FedACS: An Adaptive Client Selection Framework for Communication-Efficient Federated Graph Learning

Hongli Xu, *Member, IEEE*, Xianjun Gao, Jianchun Liu, *Member, IEEE*, Qianpiao Ma, and Liusheng Huang, *Member, IEEE*

***Abstract*—Federated graph learning (FGL) has been proposed to collaboratively train the increasing graph data with graph neural networks (GNNs) in a recommendation system. Nevertheless, implementing an efficient recommendation system with FGL still faces two primary challenges, i.e., limited communication bandwidth and non-IID local graph data. Existing works typically reduce communication frequency or transmission amount, which may suffer significant performance degradation under non-IID settings. Furthermore, some researchers propose to share the underlying structure among clients, which brings massive communication cost. To this end, we propose an efficient FGL framework, named FedACS, which adaptively selects a subset of clients for model training, to alleviate communication overhead and non-IID issues simultaneously. In FedACS, the global GNN model learns significant hidden edges and the structure of graph data among selected clients, enhancing recommendation efficiency. This capability distinguishes it from the traditional FL client selection methods. To optimize the client selection process, we introduce a multi-armed bandit (MAB) based algorithm to select participating clients according to the resource budgets and the training performance (i.e., RMSE). Experimental results indicate that FedACS improves RMSE by 5.4% over baselines with the same resource budget and reduces communication costs by up to 70.7% to achieve the same RMSE performance.**

***Index Terms*—Federated graph learning, limited communication, non-IID graph data, client selection.**

I. Introduction

WITH the proliferation of applications, such as recommendation systems, the generation of graph data worldwide has increased significantly [1], [2]. To effectively utilize graph data while preserving privacy, federated graph learning (FGL) has been proposed for collaborative training of graph neural network (GNN) models on numerous edge clients [3]. The common FGL framework comprises a parameter server (PS) and clients [4]. To illustrate this, we consider an online shopping recommendation system. The data on each online store (i.e., client) include customers' social connections (e.g., following) and the customers' product ratings. Note that customer ratings of products only reflect the customers' preferences for the items and do not disclose any private information [5]. Customer ratings are public, while the social connections between customers are private. The graph they build is the data we need to protect for privacy purposes, and it is only used for local model training. In this scenario, customers and products are nodes, with product ratings and social connections serving as edges in the graph data. Initially, the online shopping company's server (i.e., PS) broadcasts a global GNN model to each client. Upon receiving the model, each client trains the model on its own local graph data and only uploads the updated GNN model to PS for the global model aggregation.

Distinct from traditional federated learning (FL) [6], [7], FGL focuses on the graph data. It capitalizes on the rich relational information in graphs, aggregating and utilizing the features of individual graph nodes and the connections between these nodes and their neighbors. For instance, in social networks, if two people share several mutual connections, they might also share similar interests or behaviors, such as shopping preferences. This nuanced approach gives FGL a clear advantage over traditional FL for handling extensive edge graph data. Besides, distributed graph data in FGL also exhibits some hidden edges [8], especially when it is implemented in recommendation systems. For instance, consider a scenario where two family members have built social connections on Amazon.[1] They may have similar shopping preferences because they both need to purchase household items. However, on eBay,[2] these family members may not have established a direct social connection. Relying solely on data from eBay, they may be considered completely irrelevant and lose the chance of being recommended. Since FGL can learn the structure information of the graph data, it may discover the hidden social connection on the data of eBay and make more

Received 4 June 2024; revised 3 March 2025; accepted 18 April 2025. Date of publication 22 April 2025; date of current version 3 September 2025. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62132019, in part by the Jiangsu Province Science Foundation for Youths under Grant BK20230275, in part by the Anhui Province Science Foundation for Youths under Grant 2408085QF185, and in part by the Fundamental Research Funds for the Central Universities under Grant WK2150110033. Recommended for acceptance by N. Shlezinger. *(Corresponding author: Hongli Xu.)*

Hongli Xu, Xianjun Gao, Jianchun Liu, and Liusheng Huang are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China, and also with the Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou 215123, China (e-mail: xuhongli@ustc.edu.cn; gaoxj@mail.ustc.edu.cn; jcliu17@ustc.edu.cn; lshuang@ustc.edu.cn).

Qianpiao Ma is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210000, China (e-mail: maqianpiao@njust.edu.cn).

Digital Object Identifier 10.1109/TMC.2025.3563404

[1]https://www.amazon.com
[2]https://www.ebay.com

accurate product recommendations after training on the data of Amazon, which cannot be achieved by the traditional FL.

Although FGL exhibits notable advantages over traditional FL in handling distributed graph data, its application to an efficient recommendation system still presents non-trivial challenges. With the rise in online shopping, the number of customers has surged (e.g., Amazon Prime members across the U.S. reached 168 million in June 2022[3]). To efficiently leverage the rising customer browsing data, the number of transferred model parameters has grown. For example, the Graph Attention Network (GAT) model has about 40% more parameters than the Graph Convolutional Network (GCN) model, due to the introduction of attention mechanisms [5]. Moreover, customers' devices are located in different local area networks (LANs), and communicate with PS over the wide area networks (WANs) [9]. Generally, the network bandwidth between clients and PS is much lower (e.g., $15\times$) than intra-datacenter LAN bandwidth [10]. Consequently, communication is likely to be the system's bottleneck because of the *increasing communication demand and limited network bandwidth*. An intuitive solution to alleviate communication overhead is to reduce communication frequency between PS and clients. Zhang et al. [11] propose to adopt stale updates to skip some aggregations, in which PS only communicates with the clients once in certain rounds. Yao et al. [4] argue that neighboring node information should be shared only once, thus slashing communication costs. Besides, reducing the number of parameters is another solution to alleviate communication pressure. Liu et al. [12] and Wu et al. [13] propose to only upload the model gradients to PS for aggregation. However, if the data distribution on each client is inconsistent, these approaches might lead to significant performance degradation [14].

In addition to communication pressure, implementing an efficient recommendation system with FGL also encounters another challenge of data skew in practice, i.e., *heterogeneous data distribution*. Due to varying customer preferences, the local data on each client may be not independent and identically distributed (non-IID) [15]. For example, families with children prefer toys while pregnant women lean towards baby products. Generally, training models on the non-IID data will reduce the training efficiency and decrease test accuracy (up to 55% [16]). Research efforts have also been made to address the non-IID issue in FGL. Xie et al. [17] dynamically cluster the clients based on the gradients of GNN and train distinct GNN models for homogeneous clusters. Tan et al. [15] propose to extract and share underlying structural information. They define structure embeddings and encode these embeddings with an independent structure encoder to capture more structure-based information. However, these methods necessitate active participation from all clients in every training round, substantially increasing the communication cost due to the frequent communication between a large number of clients and PS. In summary, the existing research still encounters difficulties in fully addressing the aforementioned challenges in FGL. To this end, we propose an FGL framework, called FedACS, which aims to reduce the communication cost and alleviate the non-IID problem simultaneously in FGL, for efficient recommendation systems. Specifically, FedACS augments the model training with transition probabilities between nodes to capture the graph's structural information. Then it adaptively selects an appropriate subset of clients to participate in both local model training and global model aggregation according to the resource budgets and training performance (i.e., RMSE) under different data distributions, rather than involving all clients simultaneously in each round, thereby curtailing the communication overhead and alleviating the influence of the non-IID issue. Although the client selection methods [18], [19] have been proposed in traditional FL, they may not be effective when directly implemented in FGL for the following reasons. First, these client selection methods in traditional FL typically overlook the nuanced complexities of graph structure information inherent in the graph data. This omission could result in missed opportunities to leverage the structure information for effective model training. Second, they do not account for the hidden edges among clients, potentially leading to performance loss if directly applied in FGL [8]. Nevertheless, *selecting the optimal clients for model training remains a challenge in FedACS, especially when considering hidden edges between overlapping nodes and varied data distributions among clients.* The main contributions are summarized as follows:

- We propose FedACS, an efficient FGL framework using an adaptive client selection method to tackle limited communication and the non-IID data challenges for an efficient recommendation system.
- We present a multi-armed bandit (MAB) based algorithm to adaptively select participating clients based on the resource budgets and RMSE performance.
- The experimental results demonstrate that FedACS can reduce the communication cost by approximately 70.7% to achieve the same target RMSE score, while improving the RMSE score by about 5.4% under the same resource budget, compared to baselines.

II. Preliminaries and Motivation

A. Federated Graph Learning (FGL)

The FGL system usually comprises a PS and a set of N clients, which collaboratively train a global GNN model. For ease of description, some key notations are listed in Table I. Each client i holds the local graph data $d_i := (\mathcal{V}_i, \mathcal{E}_i)$, where $\mathcal{V}_i$ is the node set and $\mathcal{E}_i$ represents the edge set, with each node and edge possessing distinct features. The local graph data d_i is the subgraph of a larger global graph D and satisfies: $D = d_1 \cup d_2 \cup \ldots \cup d_N$. Let n and n_i denote the total data sample size of all subgraphs and the sample size of the local graph data d_i respectively, where $n = \sum_{i=1}^{N} n_i$. In FGL, it is necessary to incorporate the aggregation of the features of nodes, neighbor nodes and edges during the local model training. This allows FGL to learn the graph structure information so as to perform classification or prediction tasks. For each round $t \in \{1, \ldots, T\}$, where T is the total number of training rounds, there are three main steps in FGL.

