# Towards Communication-Efficient Federated Graph Learning: An Adaptive Client Selection Perspective

Xianjun Gao[1,2], *Jianchun Liu[1,2], Hongli Xu[1,2], Qianpiao Ma[3], Lun Wang[1,2]

[1]School of Computer Science and Technology, University of Science and Technology of China
[2]Suzhou Institute for Advanced Research, University of Science and Technology of China
[3]Purple Mountain Laboratories

*Abstract*—**Federated graph learning (FGL) has been proposed to collaboratively train the increasing graph data with graph neural networks (GNNs) in a recommendation system, aggregating the features of graph nodes and edges among these nodes. 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 information among all 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 can learn 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) under different data distributions. Experimental results show that, given the same resource budget, FedACS achieves the RMSE improvement of 5.4% over the baselines. Besides, when achieving the same RMSE performance, FedACS saves up to approximately 70.7% communication cost, compared with the baselines.**

*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]. 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 [2]. The common FGL framework comprises a parameter server (PS) and clients [3]. 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 these are not customer shopping records, but rather customer ratings of the products, such as liking or disliking, which does not reveal any private information [4]. 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) [5], 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 [6], 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 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 [4]. Moreover, customers' devices are located in different local area networks (LANs), and communicate with PS over the wide area networks (WANs) [7]. Generally, the network bandwidth between clients and PS is much lower (*e.g.*, $15\times$) than intra-datacenter LAN bandwidth [8]. Consequently, communication is likely to be the system's bottleneck because of the *increasing*

---

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

*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.* [9] propose to adopt stale updates to skip some aggregations, in which PS only communicates with the clients once in certain rounds. Yao *et al.* [3] 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.* [10] and Wu *et al.* [11] 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.

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) [12]. 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% [13]). Research efforts have also been made to address the non-IID issue in FGL. Xie *et al.* [14] dynamically cluster the clients based on the gradients of GNN and train distinct GNN models for homogeneous clusters. Tan *et al.* [12] 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 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 [15], [16] 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 [6]. 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. Each client $i$ holds 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: $D = d_1 \cup d_2 \cup ... \cup d_N$. Let $n$ and $n_i$ denote the total data sample size of all subgraphs and the sample size of the 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 local model training. This allows FGL to learn the graph structure information so as to perform classification or prediction tasks. For each round $t \in \{1, ..., T\}$, 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 $\omega_t$ to clients, ensuring they start the local training with the latest global model.

(2) **Local Model Updating**: Once receiving the global model $\omega_t$, 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* [4]. 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
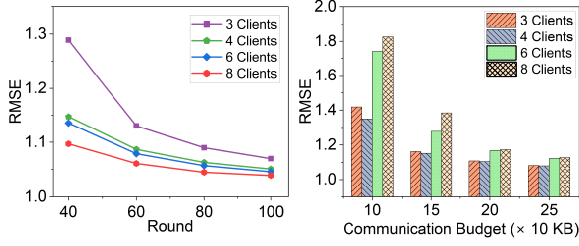
(a) RMSE vs. Round     (b) RMSE vs. Communication budget

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

function [4], 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 $i$ and aggregates the global model $\omega_{t+1}$ by weighted averaging, as follows:

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

### 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) [17]. 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 [18] 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 [10], [11]. 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 [19].

TABLE I: The RMSE score and node intersection ratio in changing the last client among four clients. A, B, C represent three clients selection schemes where the fourth client includes 2, 3, and 3 subgraphs.

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

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 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 fourth client receives 2, 3, and 3 subgraphs under schemes A, B, and C, respectively. 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 I 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 I. 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 $\omega_t$ to the client set $V_t$, after which each client performs the local model training. 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 $\omega_{t+1}$ by the global model aggregation function. Finally, PS evaluates the global model $\omega_{t+1}$ 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 [19]. 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 Eq. (4) as follows:

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

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 $\omega_{t+1}$ on the test dataset. Following this assessment, the 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 a 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
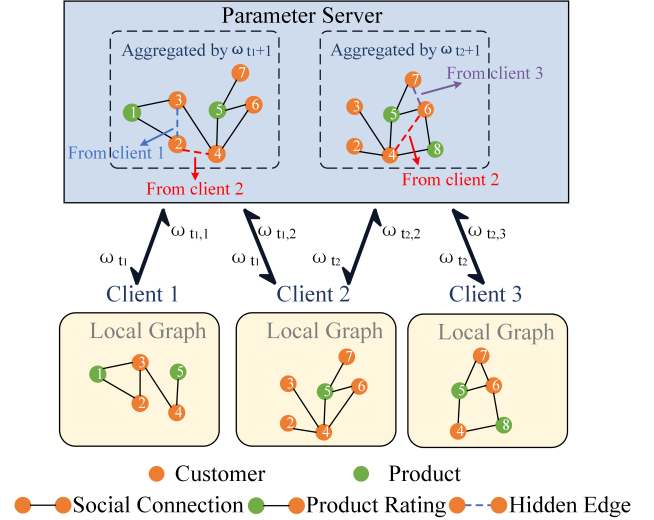


Fig. 2: Illustration of FedACS. PS selects clients 1 and 2 in round $t_1$, clients 2 and 3 in round $t_2$, respectively. The global model $\omega_{t_1+1}$ integrates hidden edges (dotted lines) of nodes 2 and 3 from local model $\omega_{t_1,1}$, nodes 2 and 4 from model $\omega_{t_1,2}$ in round $t_1$,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

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.

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 $\omega_{t_1}$ 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 $\omega_{t_1+1}$. 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 $\omega_{t_1+1}$ will learn the edge information. In the subsequent round $t_2$, the PS aggregates the local models from clients 2 and 3, enabling the global model $\omega_{t_2+1}$ to learn the edge information among neighbor nodes 4 and 6 from model $\omega_{t_2,2}$, nodes 6 and 7 from $\omega_{t_2,3}$. Through the above illustration, FedACS can effectively leverage the collaboration of multiple clients and their overlapping nodes to facilitate the 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

$\omega_t$ update function in Eq. (5) as follows:

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

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 $\nabla F_i(\omega_{t,i}; H_{t,i}) = \frac{1}{n_i} \sum_{h_{t,i} \in H_{t,i}} \nabla f(\omega_{t,i}, h_{t,i})$, $f$ 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 [20] 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 $\omega$, and $\nabla_w F_i(\omega, H_{t,i})$ 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 $\omega$ 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 loss function values of the train data and the test data are positively correlated for each client. Clients $c_t$ selected based on test data scores also have higher loss function values than the other clients. That is, $\sum_{i \in V_t} \frac{n_i}{n'} \nabla F_i(\omega_{t,i}, H_{t,i}) \geq \sum_{i=1}^{N} \frac{n_i}{n} \nabla F_i(\omega_{t,i}, H_{t,i})$*

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 $\bar{\omega}_{t+1} = \bar{\omega}_t - \eta \sum_{i \in V_t} \frac{n_i}{n'} \nabla F_i(\omega_t, H_{t,i})$. We use the expected gradient norm as the convergence metric that follows the convention in non-convex optimization [20].

Based on Assumption 1, we have

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

To enhance the understanding of the convergence analysis, let $d_t = \sum_{i=1}^{N} \frac{n_i}{n} \nabla F_i(\omega_{t,i}, H_{t,i})$. 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 $\|a + b\|^2 \leq 2(\|a\|^2 + \|b\|^2)$, we have the following result:

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

Next, we discuss the upper bound of $\mathbb{E}[\|\nabla F(\bar{\omega}_t) - d_t\|^2]$. According to previous work [20], we have the following lemma:

**Lemma 1.**
$$\mathbb{E}[\|\nabla F(\bar{\omega}_t) - d_t\|^2] \leq 3[\delta^2 + 2L^2\eta^2 \sum_{i \in N} \frac{n_i}{n} \mathbb{E}[\|\nabla F_i(\omega_t, H_{t,i})\|^2]] \quad (9)$$

According to Eq. (8), we have the following:

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

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 Eq. (10) for further analysis:

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

Next, we discuss the value of learning rate $\eta$. First, in order to ensure the convergence of the formula, we set the condition: $(\frac{\eta}{2} - L\eta^2) \geq 0, \eta \leq \frac{1}{2L}$. 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: $3(L\eta^2 + \frac{\eta}{2})2L^2\eta^2 \leq \frac{\eta}{2}$, which would be equivalent to solving the inequality $3(L\eta + \frac{1}{2})2L^2\eta^2 - \frac{1}{2} \leq 0$. Upon differentiating this inequality with respect to $\eta$, we have $2L^2\eta(9L\eta + 1)$. This derivative is non-negative for all $\eta \geq 0$. When $\eta = 0$, the result would be $-\frac{1}{2}$. Therefore, there must be $\eta \geq 0$ to make the inequality establish. We use $\bar{\eta}$ to denote the maximum value of $\eta$ which holds the inequality.

