Reinforcement Active Client Selection for Federated Heterogeneous Graph Learning

Jia Wang[1,2], Yawen Li[3*], Yingxia Shao[1,2], Zhe Xue[1,2], Zeli Guan[1,2], Ang Li[1,2], Guanhua Ye[1,2]

[1]School of Computer Science (National Pilot School of Software Engineering), Beijing University of Posts and Telecommunications

[2]Beijing Key Laboratory of Intelligent Telecommunication Software and Multimedia

[3]SChool of Economics and Management, Beijing University of Posts and Telecommunications

Beijing 100876, PR China

Wangj2021110865, shaoyx, xuezhe, guanzeli, david.lee, g.ye@bupt.edu.cn

Abstract

Careful client selection for aggregation can help improve the global model's performance. However, existing research on federated heterogeneous graph learning (FHGL) has paid limited attention to the client selection (CS) problem. Current CS algorithms struggle to accurately assess client contributions and select suitable participants in FHGL, creating a trade-off between convergence and accuracy. In this paper, we propose a Reinforcement Active client selection based Federated Heterogeneous Graph Learning (RAFHGL), which precisely evaluates the importance of local heterogeneous graph data and selects high-contributing clients for aggregation. RAFHGL employs an active learning agent to select representative nodes for local training. The statistical features of the active scores are used to assess client contributions. A client selection agent then chooses clients conducive to global model convergence for aggregation. To address heterogeneity introduced by sample and client selection, the training process stabilizes by correcting local losses based on data prototypes. Experimental results on 4 publicly available heterogeneous graph datasets show that RAFHGL outperforms existing Client Selection algorithms in federated heterogeneous graph learning scenarios in terms of performance and convergence.

Introduction

Similar to other neural networks, a substantial amount of data can enhance the performance of Heterogeneous Graph Neural Network (HGNN). However, in reality, it is often challenging to collect extensive data for training due to constraints such as hardware limitations or privacy concerns from multiple data sources (Lin et al. 2020; Zhang et al. 2022a; Chen et al. 2021; Peng et al. 2021). Federated Learning (FL) (McMahan et al. 2017; Chai et al. 2020; Dennis, Li, and Smith 2021) has emerged as a promising approach for collaborative model training without the need to exchange raw data, particularly well-suited for training machine learning models on distributed devices. Consequently, research on Federated Graph Learning (FGL) (Zhang et al. 2021; Liu et al. 2022b; Tan et al. 2023; Hu et al. 2022) has emerged, with Federated Heterogeneous Graph Learning (FHGL) allowing local clients to maintain Heterogeneous Information Network (HIN) data with multiple node or edge types. Common FHGL still requires all clients to participate in global aggregation, resulting in significant computational resource and communication time wastage or randomly select clients for global training without the ability to choose clients that could better aid global model convergence or performance improvement (Fu and King 2023; Yan et al. 2023; Qu et al. 2022; Song et al. 2023). Although existing client selection (CS) (Ribero and Vikalo 2020) algorithms have been extensively studied in the context of Euclidean data, the utilization of CS algorithms on FHGL has not received much attention. Furthermore, the unique characteristics of HIN and the complexity of HGNN make it challenging to accurately assess client contributions and select appropriate clients for aggregation in the FHGL scenario.

In FHGL scenarios, client selection (CS) faces two main challenges. On one hand, the diverse node and edge types in HINs introduce complex topological information, requiring more advanced HGNN architectures to handle more difficult tasks. Existing CS algorithms that rely on model information and performance metrics introduce biases when evaluating client quality (Wang et al. 2024; Shi and Shen 2021). As a result, when applied to FHGL, these CS algorithms struggle to accurately select clients that most contribute to global model aggregation. On the other hand, existing CS algorithms often face a trade-off between convergence and generalization due to high data heterogeneity (Zhan, Li, and Guo 2020; Guan et al. 2023). The complexity and diversity of HIN increase the data heterogeneity in FL. Data distributions across clients may vary significantly, with diverse node and edge combinations creating complex topological information, challenging FL's robustness in managing data heterogeneity (Song et al. 2023; Yang, Liu, and Kassab 2023). Traditional CS algorithms may worsen the trade-off between convergence and accuracy in such scenarios (Goetz et al. 2019; Rjoub et al. 2022; Jiang et al. 2022).