(1) *Global Model Distribution*: At the beginning of each round t, PS distributes the global model ω_t to all participating

[3]https://cirpamazon.substack.com

TABLE I
SYMBOL TABLE

Symbol	Semantics
N	the total number of clients
d_i	the local graph data of client i
T	the total number of training epochs until the training terminates
n'	the data sample size of all selected clients in a round
n_i	the data sample size of client i
$\omega_{t,i}$	the local model parameter of client i in round t
ω_t	the global model parameter of round t
V_t	the selected client set in round t
X_i^t	the selection status of client i in round t
b_i^t	the communication cost of client i in round t
$\widehat{R}_m$	the accumulated reward for each participation number m
$\overline{R}_i$	the accumulated reward for each client i
$\mathcal{F}_m^t$	the score of participation number m
$\mathcal{I}_i^t$	the score of client i

clients. This distribution ensures that each client begins their local training phase with the most up-to-date version of the global model. By doing so, the clients are able to train their individual models on their local datasets, while starting from a common, updated model base.

(2) *Local Model Updating*: Once receiving the global model ω_t, each client i will perform several local GNN model updates on its local graph data d_i. The training process of the GNN model mainly consists of two phases: *message passing* and *readout* [5]. To facilitate understanding, we take an L-layer GNN model as an example. In the *message passing* phase, for each node $u \in \mathcal{V}_i$ in the graph data d_i on client i, the GNN model aggregates the neighbor messages to generate the aggregated message $m_{i,u}^{l+1}$, where l represents the layer index. Then the model uses the aggregated message $m_{i,u}^{l+1}$ to update the hidden state $h_{i,u}^{l+1}$. Let $z_{u,v}$ represent the edge feature between nodes u and v. This phase can be formalized as follows:

$$m_{i,u}^{l+1} = \sum_{v\in\mathcal{N}_u} M(h_{i,u}^l, h_{i,v}^l, z_{u,v}) \tag{1}$$

$$h_{i,u}^{l+1} = U(h_{i,u}^l, m_{i,u}^{l+1}) \tag{2}$$

where $h_{i,u}^0 = x_{i,u}$ denotes the feature of node u on client i and $\mathcal{N}_u$ represents the neighbor set of node u. $M(\cdot)$ and $U(\cdot)$ are the message generation function and the state update function [5], respectively. In the *readout* phase, the model makes predictions by aggregating hidden states of nodes as follows:

$$r = R(\{h_{i,u}^L \mid u \in \mathcal{V}_i\}) \tag{3}$$

where $h_{i,u}^L$ is the hidden state of L layer of node u on client i. $\mathcal{V}_i$ is the node set in graph data d_i and $R(\cdot)$ produces a global representation from the hidden states of all nodes.

(3) *Global Model Aggregation*: At the end of round t, PS receives the local model $\omega_{t,i}$ from each client i. These local models have been independently trained on each client's respective dataset, incorporating local data characteristics and features. Then PS proceeds to aggregate these local models into an updated global model ω_{t+1} for the next round by weighted averaging, as follows:

$$\omega_{t+1} = \sum_{i=1}^{N} \frac{n_i}{n}\omega_{t,i} \tag{4}$$

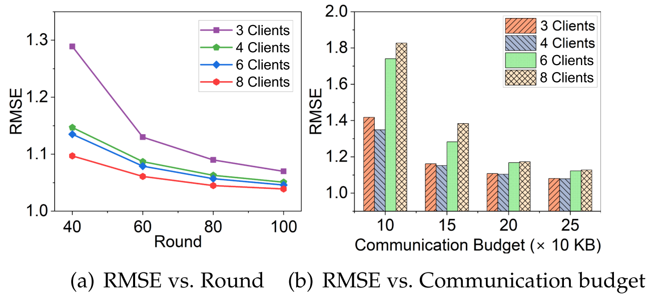


Fig. 1. Model training performance with varied participating clients.

B. Motivation for FedACS

In FGL, the variability of client participation significantly affects model training performance, which manifests in two primary ways. On one hand, the number of participating clients will substantially influence both training performance and resource consumption. On the other hand, with a constant number of participants, the specific choice of clients becomes paramount due to variances in data and differential hidden edges between clients.

To gain deeper insight into the influence of varying numbers of participating clients on model training, we conduct a set of experiments with 10 clients on the Ciao dataset (with 28 categories of items from the shopping website) [20]. We divide the Ciao dataset into 28 subgraphs based on product categories and distribute these subgraphs uniformly among 10 clients. Each client trains a two-layer GCN model [21] on their local graph. Specifically, we vary the number of participating clients as 3, 4, 6 and 8, respectively, and adopt the RMSE score as the evaluation metric, where a lower score indicates the superior performance of the model [12], [13]. As shown in Fig. 1(a), an increase in the number of clients generally enhances model training performance. However, it also escalates communication costs. Furthermore, in Fig. 1(b), we illustrate the RMSE score under the same communication budget. The results show that the scenario with 4 clients yields the optimal RMSE. This underscores the dilemma as follows: selecting fewer clients in each training round reduces resource overhead but also diminishes data richness, potentially leading to slower convergence. On the contrary, more participating clients mean more training data, which can accelerate model convergence, however, it also introduces higher communication costs [22], [23]. Thus, there is a trade-off between training performance and resource consumption in our proposed framework.

While understanding the effects associated with the number of clients is crucial, it is equally pertinent to assess the repercussions of data diversity among clients. Thus, we execute another

TABLE II
THE RMSE SCORE AND NODE INTERSECTION RATIO IN CHANGING THE LAST CLIENT AMONG FOUR CLIENTS

Scheme	A	B	C
Subgraphs	2	3	3
RMSE score	1.181	1.130	1.090
Node intersection	7.7%	10.0%	12.2%

A, B, C represent three clients selection schemes where the fourth client includes 2, 3, and 3 subgraphs.

experiment series on different client selection schemes, namely A, B and C. Each of the three schemes involves four clients. We keep the former three clients of these schemes unchanged and only alter the fourth client's data. Specifically, the diversity in the datasets is manipulated by varying the number of subgraphs each fourth client handles within their datasets. Under scheme A, the fourth client is provided with 2 subgraphs. In contrast, schemes B and C allocate 3 subgraphs each to their fourth clients. This setup permits a targeted reflection on how changes in the data complexity and volume at a single client influence the overall learning performance and model robustness within these distinct configurations. This experimental design allows us to gauge both the impact of differing subgraph counts and the nuances of varied data within the same subgraph counts.

Table II shows the RMSE score of model training under different client selection schemes. Generally, the model training performance of schemes B and C outperform that of scheme A by up to 7.7% because the model obtains more data from a greater number of subgraphs, accelerating model training. Notably, scheme C exhibits a model training performance improvement of 3.5% compared to scheme B. To understand this, we use N_t to denote the node union of the former three clients' graph data, and N_i to indicate the graph data node of the fourth client i. By evaluating the intersection ratio $|N_t \cap N_i|/|N_t|$ across all experiments, scheme C emerges with the highest node intersection rate and the best RMSE score, as shown in Table II. This indicates that a higher node intersection ratio, yielding more overlapping nodes, enables clients to contribute more hidden edges, catalyzing model convergence. In conclusion, both the quantity of participating clients and the intricacies of their data distribution profoundly affect model training and communication overheads in FGL. Strategically selecting the proper participating clients could reduce the communication cost while optimizing training performance for recommendation systems.

III. FRAMEWORK DESIGN

In this section, we propose the FedACS framework, which adaptively selects a proper subset of participating clients in each round. Different from the process of FGL, FedACS mainly consists of five steps, i.e., Client Selection, Global Model Distribution, Local Model Updating, Global Model Aggregation and Scoring, in a single global round. In each round t, FedACS first adaptively selects a set of participating clients V_t according to the proposed algorithm (Section V-C). Then, similar to FGL, PS distributes the latest global model ω_t to the client set V_t, after which each client performs the local model training. During local training on the client, in addition to the standard GNN training process, we augment each edge in the original graph data with the corresponding nodes' transition probabilities as supplementary features (obtained through initial data processing). The basic GNN training enables each node to reach any other node after a specified number of transitions (determined by the number of model layers), while the node transition probabilities assist the model in learning the likelihood of reaching particular nodes. The incorporation of these transition probabilities into the training process, combined with the standard GNN mechanism, allows the model to effectively capture the structural information of the graph data.

After several local model updates, each client $i \in V_t$ uploads its updated local model $\omega_{t,i}$ to PS, enabling PS to calculate the global model ω_{t+1} by the global model aggregation function. Finally, PS evaluates the global model ω_{t+1} on the test dataset and updates variables to determine the selected client set for the next round. The whole process will continue until the target accuracy is reached or the resource budget is exhausted.

The major differences between FGL and our proposed framework are evident in the processes of *Client Selection, Global Model Aggregation and Scoring*.