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

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

## 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 [21]. 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 (*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 [19], [22]. 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:

$$\min_{\omega} F(\omega) = \min_{\omega} \sum_{i=1}^{N} \frac{n_i}{n} \cdot f^i(\omega)$$

$$s.t. \begin{cases} \sum_{t=1}^{T} \sum_{i=1}^{N} X_i^t \cdot b_i^t \leq B, \\ \sum_{t=1}^{T} \sum_{i=1}^{N} X_i^t \geq C, \\ \sum_{i=1}^{N} X_i^t \geq 1, & \forall t \\ X_i^t \in \{0, 1\}, & \forall i, t \end{cases} \quad (13)$$

where $f^i(\cdot)$ 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 $n$. 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 one client 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.

### 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 the satisfactory model training performance. MAB is a classical problem in reinforcement learning (RL) [23] 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 [22]. 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 $t$-th 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 clients 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.

---

**Algorithm 1** The ACS algorithm in round $t$

**Input:** The accumulated reward: $\widehat{R}_m$, $\overline{R}_i$ and the selection count: $\widehat{N}_m^t$, $\overline{N}_i^t$ 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 $\{1, ..., N\text{-}1\}$ without repetition as $M$
7:      Select $M$ clients randomly as $V_t$
8: **else**
9:      Calculate $\mathcal{F}_m^t$ with Eq. (16) for each participation number $m$
10:      Select the maximum value from $\mathcal{F}_m^t$ as $M$
11:      Calculate $\mathcal{I}_i^t$ with Eq. (17) for each client $i$
12:      Select the top $M$ clients from $\mathcal{I}_i^t$ as $V_t$

---

### 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. Firstly, the ACS algorithm determines the number of participating clients. And then, it selects the most qualified clients 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 Alg. 1, where the whole training process is divided into two stages: overall exploration (Lines 2-7) and directional selection (Lines 8-12).

**Overall Exploration.** In this stage, the ACS algorithm aims to undertake an initial exploration across all potentially selected clients and the number of participating client. 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 7) 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 $\omega_{t+1}$ by Eq. (5). Then, the algorithm calculates the rewards for each selected client $i$ and participation number $m$ in round

$t$ through the following formulas:

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

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

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. 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:

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

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

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 Eq. (14) and Eq. (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 $i$. While the selection balance ensures that no client or participation number is neglected for 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 $\mathcal{F}_m^t$ with the maximum score as the selected participation number $M$ (Line 10), and selects the first $M$ clients as $V_t$ in round $t$ according to $\mathcal{I}_i^t$ for each client (Lines 11-12). Then, PS distributes the global model $\omega_t$ 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 $\omega_{t+1}$ by Eq. (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 [17] and Epinions [24], 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 [10]. 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 [18] and a two-layer GAT model [25] 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 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 [26], [27] and the GNN-based algorithm [28]. *SoRec* [26] 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* [27] models social network information as regularization terms to constrain the matrix factorization framework. *Consisrec* [28] 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 five FL-based algorithms as baselines, divided into two categories: full client participation [4], [10], [11] and client selection strategy [29], [30]. **Full client participation**:

TABLE II: 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 |
| FedACS(GAT) | 1.039 | **332** | **74** | **1.169** | **447** | **285** |
| FedACS(GCN) | **1.039** | 568 | 98 | 1.170 | 476 | 406 |

TABLE III: 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 |

*FedGNN* [11] draws on the user-item graph inferred from the local user-item interaction data for the recommendation. *FeSoG* [10] adopts relational attention and aggregation to handle the data heterogeneity and infers users' hidden states by using local data. *FedGraphNN* [4] 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* [29], denoted as POW selects clients with higher local losses to increase the rate of convergence. Random client selection strategy [30] ensures that each client has an equal probability, specifically 50%, of being selected for participation in each round of model training.

**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 [31] 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 II. The results show that FedACS reaches the best RMSE score among FL-based algorithms. Specifically, FedACS outperforms FedGNN in RMSE score by 58.9% and 64.0% on the two datasets, respectively. Compared to FedGraphNN, FedACS achieves an advantage of approximately 1.3%-4.1% with the GCN and GAT models. It indicates that FedACS achieves superior model training performance through the adaptive client selection algorithm. In the client selection algorithm, FedACS also demonstrates its own advantages. For instance, compared with POW, FedACS achieves an 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 between clients and select clients through an evaluation method. It is worth noting that the traditional FL client selection algorithm achieved poorer performance compared to the random client selection strategy. This is because the traditional FL client selection did not take into account the graph data features and hidden edges in FGL, resulting in performance degradation. Moreover, we also compare the RMSE score of FedACS with traditional centralized baselines in Table III. 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, FedACS is only 6.2% and 10.1% lower than Consisrec on the two datasets. This can be attributed to FedACS enabling the global model to learn the hidden edges between clients through the adaptive client selection algorithm.