In this paper, we introduce a novel approach, Reinforcement Active Client Selection-based Federated Heterogeneous Graph Learning (RAFHGL), which combines Reinforcement Learning (RL) and Active Learning (AL) to

*Corresponding author.(warmly0716@126.com)

enhance CS in FHGL. Our RAFHGL algorithm dynamically adapts to evolving client scenarios, offering an effective CS method that balances short-term convergence with long-term contributions to the global model. By leveraging reinforcement learning's ability to handle delayed rewards, RAFHGL enables intelligent client decision-making, benefiting both local models and the overall performance of the federated learning framework. The following sections will provide a detailed exposition of the proposed algorithm and present experimental results demonstrating its effectiveness across various federated learning scenarios.

The primary contribution of this paper can be summarized as follows:

1. We present a reinforcement active learning based method for evaluating the information content of nodes in HINs prioritizing the participation of nodes with higher perplexity in local training. Evaluate the distribution of effective samples as an indicator of client contributions, addressing the challenge in accurately evaluating the quality of client data.
2. We present a CS strategy designed for the FHGL scenario, which employs a RL agent to assess the distribution of locally effective nodes. It selectively chooses clients for aggregation training, aiming to mitigate the trade-off between convergence and performance inherent in traditional CS approaches.
3. We propose a data prototype-based local training correction method, which constrains the distance between the node embeddings obtained by different clients in the feature space and the prototype set. Effectively alleviate client drift caused by active selection on both the client and server sides.

Related Work

Federated Learning Client Selection

Existing CS strategies based on Euclidean data primarily rely on model features (Fraboni et al. 2021; Balakrishnan et al. 2022), such as parameters, gradients, parameter variations, and performance metrics (Huang et al. 2023; Cho, Wang, and Joshi 2020; Tang et al. 2022), including loss, accuracy, and accuracy variations, to decide client participation in training. However, model performance results from the interaction between the global model and local data, with differences primarily arising from variations in client-side data.

Heterogeneous Graph Learning

HGNN is an extension of graph neural networks designed to handle HIN. In heterogeneous graphs, nodes and edges of different types have distinct semantics and attributes (Wang et al. 2022; Tian et al. 2023; Liu et al. 2023; Yang et al. 2023). This heterogeneity presents challenges for directly applying conventional homogeneous graph neural networks. Consequently, HGNN shows great potential in addressing the complex relationships in HIN and has been widely studied in practical applications, such as legal statute identification (Paul, Goyal, and Ghosh 2022), drug repurposing (Mei et al. 2022), and sentiment analysis (Lu, Li, and Wei 2022; An et al. 2022).

Federated Active Selection Algorithm

RL provides a compelling solution for dynamic CS in federated learning (Wang et al. 2020; Zhang, Lin, and Zhang 2022). Unlike traditional models that require pre-training on extensive selection data, RL models can adaptively learn CS patterns (Tang et al. 2022). The introduction of AL (Ren et al. 2021; Liu et al. 2022a), by focusing on assessing the quality of client-contributed data, further enhances the CS process. Integrating AL into the process allows for a more nuanced evaluation of client data quality (Chen et al. 2019; Zhang et al. 2022c), using the distribution of effective samples as an indicator of client contributions, while respecting privacy constraints.

Methodology

Problem Formulation

In this paper, we define a federated heterogeneous graph learning framework consisting of a global server and K clients. Each client, indexed as the k^{th} client, stores and maintains an independent heterogeneous graph dataset $D^k = (\mathcal{G}^k, X^k, Y^k)$, where X^k and Y^k are feature matrix and label matrix, $\mathcal{G}^k = (V^k, E^k, A^k, R^k)$ is the heterogeneous graph. All clients will engage in federated learning to jointly train a HGNN model without compromising local privacy.