- In client selection, the PS selects a proper set of participating clients for model training. In general, involving more clients in training can accelerate convergence due to increased data diversity and richer gradient information, but also increases the data exchange between the clients and PS, resulting in higher communication consumption, and vice versa [22], [23]. Moreover, since the local model of each client should incorporate the edge information from local graph data, the hidden edges among clients will potentially influence the model to acquire graph structure information. Consequently, if these hidden edges cannot be adequately learned, it may lead to the global model training performance loss. Furthermore, given each client's distinct graph data, choosing various clients can produce varied effects on global model aggregation, especially when the data distribution on clients is non-IID. Thus, these data features require us to design a reasonable client selection strategy to ensure the model training performance.
- In terms of global model aggregation, since only a subset of the clients are selected to participate in the model training, it becomes necessary to modify the global model aggregation function in (4) as follows:

$$\omega_{t+1} = \sum_{i \in V_t} \frac{n_i}{n'} \omega_{t,i} \tag{5}$$

  where V_t denotes the set of selected clients in round t. Let n_i and $n' = \sum_{i \in V_t} n_i$ represent the data sample size of client i and the total data sample size of all selected clients, respectively. In doing so, we ensure that the aggregation accurately reflects the participation from the selected subset of clients. This provides a representative global model, capturing the variations in the data of selected clients.
- In the scoring process, PS evaluates the global model ω_{t+1} on the test dataset. Following this assessment, the

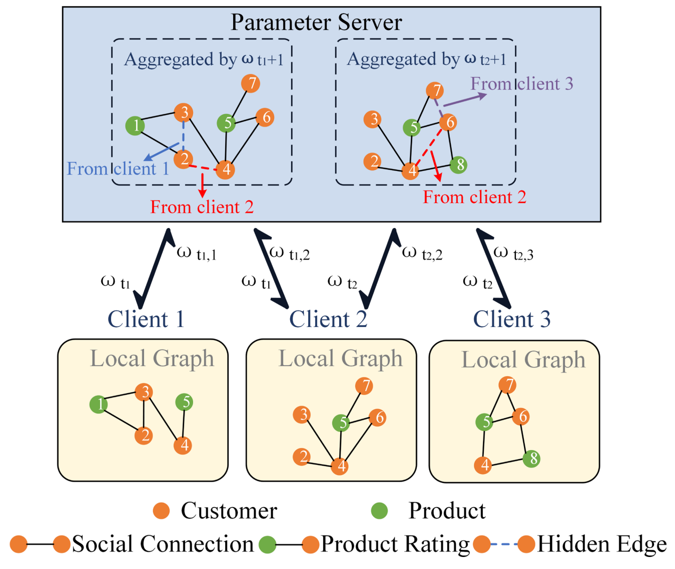


Fig. 2. Illustration of FedACS. PS selects clients 1 and 2 in round t_1, clients 2 and 3in round t_2, respectively. The global model ω_{t_1+1} integrates hidden edges (dotted lines) of nodes 2 and 3 from local model $\omega_{t_1,1}$, nodes 2 and 4 from model $\omega_{t_1,2}$ in round t_1, while the global model ω_{t_2+1} learns the edge information among nodes 4 and 6 from local model $\omega_{t_2,2}$, nodes 6 and 7 from model $\omega_{t_2,3}$ in round t_2.

PS updates several key variables, such as the count of the selected clients and the total communication cost incurred during the process. These updates are vital for maintaining a comprehensive understanding of the ongoing training dynamics. Then FedACS employs an efficient algorithm (Section V-C) to adaptively select clients for participation in subsequent training rounds. The algorithm meticulously calculates scores for each potential participant, taking into account various factors that influence their suitability and potential contribution to the model's training. Based on these scores, PS will choose the clients that are deemed to have the highest beneficial impact for the global model training as the selected clients for the next round. This process will continue and be updated based on the variables to adapt to the constantly changing system environment until the global model converges. By strategically choosing participants in this manner, FedACS can enhance the overall efficiency and effectiveness of the model training process.

For a better explanation of FedACS, we illustrate it in Fig. 2. Consider a system that consists of one parameter server and three clients, where the clients have overlapping graph nodes. Within client 1, nodes 2 and 3 are interconnected, whereas nodes 2 and 4 are not. Conversely, within client 2, nodes 2 and 4 share an edge. Using round t_1 in Fig. 2 as an illustrative case, PS initially selects clients 1 and 2 as V_{t_1} to participate in training. Then PS distributes the global model ω_{t_1} to V_{t_1} for the local GNN model training. Upon receiving the two local models $\omega_{t_1,1}$ and $\omega_{t_1,2}$, PS performs the model aggregation to obtain ω_{t_1+1}. It is important to highlight that the local model $\omega_{t_1,1}$ contains the edge information between nodes 2 and 3, while $\omega_{t_1,2}$ carries the edge information of nodes 2 and 4. Following the global model aggregation, the global model ω_{t_1+1} will learn the edge information. In the subsequent round t_2, the PS aggregates the local models from client 2 and client 3, enabling the global model ω_{t_2+1} to learn the edge information between neighbor nodes 4 and 6 from model $\omega_{t_2,2}$, and the edge information between nodes 6 and 7 from $\omega_{t_2,3}$. Through the above illustration, the selection of different client influences the hidden edges that the global model can learn, consequently affecting the training performance. By adaptively selecting clients, FedACS can effectively leverage the collaboration of multiple clients and their overlapping nodes to facilitate global model learning by adaptively selecting clients.

IV. Convergence Analysis

In this section, we analyze the convergence property of the proposed FedACS approximately. We rewrite the global model ω_t update function in (5) as follows:

$$\omega_{t+1} = \sum_{i\in V_t} \frac{n_i}{n'}\omega_{t,i} = \sum_{i\in V_t}\frac{n_i}{n'}\left(\omega_t - \eta\sum_{k=1}^{K} \nabla F_i(\omega_{t,i}^k; H_{t,i})\right) \tag{6}$$

where k represents the step of the model updating. We define $H_{t,i} = (x_{i,u}, x_{i,v}, z_{u,v}), u, v \in V_i, v \in \mathcal{N}_u$, as a mini-batch of samples graph data d_i. n_i represents the size of data samples while $\mathcal{N}_u$ is the neighbor set of node u. Let $\nabla F_i(\omega_{t,i}^k; H_{t,i}) = \frac{1}{n_i}\sum_{h_{t,i}\in H_{t,i}} \nabla f(\omega_{t,i}^k, h_{t,i})$, f is the loss function of the global GNN model with $h_{t,i}$ data samples. To analyze the convergence of FedACS, we assume that our problem satisfies the following assumptions, which have been adopted in previous works [24] on the convergence analysis.

Assumption 1: Each function $F_i(\omega, H_{t,i}), \forall i \in [N]$, has an L-lipschitz gradient with respect to ω, and $\nabla_\omega F_i(\omega, H_{t,i})$ is L_e-lipschitz with respect to $H_{t,i}$.

Assumption 2: All neighbors of a node are used for embedding generation, the gradient computed using the mini-batches w.r.t ω is an unbiased estimator of the true gradient and has a bounded variance at each client i. That is, $\mathbb{E}[\nabla F_i(\omega, H_{t,i})] = \nabla F_i(\omega, \mathcal{V}_i), \mathbb{E}[\|\nabla F_i(\omega, H_{t,i}) - \nabla F_k(\omega, \mathcal{V}_i)\|^2] \leq \delta^2, \forall i$.

Assumption 3: The average model gradient value of the selected clients V_t is higher than that across all clients. That is, $\sum_{i\in V_t}\frac{n_i}{n'}\sum_{k=1}^{K}\nabla F_i(\omega_{t,i}^k, H_{t,i}) \geq \sum_{i=1}^{N}\frac{n_i}{n}\sum_{k=1}^{K} \nabla F_i(\omega_{t,i}^k, H_{t,i})$.

To facilitate convergence analysis, we use an intermediate variable $\bar{\omega}_t = \sum_{i\in V_t}\frac{n_i}{n'}\omega_{t,i}$. So we have $\bar{\omega}_{t+1} = \bar{\omega}_t - \eta\sum_{i\in V_t}\frac{n_i}{n'}\sum_{k=1}^{K}\nabla F_i(\omega_{t,i}^k, H_{t,i})$. We use the expected gradient norm as the convergence metric that follows the convention in non-convex optimization [24].

