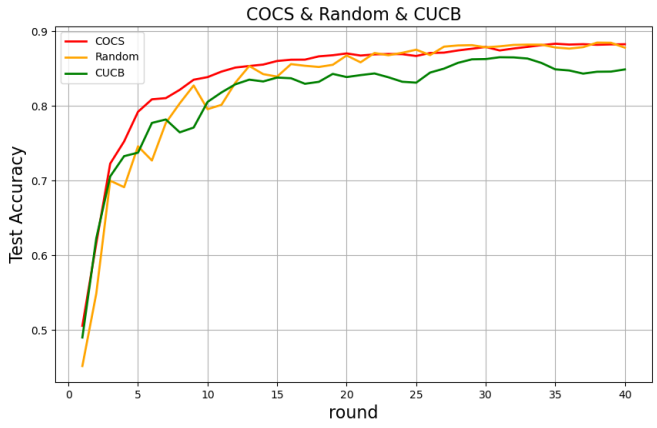

Figure 2. Training performance of 3 client selection policies based on logistic regression.

Fig.2 and Fig.3 show the model test accuracy for each global aggregation and the number of clients selected and successfully reached by all edge servers in each edge aggregation during the training process, respectively.

In Fig. 2, we can clearly see that the COCS algorithm converges faster than the randomized and benchmark CUCB algorithms. Since the CUCB algorithm is not able to adapt to changing context states, the convergence speed and test accuracy of CUCB are even worse than the results of the randomized algorithm. In contrast to

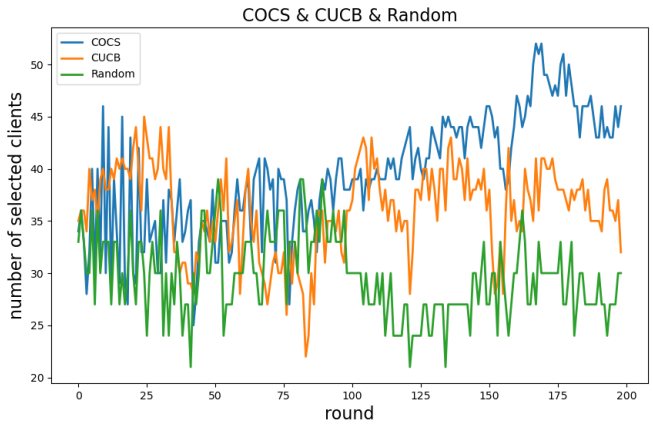

Figure 3. Temporal number of successful participated clients in each edge aggregation round.

the CUCB algorithm, the COCS algorithm is able to adapt to changing network conditions at the client and converges faster than the randomized algorithm.

In Fig. 3, we can see the total number of clients that can be selected and successfully upload model updates in each edge aggregation. the selection strategy made by the COCS algorithm in the early stage does not pull away from the CUCB algorithm and the random algorithm due to its cold start. In the later stages, the COCS algorithm is able to make better choices in conjunction with the context to select more clients that are potentially successful in connecting, and therefore it is able to connect to more clients than the random and CUCB algorithms.

6 Conclusion and future work

In this paper, I present the implementation and simulation results of the implemented COCS algorithm. The results show that the COCS algorithm is practical and converges faster than the benchmark. However, there is still room for the COCS algorithm to be improved, such as considering more network factors or non-network factors as the client's context. In the future, I will experiment with algorithmic improvements to COCS by adding, for example, the context of computational resources. Based on HFL with COCS, I will make my own work considering its shortcomings in privacy aspects and communication range coverage.

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. Vondrak, "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, "Communicationefficient 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. Vondrak, 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.