(2) **Resource Consumption.** To evaluate the resource efficiency of FedACS, we also compare the communication and time costs for FedACS and baselines. Due to inherent differences between 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 II. It is
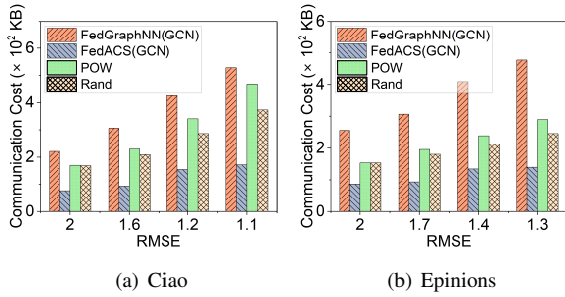
(a) Ciao  (b) Epinions

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

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) achieves 61.0% communication bandwidth and 17.8% time saving on the Ciao dataset. Besides, on the Epinions dataset, FedACS(GCN) only takes 476KB and 406s to complete the training while FedGraphNN(GCN) requires 851KB and 538s, respectively. Compared to other client selection methods, FedACS also has advantages. For example, compared with Rand, FedACS(GAT) achieves 56.3% communication bandwidth and 50.8% time saving on the Ciao dataset. The reason lies in that FedACS adaptively selects the number of 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 the poor model training performance of FedGNN and FeSoG, and to ensure a fair evaluation of FedACS, we choose only FedGraphNN, POW and Rand to compare the performance under different constraints with FedACS in the following.

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 reducing the required 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. For instance, as illustrated in Fig. 3(a), FedACS(GCN) requires 172KB to achieve the RMSE score of 1.1 on the Ciao dataset, while FedGraphNN(GCN) and Rand consume 527KB and 373KB, 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 efficient communication during the training process.

*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 dataset, *i.e.*, Average, Skewed and Extreme. We measure the communication cost, under varying data distributions, when both FedACS and the baselines attain different target RMSE scores (*e.g.*, 1.055, 1.071 and 1.041 on Ciao under three data distributions). Because the achievable RMSE scores for FedACS and baselines vary under different

data distributions, we set the target RMSE score that FedACS and baselines can achieve in various models under three data distributions, respectively. The results in Fig. 4 reveal that FedACS outperforms the baselines in terms of reducing the required communication to attain the target RMSE score. For instance, as illustrated in Fig. 4(a), FedACS(GCN) takes 209KB to achieve the RMSE score of 1.071 on the Ciao dataset under the Skewed data distribution, while FedGraphNN(GCN), POW and Rand take 851KB, 715KB and 713KB, respectively. On the Epinions dataset, as depicted in Fig. 4(b), FedACS(GCN) reduces the communication cost by 73.0% compared to FedGraphNN(GCN) when attaining the same RMSE score of 1.190 under the Extreme data distribution. Compared to POW and Rand, FedACS(GCN) has also achieved an average communication performance advantage of 52.5% when attaining the same RMSE score of 1.122 under the Skewed data distribution.

Additionally, we also record the RMSE score of FedACS and baselines within the given communication budget of 850KB across the two datasets in Fig. 5. Impressively, as the skewness of data distribution amplifies, the RMSE score of models trained across all systems initially worsens, and then improves. 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 differences from clients with minimal data will be diminished due to the global aggregation using data size as the aggregation parameter. 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 FedGraphNN. Similar to the Average data distribution, FedACS always achieves superior RMSE scores compared to FedGraphNN under both the Skewed and Extreme data distributions. For instance, as illustrated in Fig. 5(b), FedACS(GCN) outperforms FedGraphNN(GCN) in terms of the RMSE score by 5.4% under the Extreme data distribution on the Epinions dataset. On the Ciao dataset, under the Extreme data distribution and with an 850KB communication budget, FedACS(GCN) achieves a 2.2% performance advantage over FedGraphNN(GCN) and Rand. These above experimental results demonstrate the superiority of FedACS in terms of reducing communication cost and alleviating non-IID issue during the training process.