Based on Assumption 1, we have:

$$\mathbb{E}[F(\bar{\omega}_{t+1})] \leq \mathbb{E}[F(\bar{\omega}_t)] + \mathbb{E}[\langle\nabla F(\bar{\omega}_t), \bar{\omega}_{t+1} - \bar{\omega}_t\rangle] + \frac{L}{2}\mathbb{E}[\|\bar{\omega}_{t+1} - \bar{\omega}_t\|^2] \tag{7}$$

To enhance the understanding of the convergence analysis, let $d_t = \sum_{i\in V_t}\frac{n_i}{n'}\sum_{k=1}^{K}\nabla F_i(\omega_{t,i}^k, H_{t,i})$. And according to the inner product formula $\langle a, b\rangle = \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2 - \frac{1}{2}\|a - b\|^2$ and the norm inequality $\|a+b\|^2 \le 2(\|a\|^2+\|b\|^2)$, we have the following result:

$$\mathbb{E}[F(\bar{\omega}_{t+1})] \le \mathbb{E}[F(\bar{\omega}_t)] + \left(L\eta^2 - \frac{\eta}{2}\right)\mathbb{E}[\|\nabla F(\bar{\omega}_t)\|^2] - \frac{\eta}{2}\mathbb{E}[\|d_t\|^2] + \left(L\eta^2 + \frac{\eta}{2}\right)\mathbb{E}[\|\nabla F(\bar{\omega}_t) - d_t\|^2] \tag{8}$$

Next, we discuss the upper bound of $\mathbb{E}[\|\nabla F(\bar{\omega}_t) - d_t\|^2]$. According to previous work [24], we have the following lemma:

Lemma 1:

$$\mathbb{E}[\|\nabla F(\bar{\omega}_t) - d_t\|^2] \le 3[\delta^2 + 2L^2\eta^2 \sum_{i\in N}\frac{n_i}{n}\mathbb{E}[\|\nabla F_i(\omega_t, H_{t,i})\|^2]] \tag{9}$$

According to (8), we have the following:

$$\left(\frac{\eta}{2} - L\eta^2\right)\mathbb{E}[\|\nabla F(\bar{\omega}_t)\|^2] \le (\mathbb{E}[F(\bar{\omega}_t)] - \mathbb{E}[F(\bar{\omega}_{t+1})]) - \frac{\eta}{2}\mathbb{E}[\|d_t\|^2] + \left(L\eta^2 + \frac{\eta}{2}\right)\mathbb{E}[\|\nabla F(\bar{\omega}_t) - d_t\|^2] \tag{10}$$

We use F' and T to denote the optimal value of function F and the number of total round, respectively. We incorporate the findings derived from Lemma 1 into (10) for further analysis:

$$\left(\frac{\eta}{2} - L\eta^2\right)\frac{\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\bar{\omega}_t)\|^2]}{T} \le \frac{F(\bar{\omega}_1) - F'}{T} - \frac{\eta}{2}\frac{\sum_{t=1}^{T}\mathbb{E}[d_t]}{T} + 3\left(L\eta^2 + \frac{\eta}{2}\right)\left(\delta^2 + 2L^2\eta^2\frac{\sum_{t=1}^{T}\mathbb{E}[\|d_t\|^2]}{T}\right) \tag{11}$$

Next, we discuss the value of learning rate η. First, in order to ensure the convergence of the formula, we set the condition: $(\frac{\eta}{2} - L\eta^2) \ge 0, \eta \le \frac{1}{2L}$. Next, to simplify the above result, we introduce another condition to eliminate the term: $\frac{\sum_{t=1}^{T}\mathbb{E}[\|d_t\|^2]}{T}$. This is achieved by setting: $3(L\eta^2 + \frac{\eta}{2})2L^2\eta^2 \le \frac{\eta}{2}$, which would be equivalent to solving the inequality $3(L\eta + \frac{1}{2})2L^2\eta^2 - \frac{1}{2} \le 0$. Upon differentiating this inequality with respect to η, we have $2L^2\eta(9L\eta + 1)$. This derivative is non-negative for all $\eta \ge 0$. When $\eta = 0$, the result would be $-\frac{1}{2}$. Therefore, there must be $\eta \ge 0$ to make the inequality establish. We use $\bar{\eta}$ to denote the maximum value of η which holds the inequality.

To sum up, when the learning rate $\eta \le min\{\frac{1}{2L}, \bar{\eta}\}$, we have

$$\frac{\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(\bar{\omega}_t)\|^2]}{T} \le \frac{F(\bar{\omega}_1) - F'}{T(\frac{\eta}{2} - L\eta^2)} + 3\frac{(L\eta^2 + \frac{\eta}{2})}{(\frac{\eta}{2} - L\eta^2)}\delta^2 \tag{12}$$

V. Algorithm Design

A. Problem Formulation

In FedACS, for a client i in round t, the communication cost b_i^t is divided into two distinct segments: the cost associated with receiving the global model from PS and the cost of uploading the updated local model to PS. The communication cost for individual clients remains consistent across rounds. This consistency stems from the fact that within this framework, clients only modify the model parameters without altering the foundational model structure [25]. In each round t, let X_i^t denote whether client i is selected to participate in the model training or not. For example, $X_i^t = 1$ indicates client i is selected while $X_i^t = 0$ indicates it is not. The number of participating clients is denoted by $\sum_{i=1}^{N} X_i^t$. To ensure a persistent involvement of clients across all training rounds, we set a minimum threshold L (e.g., 1) for the number of participating clients in each round. The overall communication cost for the whole process is given by $\sum_{t=1}^{T}\sum_{i=1}^{N} X_i^t \cdot b_i^t$. Concurrently, the total number of client selections throughout the process is represented by $\sum_{t=1}^{T}\sum_{i=1}^{N} X_i^t$. Given the underlying importance of retaining a rich diversity in client data to ensure robust learning of the graph structure, it is prudent to set a minimum limit for the total number of client selections [22], [26]. Let B and C represent the communication budget and the lower bound for the total number of selected clients during the model training, respectively. Accordingly, we formulate the client selection optimization problem in FedACS as follows:

$$\min_{\mathcal{X}} F(\omega^T)$$
$$\text{s.t.} \begin{cases} \mathcal{X} = \{X_i^t | X_i^t \in \{0,1\}, t \in [T], i \in [N]\} \\ \sum_{t=1}^{T}\sum_{i=1}^{N} X_i^t \cdot b_i^t \le B, \\ \sum_{t=1}^{T}\sum_{i=1}^{N} X_i^t \ge C, \\ \sum_{i=1}^{N} X_i^t \ge L, \quad \forall t \in [T] \end{cases} \tag{13}$$

where ω^T is the model in the last round T and $F(\omega) = \sum_{i\in V_t}\frac{n_i}{n'} \cdot X_i^t \cdot f^i(\omega)$ represents the loss value for the selected clients in each round t. V_t denotes the set of selected clients in round t and n' represents the total data sample size of all selected clients. $f^i(\cdot)$ is the loss function of client i and $\frac{n_i}{n'}$ is the ratio of the data sample size n_i of client i to the total data sample size n' of all selected clients. The first inequality expresses the communication resource constraints in the overall training process. The second inequality signifies the lower bound of total client selection in total T training rounds. The third set of inequalities ensures the progression of the training process by selecting at least L clients in each round t to maintain the updates and learning of the model. The objective of the problem is to minimize the loss function of FGL in the last round.

B. Multi-Armed Bandit

Due to the complex influence factors of FedACS (e.g., resource constraints, graph data), it is difficult to determine the selected clients before training. Therefore, we adopt a multi-armed bandit (MAB) based algorithm to select clients, which can reduce the communication cost and achieve a satisfactory model training performance. MAB is a classical problem in reinforcement learning (RL) [27] that faces the exploration versus exploitation dilemma, i.e., searching for a balance between exploring the environment to find profitable operations and exploiting more empirically best operations [26], [28]. MAB problem can be expressed as a tuple $(\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ is a set comprising k operations while $\mathcal{R}$ is the reward probability distribution. Each operation $a_i \in \mathcal{A}$ corresponds to a reward probability distribution $\mathcal{R}(r_i \mid a_i)$, from which a return r_i is obtained when performing the operation a_i. The goal of MAB is to maximize the return, i.e., $\max \sum_{t=1}^{T} r_t, r_t \sim \mathcal{R}(\cdot \mid a_t)$, where a_t and r_t denote the operation and reward in the tth operation, respectively.

In the FGL system, the process of client selection, model training performance, and the system environment are integral components that can be analogously related to the key elements of the MAB problem, namely operations, rewards, and the environment, respectively. Building upon these similarities as a foundation, an effective strategy can be adopted by FedACS, which begins with a few rounds of random client selection. This approach ensures that each client receives an initial opportunity to participate, allowing the system to gather initial data on the performance rewards associated with different clients (corresponding to the exploration of the MAB problem). Then, FedACS chooses the clients that are expected to yield the highest rewards based on the model training performances for each round (corresponding to the exploitation of the MAB problem). The decision-making process in FedACS is decided by an iterative update of the expected rewards, constantly refining its client selection strategy based on ongoing model training performance feedback.

C. Algorithm Description

In this section, we describe the client selection algorithm in FedACS in detail, referred to as ACS. The core concept of the ACS algorithm is to represent the client selection problem in FedACS as the MAB problem. Let V_t represent the selected clients in round t, which can be regarded as the operation in the MAB problem while the model training performance for each round can be correlated to the reward. However, a direct application of this approach faces a significant challenge due to the exponential increase in the potential combinations of client selections. With N clients, the number of possible combinations for V_t will reach an overwhelming $2^N - 1$. To make this problem tractable, the ACS algorithm should employ a two-step approach. First, the ACS algorithm determines the number of participating clients as M. Then, it selects clients with the top M scores as V_t in round t. This strategy narrows down the choices from $2^N - 1$ to a more feasible $2N$. We show the ACS algorithm in Algorithm 1, where the whole training process is divided into two stages: overall exploration (Lines 2-8) and directional selection (Lines 9-14).