Framework Overview

The objective of RAFHGL is to collaboratively train a unified HGNN without disclosing local client data. Through AL to assess the quality of client-side data, we selectively choose samples that are more informative for the model. Utilizing the distribution of effective samples as an indicator for evaluating client quality can mitigate interference from other factors while ensuring data privacy. The overall framework of RAFHGL is illustrated in Figure 1.

Active Learning Agent on Local Clients

On the local client side, ALA calculates the active scores of samples based on the local state. It prioritizes selecting nodes with higher scores to participate in the training of the local HGNN. Simultaneously, the statistical features of active scores are transmitted to the server. Since the outcomes of local AL serve as crucial inputs for the global reinforcement learning state, generating stable and accurate AL results is paramount. RL algorithm can adaptively adjust sample selection strategies to address diverse usage scenarios and the continually evolving global model.

To achieve this objective, the instantiation of AL on the client side is structured as a Markov Decision Process (MDP). The ALA on the k^{th} client can be represented as a tuple $\mathcal{M}^k = \langle S^k, U^k, r^k \rangle$, where S^k denotes the local state set, U^k signifies the action set, specifically the probability distribution over output selections, and r^k signifies the reward associated with the chosen action.

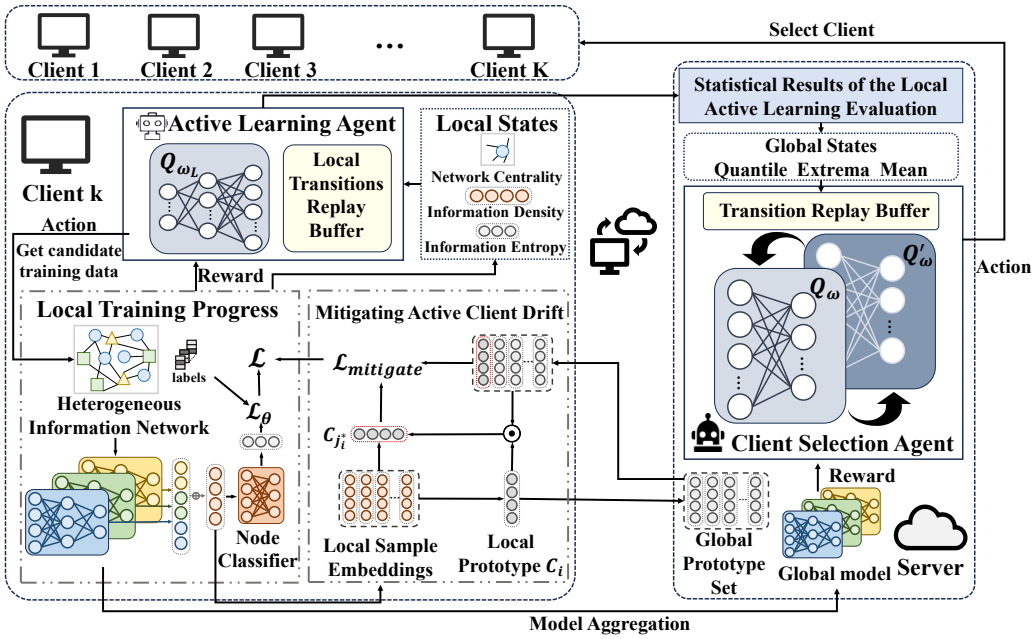


Figure 1: Architecture of the proposed Reinforcement Active Client Selection based Federated Heterogeneous Graph Learning (RAFHGL), which primarily relying on 2 distinct agents, the local Active Learning Agent (ALA) and the global Client Selection Agent (CSA), to assist the server in precisely assessing client information and selecting the most informative clients for global model convergence. Simultaneously, the proactive selection by the 2 agents introduces bias into the training data, thereby prompting the proposal of a data-prototype-based mitigating method, which aims to alleviate the overly personalized development of local models.