## VIII. CONCLUSION

In this work, we propose FedACS to adaptively select 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.
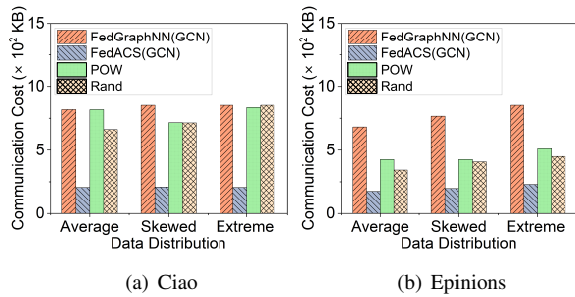
Fig. 4: Communication cost to reach different target RMSE scores (Ciao: 1.055, 1.071 and 1.041; Epinions: 1.219, 1.222 and 1.190) on the two datasets under different data distributions.
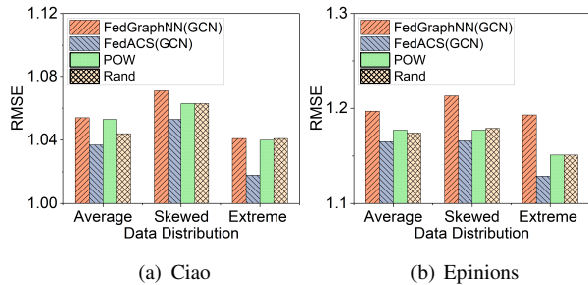


Fig. 5: RMSE for FedACS and baselines on the two datasets with communication budget (850 KB) under different data distributions.

## REFERENCES

[1] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[2] Z. Li, M. Bilal, X. Xu, J. Jiang, and Y. Cui, "Federated learning-based cross-enterprise recommendation with graph neural networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 673–682, 2022.

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

[4] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, S. Y. Philip, Y. Rong *et al.*, "Fedgraphnn: A federated learning benchmark system for graph neural networks," in *ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML)*, 2021.

[5] J. Yan, J. Liu, H. Xu, Z. Wang, and C. Qiao, "Peaches: Personalized federated learning with neural architecture search in edge computing," *IEEE Transactions on Mobile Computing*, 2024.

[6] J. Baek, W. Jeong, J. Jin, J. Yoon, and S. J. Hwang, "Personalized subgraph federated learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1396–1415.

[7] Z. Yao, J. Liu, H. Xu, L. Wang, C. Qian, and Y. Liao, "Ferrari: A personalized federated learning framework for heterogeneous edge clients," *IEEE Transactions on Mobile Computing*, 2024.

[8] J. Huang, B. Liu, C. Miao, X. Zhang, J. Liu, L. Su, Z. Liu, and Y. Gu, "Phyfinatt: An undetectable attack framework against phy layer fingerprint-based wifi authentication," *IEEE Transactions on Mobile Computing*, 2023.

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

[10] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–24, 2022.

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

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

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

[14] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-iid graphs," *Advances in neural information processing systems*, vol. 34, pp. 18 839–18 852, 2021.

[15] 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. 24 462–24 474, 2021.

[16] J. Liu, J. Yan, H. Xu, Z. Wang, J. Huang, and Y. Xu, "Finch: Enhancing federated learning with hierarchical neural architecture search," *IEEE Transactions on Mobile Computing*, 2023.

[17] J. Tang, H. Gao, and H. Liu, "mtrust: Discerning multi-faceted trust in a connected world," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 93–102.

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

[19] L. Wang, Y. Xu, H. Xu, J. Liu, Z. Wang, and L. Huang, "Enhancing federated learning with in-cloud unlabeled data," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 136–149.

[20] B. Du and C. Wu, "Federated graph learning with periodic neighbour sampling," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 2022, pp. 1–10.

[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

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

[23] C. Dann, T. Lattimore, and E. Brunskill, "Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *International semantic Web conference*. Springer, 2003, pp. 351–368.

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

[26] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: Social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008, pp. 931–940.

[27] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 287–296.

[28] L. Yang, Z. Liu, Y. Dou, J. Ma, and P. S. Yu, "Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation," in *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*, 2021, pp. 2141–2145.

[29] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 10 351–10 375.

[30] M. Tang, X. Ning, Y. Wang, J. Sun, Y. Wang, H. Li, and Y. Chen, "Fedcor: Correlation-based active client selection strategy for heterogeneous federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 102–10 111.

[31] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.