Overall Exploration: In this stage, the ACS algorithm aims to undertake an initial exploration across all potentially selected clients and the number of participating clients. Initially, ACS selects all clients to participate in training, establishing a foundational reward for all clients (Lines 3-4). Then, we randomly determine the selected participation number M from {1,..., $N-1$} without repetition (Line 6) and select M clients randomly as V_t (Line 8) until round N. Each selected client $i \in V_t$ performs model training on its local graph data and uploads the updated local model $\omega_{t,i}$ to PS in round t. PS aggregates these local models to generate the global model ω_{t+1} by (5). Then, the algorithm calculates the rewards for each selected client i and participation number m in round t through the following formulas:

$$\widehat{R}_m = \widehat{R}_m + \left(1 - \frac{r_t}{Tg}\right) \cdot \log t - \frac{b_t}{B} \tag{14}$$

$$\overline{R}_i = \overline{R}_i + \left(1 - \frac{r_t}{Tg}\right) \cdot \log t \tag{15}$$

where $\widehat{R}_m$ represents the accumulated reward for each participation number m while $\overline{R}_i$ denotes the accumulated reward for each client i. r_t and Tg represent the model training performance on the test dataset (i.e., RMSE score) in round t and the expected result, respectively. Here, we adopt an indirect approach to evaluate the performance of structural information on each client, as raw structural information (e.g., node connection patterns and density) contains sensitive data privacy and is unsuitable as a direct selection criterion. The RMSE score is the performance result after model aggregation, which includes the combined structural information. Therefore, it can indirectly reflect the impact of structural information on performance. In FGL, the RMSE score drops rapidly at the start of training and then stabilizes, making it difficult to use the RMSE score directly as a consistent reward due to the conflicting nature between the variability of RMSE scores in graph data and our need for a consistent return. To ensure a smoother reward curve, we multiply the RMSE score by $\log t$, where t is the number of the current round, because the test RMSE score is negatively related to the training rounds in FGL. For calculating the reward of participation number, the ACS algorithm introduces additional terms for calculating communication cost, where b_t and B represent the communication cost in round t and the total communication budget. In this stage, we complete the record for all possible participation numbers and clients, which will be used to select the clients pertinently in the subsequent stage.

Directional Selection: At this stage, we select clients V_t adaptively by first determining the participating client number and then selecting the appropriate clients. Let $\mathcal{F}_m^t$ represent the score of participation number m and $\mathcal{I}_i^t$ denote the score of client i in round t, respectively. At the beginning of each round, these two scores will be calculated by the following formulas:

$$\mathcal{F}_m^t = \frac{\widehat{R}_m}{\widehat{N}_m^t} + \sqrt{2\frac{\log t}{\widehat{N}_m^t}} \tag{16}$$

$$\mathcal{I}_i^t = \frac{\overline{R}_i}{\overline{N}_i^t} + \sqrt{2\frac{\log t}{\overline{N}_i^t}} \tag{17}$$

where $\widehat{N}_m^t$ and $\overline{N}_i^t$ represent the selection count of each participation number m and each client i, respectively, up to round t. Let t represent the number of the current round. The rewards $\widehat{R}_m$ and $\overline{R}_i$ are calculated by (14) and (15). The whole formula consists of two parts: reward evaluation and selection balance. The reward evaluation gauges the average reward for each participation number m and each client i. The selection balance ensures that no client or participation number is neglected for

Algorithm 1: The ACS Algorithm in Round t.

```
Input: The accumulated reward: $\widehat{R}_m$, $\overline{R}_i$ and the selection
   count: $\widehat{N}^t_m$, $\overline{N}^t_i$ for each participation number $m$ and each
   client $i$; the number of clients: $N$
Output: The set of selected clients $V_t$
 1: Initialize the selected participation client number
    $M = 0, V_t = \emptyset$
 2: if round $t \leq N$ then
 3:     if round $t = 1$ then
 4:         $M = N$
 5:     else
 6:         Randomly select a number from {1,..., $N$-1}
            without repetition as $M$
 7:     end if
 8:     Select $M$ clients randomly as $V_t$
 9: else
10:     Calculate $\mathcal{F}^t_m$ with (16) for each participation
        number $m$
11:     Select the maximum value from $\mathcal{F}^t_m$ as $M$
12:     Calculate $\mathcal{I}^t_i$ with (17) for each client $i$
13:     Select the top $M$ clients from $\mathcal{I}^t_i$ as $V_t$
14:     end if
```

extended periods, thereby keeping continuous learning of each client. If a client or a participation number is not chosen for prolonged periods, the selection balance increases until it meets the selection criteria.

The ACS algorithm designates the $\mathcal{F}^t_m$ with the maximum score as the selected participation number M (Line 11), and selects the first M clients as V_t in round t according to $\mathcal{I}^t_i$ for each client (Lines 12-13). Then, PS distributes the global model ω_t to V_t for local training. Once the selected client $i \in V_t$ completes local training, it uploads the model $\omega_{t,i}$ to PS for aggregating the global model ω_{t+1} by (5). In the next round, rewards for each participation number and client will be updated, ensuring continuous refinement based on the evolving performance and contribution of each participation number and client. This process continues until the global model converges.

VI. Evaluation Preparation

A. Datasets and Models

Datasets and Distribution: We use two benchmark datasets, i.e., Ciao [20] and Epinions [29], which are commonly adopted in product recommendation. The Ciao and Epinions datasets consist of product categories and user ratings of products on shopping websites. These datasets include rating scores assigned by users to items, information regarding product categories, and trust data representing social connections between users. The Ciao dataset includes 7,317 users, 104,975 items (which are labeled in 28 categories), 283,320 ratings and 111,781 social connections. The Epinions dataset contains 18,069 users, 261,246 items (which are labeled in 27 categories), 762,938 ratings and 355,530 social connections [12]. In the experiments, we divide these datasets into several subgraphs based on the product categories (28 in Ciao and 27 in Epinions) and distribute them to each client. To test the model training performance on non-IID data, we set three different data distributions, i.e., 1) Average: each client is assigned 2-3 categories; 2) Skewed: each client is assigned 2-4 categories; and 3) Extreme: each client is assigned 1-5 categories. We adopt Average as the default data distribution.

Models: Two well-known graph neural network (GNN) models are implemented based on the aforementioned datasets: a five-layer GCN model [21] and a two-layer GAT model [30] on both Ciao and Epinions datasets. GCN aggregates the features of nodes, their neighbor nodes and edges for each node. GAT introduces the attention mechanism and assigns different weights to each neighbor node. We configured the GCN and GAT models with identical parameters: the hidden layer dimension is 32, the node embedding dimension is 64, and the GAT model employs a two-head architecture. Although GAT requires additional parameters for the attention mechanism, the two-layer GAT model has the same parameter size as the five-layer GCN model, with the same model parameter settings.

B. Evaluation Setup

Baselines: We adopt two types of baselines for comparison: traditional centralized algorithms and FL-based algorithms. Traditional centralized algorithms can be divided into the matrix factorization (MF) based algorithm [31], [32] and the GNN-based algorithm [33]. *SoRec* [31] factorizes the user-item interaction matrix and the user-user social network in parallel, employing the low-rank user latent feature space that reflects the social relationships between users. *SoReg* [32] models social network information as regularization terms to constrain the matrix factorization framework. *Consisrec* [33] samples consistent neighbors by relating sampling probability with consistency scores between neighbors and assigns consistent relations with high-importance factors for aggregation.

We also adopt seven FL-based algorithms as baselines, divided into two categories: full client participation [5], [12], [13] and client selection strategy [34], [35], [36], [37].

Full client participation: *FedGNN* [13] draws on the user-item graph inferred from the local user-item interaction data for the recommendation. *FeSoG* [12] adopts relational attention and aggregation to handle the data heterogeneity and infers users' hidden states by using local data. *FedGraphNN* [5] summarizes the training process of the GNN model, simplifying both the training and evaluation of the GNN model. Since FedGraphNN employs both the GCN model and the GAT model as baselines, we use FedGraphNN(GCN) and FedGraphNN(GAT) to represent the evaluation results of FedGraphNN using the GCN model and the GAT model respectively.

Client selection strategy: *POWER-OF-CHOICE* [34], denoted as POW selects clients with higher local losses to increase the rate of convergence. Random client selection strategy [35] ensures that each client has an equal probability, specifically 50%, of being selected for participation in each round of model training. *Oort* [36] combines the training time and the importance of the training samples as the utility function to determine

TABLE III
TRAINING PERFORMANCE AND RESOURCE COST FOR FEDACS AND FL-BASED BASELINES ON THE TWO DATASETS

Method	Ciao			Epinions		
	RMSE	Communication (KB)	Time (s)	RMSE	Communication (KB)	Time (s)
FedGNN	2.528	9996	6009	3.250	9996	15599
FeSoG	2.062	21990	12546	1.833	21990	32801
FedGraphNN(GAT)	1.055	851	90	1.219	851	382
FedGraphNN(GCN)	1.053	851	119	1.196	851	538
POW	1.064	690	294	1.184	690	703
Rand	1.049	760	191	1.178	702	627
Oort	1.051	673	173	1.209	605	895
Harmony	1.047	661	288	1.200	622	872
FedACS(GAT)	1.039	**332**	**74**	**1.169**	**447**	**285**
FedACS(GCN)	**1.039**	568	98	1.170	476	406

the subset of clients to train in each round. *Harmony* [37] takes into account both the data distribution and training capability of different clients when selecting clients.

Performance Metrics: We mainly employ the following metrics to evaluate the performance of different algorithms: (1) RMSE: It evaluates the difference between the predicted results and the actual data. The smaller the value is, the model training performance is better. (2) Time cost: We record the duration time from the beginning of model training to the end, encompassing the time for both training and transmission. (3) Communication cost: We record the bandwidth consumption throughout the training process, covering both the cost of the PS distributing the global model to clients and the clients uploading their local models to the PS.