State. The local active state s^k comprises 3 key aspects: *Node centrality*, reflecting a node's significance in the data network, is effectively calculated using the PageRank algorithm (Zhang et al. 2022b). *Information entropy* gauges the classifier's ease in accurately classifying a node's embedding, with high entropy aiding the learning of ambiguous information and low entropy reinforcing existing knowledge. In a federated system, *information density* involves AL on a client, selecting samples with the highest training priority instead of creating a core set for labeling. This paper utilizes the hidden layer representation of nodes to assess the embedding density of samples, as traditional methods relying on distance computation with core sets are challenging to apply.

Action. In accordance with the state of the local node, ALA computes active scores for all local samples using a fixed batch size. The top-B_s samples with the highest proactive scores are selected to participate in local training. Simultaneously, statistical features of the active scores are calculated.

Reward. The objective of local AL can be summarized as enhancing the model's performance, specifically in terms of both local training performance Acc^L and global model performance Acc^G. It is crucial to note that elevating the global performance takes precedence; hence, the change in global performance is computed and exponentially amplified through an exponential base B_L. To ensure that samples possessing lower or even negative local performance changes, but which could contribute to the enhancement of global performance, are not overlooked, local performance is accorded a linear weight. The specific computational formula is provided in Eq. 1.

$$r_t^k = B_L^{(Acc_t^G - Acc_{t-1}^G)} + Acc_t^L. \tag{1}$$

Client Selection Agent on Global Server

On the global server side, CSA on global server constructs a global state derived from the aggregated statistical features of active scores across all clients. Subsequently, the server identifies clients most likely to make significant contributions to the global model for model aggregation. Therefore, in each round of federated communication, we employ CSA to select K devices for participation in training. The specific design of the agent CSA for CS is outlined as follows:

State. The selection state of the global client can be represented as a vector $s_G = \{u_1, u_2, ..., u_K\}$, where u_k signifies the statistical features of the active score results of the k^{th} client. For a given parameter n_q, the statistical features correspond to the $n_q^{k^{th}}$ quantile of the assessment results.

Action. Similar to the work presented in (Wang et al. 2020), the action space is defined as $\{1, 2, ..., K\}$, where $a = k$ indicates the selection of the k^{th} device for participation in federated training.

Reward. The reward, on the other hand, is computed according to Eq. 2. This is designed to incentivize the agent to approach the target accuracy Acc^T, thereby achieving higher

precision while concurrently controlling the number of training rounds. Here, B_G represents the global exponential base.

$$r_{global} = B_G^{(Acc^G - Acc^T)} - 1. \quad (2)$$

Mitigating Active Drift through Data Prototype

Due to the data selection process on both the client and server sides, the global model in each aggregation round is influenced by only a subset of distinctive data. This may potentially lead to the convergence of the global model to a local optimum and exacerbate client drift phenomena. To address this issue, we propose a mitigation strategy based on data prototypes (Tan et al. 2022; Long et al. 2023). This strategy aims to constrain local training, thereby preventing overly significant impacts of outlier samples on the model and ensuring training stability.

Specifically, each client initially calculates the local data prototypes C_k based on the hidden layer representations of their local dataset D_k.

$$C_k = \frac{1}{|D_k|} \sum_{(x,y)\in D_k} f_k(\theta_k; x), \quad (3)$$

where $f_k(\theta_k; \cdot)$ represents the local heterogeneous graph neural network on k^{th} client.