Platform and Parameter Settings: Our experiments are conducted on a deep learning platform. This workstation is equipped with an Intel(R) Xeon(R) Gold 5218R, 8 NVIDIA GeForce RTX 3090Ti GPUs and 256 GB RAM. We use the hidden feature size of 32 and the node embedding dimension of 64 for GCN and GAT. Besides, SGD [38] is selected as the optimizer for GCN and GAT. The attention head of the GAT model is set to 2. We divide the dataset into a ratio of 8:1:1 for the training set, validation set and test set.

VII. EVALUATION RESULTS

A. Overall Performance Comparison

(1) *RMSE Performance:* We first present the RMSE score of FedACS and FL-based baselines on the two datasets in Table III. The results show that FedACS can reach the best RMSE score among all algorithms. Specifically, FedACS outperforms FedGNN in the RMSE score by 58.9% and 64.0% on the two datasets, respectively. Compared to FedGraphNN, FedACS also achieves an advantage of approximately 1.3%-4.1% in RMSE score with the GCN and GAT models, indicating superior model training performance of FedACS through adaptive client selection. Besides, compared to the baselines with client selection strategy, FedACS also demonstrates its own advantages. For instance, compared with POW, FedACS achieves the RMSE score advantage of approximately 2.3%-4.2% with the GCN and GAT models on the two datasets. This advantage is due to the fact that FedACS can approximate the differences among clients and select the appropriate clients through an evaluation method. It is worth noting that Oort and Harmony perform well on the Ciao dataset, but perform poorly on the Epinions dataset, including in terms of RMSE scores and completion time. This is because both client selection algorithms require additional computational resources to determine whether the data on different clients meet the selection criteria, and this computational cost increases with the scale and complexity of the graph data. However, this additional overhead neglects the structural information of the graph itself, which leads to a clear disadvantage compared to FedACS. Moreover, we also compare the RMSE score of FedACS with centralized baselines in Table IV. FedACS outperforms SoRec and SoReg. For instance, FedACS improves the RMSE score by at least 17.0% over SoRec because FedACS can obtain and utilize graph structure information effectively to boost the model training. Besides, the RMSE score of FedACS is only 6.2% and 10.1% lower than that of Consisrec on the two datasets, respectively. This can be attributed to FedACS enabling the global model to learn the hidden edges among clients through the adaptive client selection algorithm.

TABLE IV
RMSE FOR FEDACS AND CENTRALIZED BASELINES ON THE TWO DATASETS

Method	Ciao	Epinions
	RMSE	RMSE
SoRec	1.252	1.482
SoReg	1.233	1.480
Consisrec	0.978	1.062
FedACS(GAT)	1.039	1.169
FedACS(GCN)	1.039	1.170

(2) *Resource Consumption:* To evaluate the resource efficiency of FedACS, we also compare the communication and

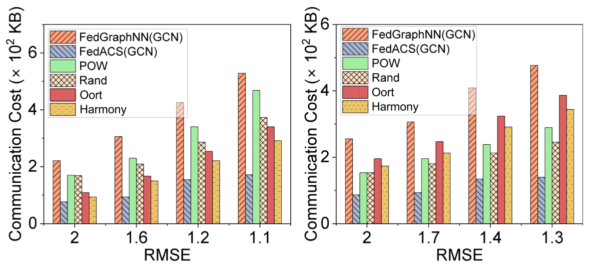


Fig. 3. Communication cost to reach different target RMSE scores on the two datasets.

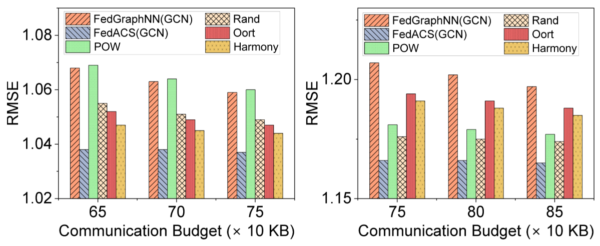


Fig. 4. RMSE for FedACS and baselines on the two datasets under different communication budgets.

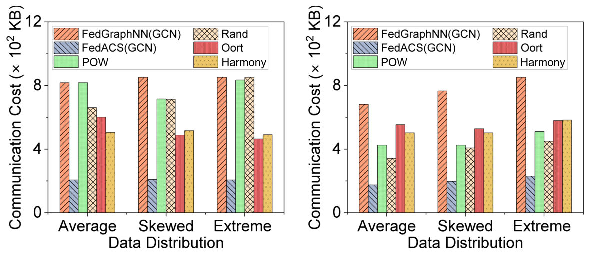


Fig. 5. Communication cost to reach different target RMSE scores (Ciao: 1.055, 1.071 and 1.041; Epinions: 1.219, 1.222 and 1.190) on the two datasets under different data distributions.

time costs for FedACS and baselines. Due to inherent differences among algorithms relying on traditional centralized data storage systems and those predicated on FL-based systems, our comparison is narrowed down to only FL-based baselines (e.g., FedGNN, FedGraphNN and POW). The final experimental results are shown in Table III. It is obvious that FedACS achieves the best performance among baselines in terms of both time and communication costs. For instance, compared with FedGraphNN(GAT), FedACS(GAT) saves 61.0% communication bandwidth and 17.8% time on the Ciao dataset. Besides, on the Epinions dataset, FedACS(GCN) only takes 476 KB of bandwidth and 406 s of completion time for training while Harmony requires 622 KB and 872 s, respectively. Compared to other client selection methods, FedACS also has advantages. For example, compared with Rand, FedACS(GAT) saves 56.3% communication bandwidth and 50.8% time on the Ciao dataset. The reason lies in that FedACS adaptively selects proper participating clients rather than taking all clients into model training, which can reduce the communication cost and the training time in each round.

Due to poor model training performance of FedGNN and FeSoG, we choose FedGraphNN, POW, Rand, Oort and Harmony to conduct fair evaluation comparisons with FedACS under different constraints. Fig. 3 depicts the communication cost of FedACS and baselines as they attain different target RMSE scores on the two datasets. These results reveal that FedACS outperforms the baselines in terms of the communication cost to attain the same RMSE score (e.g., 2, 1.6, 1.2 and 1.1 on Ciao), as FedACS effectively selects the clients that contribute more to training. As illustrated in Fig. 3(a), FedACS(GCN) requires a bandwidth cost of 172 KB to achieve the RMSE score of 1.1 on the Ciao dataset, while Rand and Oort consume 373 KB and 340 KB, respectively. Besides, by Fig. 3(b), FedACS(GCN) reduces the communication cost by 70.7% compared to FedGraphNN(GCN), when attaining the same RMSE score of 1.3 on the Epinions dataset. These results demonstrate the superiority of FedACS in terms of communication cost during the training process.

As shown in Fig. 4, we also measure the RMSE score of FedACS and baselines within the given communication budgets (e.g., 650 KB, 700 KB and 750 KB for Ciao) on the two datasets. Impressively, FedACS consistently surpasses baselines on both the Ciao and Epinions datasets. For instance, in Fig. 4(b), FedACS(GCN) outperforms FedGraphNN(GCN) in the RMSE score by about 3.5% with the communication budget of 750 KB on the Epinions dataset. Compared with Oort, FedACS also achieves improvements of 2.3%. Besides, by Fig. 4(a), FedACS(GCN) achieves an average advantage of 2.9% compared to FedGraphNN(GCN) and POW, when both methods consume 650 KB of communication bandwidth on the Ciao dataset. This is because FedACS evaluates each client's contribution to model training by analyzing the quality, relevance, and potential impact of their data on model accuracy. Then FedACS adaptively selects those that yield higher returns, rather than selecting all clients. This strategy enables FedACS to selectively incorporate updates from clients that significantly enhance the model's performance.

B. *Effect of Different Data Distributions*

Considering that clients in FGL collect data from their physical locations directly, the graph data among clients are usually non-IID. We discuss the impact of non-IID data on training performance. We use three different data distributions on the Ciao and Epinions datasets, i.e., Average, Skewed and Extreme. We measure the communication cost under varying data distributions, when FedACS and the baselines attain different target RMSE scores (e.g., 1.055, 1.071 and 1.041 on Ciao under three data distributions). Since the achievable RMSE scores for FedACS and baselines vary under different data distributions, we set the minimal target RMSE scores by all schemes. The results in Fig. 5 reveal that FedACS outperforms the baselines in terms of communication cost to attain the target RMSE score. For instance, as illustrated in Fig. 5(a), FedACS(GCN) takes a

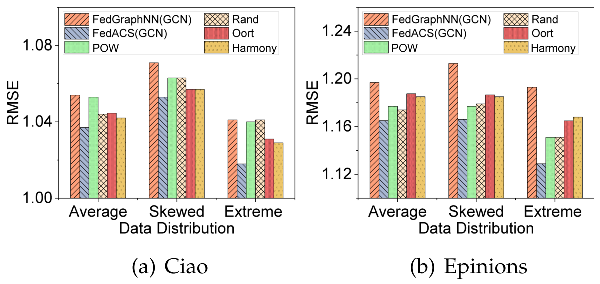


Fig. 6. RMSE for FedACS and baselines on the two datasets wit given communication budget (850 KB) under different data distributions.

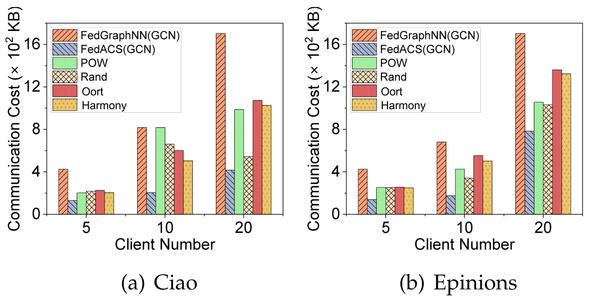


Fig. 7. Communication cost to reach different target RMSE scores (Ciao: 1.058, 1.055 and 1.012; Epinions: 1.182, 1.219 and 1.230) on the two datasets under different numbers of participating clients.

bandwidth cost of 209 KB to achieve the RMSE score of 1.071 on the Ciao dataset under the Skewed data distribution, while FedGraphNN(GCN), POW and Rand take 851 KB, 715 KB and 713 KB, respectively. On the Epinions dataset, as depicted in Fig. 5(b), FedACS(GCN) reduces the communication cost by 73.0% compared to FedGraphNN(GCN) when attaining the RMSE score of 1.190 under the Extreme data distribution. Compared to Oort and Harmony, FedACS(GCN) has also achieved an average communication performance advantage of 62.1% when attaining the same RMSE score of 1.222 under the Skewed data distribution.

Additionally, we record the RMSE score of FedACS and baselines within the given communication budget of 850 KB on the two datasets in Fig. 6. As the skewness of data distribution amplifies, the RMSE score of all systems initially worsens, and then gets better. This initial decline in performance can be directly attributed to the significant discrepancy in data distribution, resulting in poor performance. However, as the skewness becomes more pronounced, the data starts to aggregate and becomes increasingly concentrated on certain clients. Specifically, fewer than 40% of clients possess about 75% of data. The global aggregation, which uses data size as the aggregation parameter, will minimize the differences from clients with minimal data. In addition to these overall insights into different data distributions, we now turn our attention to a direct comparison of the RMSE score between FedACS and baselines. Similar to the Average data distribution, FedACS always achieves superior RMSE scores compared to baselines under both Skewed and Extreme data distributions. For instance, as illustrated in Fig. 6(b), FedACS(GCN) outperforms FedGraphNN(GCN) in terms of the RMSE score by 5.4% under the Extreme data distribution on the Epinions dataset. In Fig. 6(a), on the Ciao dataset, under the Extreme data distribution and with the communication budget of 850 KB, FedACS(GCN) achieves an average performance advantage of 2.2% over FedGraphNN(GCN) and Rand. The above experimental results demonstrate the superiority of FedACS in terms of communication cost and alleviating the non-IID issue.

C. Effect of the Number of Clients

We evaluate the performance of FedACS and baselines with three different numbers of participating clients, namely 5, 10, and 20. In this set of experiments, data is evenly distributed among the clients. We measure the communication cost with different numbers of participating clients, when FedACS and baselines attain different target RMSE scores (e.g., 1.182, 1.219 and 1.230 on the Epinions dataset). The experimental results in Fig. 7 show that FedACS outperforms the baselines when changing the number of participating clients. For instance, in Fig. 7(a), FedACS(GCN) only takes 417 KB of bandwidth cost to achieve the RMSE score of 1.012 on the Ciao dataset with 20 participating clients, while POW, Rand, Oort and Harmony take 986 KB, 545 KB, 1073 KB and 1025 KB, respectively. Given 5 participating clients, as depicted in Fig. 7(b), FedACS(GCN) reduces the communication cost by 67.2% compared to FedGraphNN(GCN) when attaining the same RMSE score of 1.182 on the Epinions dataset. Additionally, comparing FedACS(GCN) to Oort and Harmony, it shows an average communication performance improvement of 41.6% while reaching the same RMSE score of 1.230 with 20 participating clients.

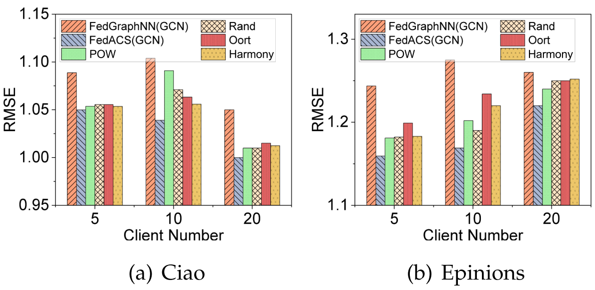


Fig. 8. RMSE for FedACS and baselines on the two datasets with communication budget (250 KB, 500 KB and 1,000 KB) under different numbers of participating clients.

Furthermore, we record the RMSE score of FedACS and baselines on the two datasets within the given communication budgets. Since the number of participating clients changes in this set of experiments, we track the RMSE score under various communication budgets for different numbers of participating clients. Specifically, we document the performance of different methods with communication overhead of 250 KB, 500 KB and 1000 KB for 5, 10 and 20 participating clients, respectively. The experimental results are presented in Fig. 8. In general, FedACS always achieves the best performance under different client participation numbers. For instance, on the Ciao dataset, given 10 participating clients and a communication budget of 500 KB, the

TABLE V
RMSE SCORE OF CS AND ND

Method	Ciao	Epinions
	RMSE	RMSE
FedACS	1.039	1.170
ND	1.057	1.185
CS	1.050	1.180

RMSE score of FedACS(GCN) is 1.039, as shown in Fig. 8(a). In comparison, the RMSE scores for FedGraphNN(GCN), POW, Rand, Oort and Harmony are 1.104, 1.091, 1.071, 1.063 and 1.056, respectively. In Fig. 8(b), FedACS(GCN) outperforms FedGraphNN(GCN) in the RMSE score by 6.8% with 5 participating clients. Additionally, compared to POW and Rand, FedACS(GCN) also has an average performance advantage of 1.9%. The experimental results above clearly illustrate that FedACS excels in lowering communication costs throughout the training process.

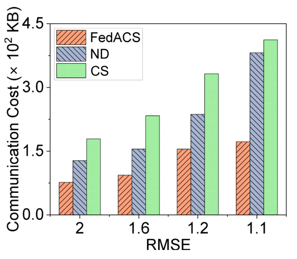
D. Component-Wise Evaluation

Finally, we evaluate the effectiveness of different components of FedACS. Specifically, the client selection algorithm in FedACS is designed by first determining the number of selected clients and then selecting the corresponding number of clients based on their scores. Therefore, we implement two segmented versions of FedACS to evaluate and understand the effectiveness of these two key components in FedACS.

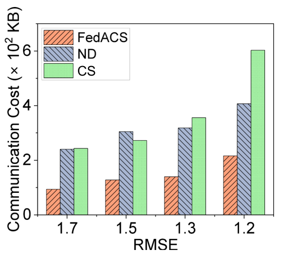
(1) *FedACS w/o score-based client selection (ND):* We disable the score-based client selection component. In each round, the PS decides the number of participating clients according to our proposed algorithm in Section V-C, but randomly selects the corresponding number of participating clients. Therefore, FedACS may select the client with low score, resulting in slow model convergence.

(2) *FedACS w/o participation number determination (CS):* We disable the part for determining the number of participating clients. In each round, the PS randomly determines the number of participating clients in the current round, and then selects the corresponding number of participating clients according to their score. As such, FedACS will cause an unreasonable number of selected clients, wasting precious communication traffic.

FedACS achieves a balance between system communication efficiency and RMSE score performance through an adaptive client selection strategy: We show the RMSE score for ND and CS when the model reaches convergence in Table V. Overall, CS facilitates model training better than ND. For instance, CS achieves an advantage of 0.7% in the RMSE score on the Ciao dataset in Table V, indicating that CS is more beneficial for model training. Besides, FedACS achieves a 1.7% advantage in the RMSE score over ND on the Ciao dataset. In addition, we also evaluate the communication cost of ND and CS when they achieve different RMSE scores, as illustrated in Fig. 9. It is evident that the component of participation number determination (ND) significantly reduces communication costs. For example,



(a) Ciao



(b) Epinions

Fig. 9. Communication cost to reach different target RMSE scores of CS and ND.

as illustrated in Fig. 9(a), ND takes 236 KB of bandwidth cost to achieve the RMSE score of 1.2 on the Ciao dataset, while CS requires 333 KB. By Fig. 9(b), ND reduces the communication cost by 32.4% compared to CS when achieving the same RMSE score of 1.1 on the Epinions dataset. Therefore, the experimental results demonstrate that the two proposed components can separately contribute to optimizing model performance and reducing communication overhead, highlighting the necessity and importance of both components.

VIII. RELATED WORK

Federated Graph Learning (FGL) is an emerging approach that combines the principles of federated learning and graph neural networks. This technique is designed to handle graph-structured data across multiple decentralized entities while maintaining privacy and security. In FGL, data remains at its source, and only model updates are exchanged between participants. This allows for collaborative learning without direct sharing of sensitive information. FGL is particularly useful in scenarios where data is naturally structured in graphs, such as social networks, communication networks, and recommendation systems, providing a way to leverage relational information while respecting data privacy constraints.

However, due to the limited communication capabilities of communication equipment and the growing contradiction with the increasing communication demands, communication issues have gradually become a bottleneck in realizing the FGL system. A related research area to FGL focuses on communication resource efficiency. Ghari et al. [39] propose a framework to learn a set of pre-trained models. At each round, PS selects a subset of pre-trained models based on the structure of a graph and only distributes the model that satisfies the resource constraints to clients. Zhang et al. [11] propose to adopt a client's up-to-date node representations and other clients' stale node representations to perform model updates in every round. Pan et al. [40] propose to adopt a hierarchical clustering tree constructor to extract the feature of each node and use a random walk generator to update the node's embedding. Wu et al. [13] design a novel federated framework named FedGNN, which collectively trains GNN models from decentralized user data and only uploads the local gradients of GNN to a server for aggregation. While Lie et al. [12] adopt relational attention and aggregation to handle graph data. While the above works mainly focus on the communication efficiency of FGL, in scenarios where the data distribution of client is non-IID, these approaches might lead to significant performance degradation.

To address the non-IID problem, some studies have made efforts. Zhang et al. [41] generate graph embeddings from the client's local graph dataset and integrate information from other clients. They also adopt the embedding alignment technique to convert the latent representations from other clients into a form that the local client can understand. Xie et al. [17] introduce a framework for graph clustered federated learning that dynamically identifies clusters among local systems by analyzing the gradients from graph neural networks, and theoretically justify that the heterogeneity among graphs can be reduced by the framework they proposed. Wu et al. [42] propose to alleviate the non-IID problem by transfer learning. They design a novel graph subtree discrepancy to measure the graph distribution shift between different graphs, and minimize the distribution shift between source and target graphs for cross-network transfer learning. Tan et al. [15] propose to share the underlying structure information of graph data and capture more structure-based information by structure embeddings. However, these approaches require the active participation of all clients, resulting in significant communication cost. Despite the advances and extensive research endeavors in the realm of FGL, the current research has yet to fully tackle the aforementioned challenges in FGL. To this end, we propose FedACS, which can reduce the communication cost and alleviate the non-IID problem simultaneously.

IX. Conclusion

In this work, we propose the FedACS framework, which adaptively selects clients to address the challenges of limited resources and the non-IID problem raised by FGL, for efficient recommendation systems. It employs an MAB-based online learning algorithm to adaptively determine the number of participating clients and select the clients that meet the criteria in each round. The dynamic optimization of client selection can contribute to the balance between the global model training performance and resource efficiency. The extensive experimental results demonstrate that FedACS significantly outperforms the existing baselines.

References

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, 2019.

[2] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.

[3] Z. Li, M. Bilal, X. Xu, J. Jiang, and Y. Cui, "Federated learning-based cross-enterprise recommendation with graph neural networks," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 673–682, Jan. 2023.

[4] Y. Yao, W. Jin, S. Ravi, and C. Joe-Wong, "FedGCN: Convergence and communication tradeoffs in federated training of graph convolutional networks," 2022, *arXiv:2201.12433*.

[5] C. He et al., "FedGraphNN: A federated learning benchmark system for graph neural networks," in *Proc. ICLR 2021 Workshop Distrib. Private Mach. Learn.*, 2021.

[6] J. Yan, J. Liu, H. Xu, Z. Wang, and C. Qiao, "Peaches: Personalized federated learning with neural architecture search in edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 11, pp. 10296–10312, Nov. 2024.

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

[8] J. Baek, W. Jeong, J. Jin, J. Yoon, and S. J. Hwang, "Personalized subgraph federated learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 1396–1415.

[9] Z. Yao, J. Liu, H. Xu, L. Wang, C. Qian, and Y. Liao, "Ferrari: A personalized federated learning framework for heterogeneous edge clients," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 10031–10045, Oct. 2024.

[10] J. Huang et al., "PhyFinAtt: An undetectable attack framework against PHY layer fingerprint-based WiFi authentication," *IEEE Trans. Mobile Comput.*, vol. 23, no. 7, pp. 7753–7770, Jul. 2024.

[11] X. Zhang, M. Hong, and J. Chen, "GLASU: A communication-efficient algorithm for federated learning with vertically distributed graph data," 2023, *arXiv:2303.09531*.

[12] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–24, 2022.

[13] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "FedGNN: Federated graph neural network for privacy-preserving recommendation," 2021, *arXiv:2102.04925*.

[14] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 4387–4398.

[15] Y. Tan, Y. Liu, G. Long, J. Jiang, Q. Lu, and C. Zhang, "Federated learning on non-IID graphs via structural knowledge sharing," 2022, *arXiv:2211.13009*.

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

[17] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-IID graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 18839–18852.

[18] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-IID data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24462–24474, 2021.

[19] Y. Zhang, C. Xu, H. H. Yang, X. Wang, and T. Q. Quek, "DPP-based client selection for federated learning with non-IID data," in *Proc. 2023 IEEE Int. Conf. Acoust. Speech Signal Process.*, 2023, pp. 1–5.

[20] J. Tang, H. Gao, and H. Liu, "mTrust: Discerning multi-faceted trust in a connected world," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 93–102.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[22] L. Wang, Y. Xu, H. Xu, J. Liu, Z. Wang, and L. Huang, "Enhancing federated learning with in-cloud unlabeled data," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 136–149.

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

[24] B. Du and C. Wu, "Federated graph learning with periodic neighbour sampling," in *Proc. IEEE/ACM 30th Int. Symp. Qual. Service*, 2022, pp. 1–10.

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

[26] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235–256, 2002.

[27] C. Dann, T. Lattimore, and E. Brunskill, "Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5717–5727.

[28] S. Bubeck et al., "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012.

[29] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the Semantic Web," in *Proc. Int. Semantic Web Conf.*, Springer, 2003, pp. 351–368.

[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv: 1710.10903*.

[31] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 931–940.

[32] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, “Recommender systems with social regularization,” in *Proc. 4th ACM Int. Conf. Web search Data Mining*, 2011, pp. 287–296.
[33] L. Yang, Z. Liu, Y. Dou, J. Ma, and P. S. Yu, “ConsisRec: Enhancing GNN for social recommendation via consistent neighbor aggregation,” in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 2141–2145.
[34] Y. J. Cho, J. Wang, and G. Joshi, “Towards understanding biased client selection in federated learning,” in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2022, pp. 10351–10375.
[35] M. Tang et al., “FedCor: Correlation-based active client selection strategy for heterogeneous federated learning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10102–10111.
[36] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, “Oort: Efficient federated learning via guided participant selection,” in *Proc. 15th USENIX Symp. Operating Syst. Des. Implementation*, 2021, pp. 19–35.
[37] C. Tian, L. Li, Z. Shi, J. Wang, and C. Xu, “HARMONY: Heterogeneity-aware hierarchical management for federated learning system,” in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture*, 2022, pp. 631–645.
[38] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proc. 19th Int. Conf. Comput. Statist.*, Paris France, Springer, 2010, pp. 177–186.
[39] P. M. Ghari and Y. Shen, “Graph-assisted communication-efficient ensemble federated learning,” in *Proc. 30th Eur. Signal Process. Conf.*, 2022, pp. 737–741.
[40] Q. Pan and Y. Zhu, “FedWalk: Communication efficient federated unsupervised node embedding with differential privacy,” in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 1317–1326.
[41] K. Zhang et al., “Privacy-preserving federated graph neural network learning on non-IID graph data,” *Wireless Commun. Mobile Comput.*, vol. 2023, 2023, Art. no. 8545101.
[42] J. Wu, J. He, and E. Ainsworth, “Non-IID transfer learning on graphs,” in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 10342–10350.

Hongli Xu (Member, IEEE) received the BS degree in computer science from the University of Science and Technology of China, China, in 2002, and the PhD degree in computer software and theory from the University of Science and Technology of China, China, in 2007. He is a professor with the School of Computer Science and Technology, University of Science and Technology of China (USTC), China. He was awarded the Outstanding Youth Science Foundation of NSFC, in 2018. He has won the best paper award or the best paper candidate in several famous conferences. He has published more than 100 papers in famous journals and conferences, including *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, Infocom and ICNP, etc. He has also held more than 30 patents. His main research interest is software defined networks, edge computing and Internet of Thing.

Xianjun Gao received the BS degree from Jilin University, in 2022. He is currently working toward the master’s degree with the School of Computer Science, University of Science and Technology of China (USTC). His main research interests are edge computing and federated learning.

Jianchun Liu (Member, IEEE) received the PhD degree from the School of Data Science, University of Science and Technology of China, in 2022. He is currently an associate researcher with the School of Computer Science and Technology, University of Science and Technology of China. His main research interests are software defined networks, network function virtualization, edge computing and federated learning. He is a member of ACM.

Qianpiao Ma received the BS degree in computer science and technology and the PhD degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2014 and 2022, respectively. He is currently an associate professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His primary research interests include federated learning, edge computing, and distributed machine learning.

Liusheng Huang (Member, IEEE) received the MS degree in computer science from the University of Science and Technology of China, in 1988. He is currently a senior professor and a PhD supervisor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or co-authored six books and more than 300 journal/conference papers. His research interests are in the areas of the Internet of Things, vehicular Ad-Hoc networks, information security, and distributed computing.