Subsequently, the RAFHGL server collects all client prototypes to obtain a global prototype set $\mathcal{P} = \{C_1, C_2, ..., C_K\}$, which is then distributed to the clients. During this process, the server refrains from aggregating the prototypes. Instead, it introduces a regularization term $\mathcal{L}^k_{mitigate}$ to penalize those nodes whose hidden layer features deviate from the entire federated system.

$$\mathcal{L}^k_{mitigate} = \sum_{(x,y)\in D_k} \min_{i^*_k} \|f_k(\theta_k; x) - C_{i^*_k}\|^2. \quad (4)$$

The addition of the regularization term aims to mitigate client drift induced by repeated active selections between both the client side and the server side. Excessive penalization, however, might restrict the pace of client exploration in the solution space, leading to a reduced convergence speed. Therefore, rather than computing a global prototype, we intuitively select the local prototype with the smallest deviation from the respective local client for correction.

$$C_{i^*_k} = \mathcal{P} \odot C_k = \min_{0<i<N, i\neq k} \|C_k - C_i\|^2. \quad (5)$$

Training Process

The complete training process of RAFHGL is outlined in Algorithm 1. On the client side, ALA evaluates local heterogeneous graph nodes based on topology, entropy, and embedding information, generating statistical features of the active score. It prioritizes the top-B_s samples with the highest active scores for local updates. During local updates, the algorithm calculates the loss $\mathcal{L}^k\theta$ of the HGNN and simultaneously computes the mitigation term $\mathcal{L}^k mitigate$ based on Eq. 4. The HGNN is then updated according to the total loss $\mathcal{L}^k$. After each training round, the state transition is written into the Local Transition Replay Buffer (LTRB). Randomly selected transition samples from LTRB are used to train the ALA. On the server side, a stochastic probability ϵ is employed to determine whether to select a client using actions generated by CSA. With probability ϵ, the global server selects a client based on CSA-generated actions, and with probability $1-\epsilon$, it adopts a CS strategy using AFL with the mean active score. In the initial stages, when ϵ is relatively large, clients are selected using a heuristic approach, preparing a substantial training dataset for CSA. Similarly, the global state transition is recorded in a global replay buffer for CSA training. During each communication round using CSA to select clients, the server utilizes stored statistical features of active scores to obtain global states. Using actions generated by CSA, the server selects clients S_t to participate in aggregation. It obtains local models from the selected clients to aggregate the global model, and collects corresponding updates to the global prototype set $\mathcal{P}$. The selected clients also upload statistical feature updates of active scores to update the CSA state, eliminating additional communication overhead.

Algorithm 1: Training Process of RAFHGL

Procedure SERVEREXECUTION
Initialize CSA, $\mathcal{P}$ and $f_k(\theta_k)$ for all clients.
for each communication round t = 1, 2, ..., T **do**
  CSA selects a subset S_t of clients.
  for each client $s \in S_t$ **do**
    $f^t_s(\theta^t_s),\ u^t_s,\ C^t_s \leftarrow$ CLIENTUPDATE(s, $\mathcal{P}^{(t-1)}$, $f^{(t-1)}(\theta^{(t-1)})$)
  end for
  Update global information $f^t(\theta^t)$, $\mathcal{P}^t$, s^t_G.
  Store global transition and train CSA.
end for

Procedure CLIENTUPDATE(k, $\mathcal{P}$, $f(\theta)$)
for epoch e = 1, 2, ..., E **do**
  CSA calculates active scores for all nodes.
  Select top-B_s samples as D^k_s.
  for minibatch m sampled from D^k_s **do**
    Calculate $\mathcal{L}^k_\theta$ using Y^k and $f_k(\theta_k; x^k)$.
    Calculate $\mathcal{L}^k_{mitigate}$ according to Eq. 4.
    $\mathcal{L}^k = \mathcal{L}^k_\theta + \lambda\mathcal{L}^k_{mitigate}$
    Update local model according to $\mathcal{L}^k$.
  end for
  Store local transition and train ALA.
end for
Calculate u^t_k and C^t_k.
Update $f^t_k(\theta^t_k)$, u^t_k, C^t_k to Server.

Experiments

Experiment Setting

Datasets. We conducted node classification experiments on 4 popular public HG datasets, as shown in Table 1.

Data partition. We employed 2 distinct data partitioning methods for creating a federated environment. *Uniform*: nodes in the HIN were randomly divided into K subsets. Subsequently, a subgraph for each subset was constructed

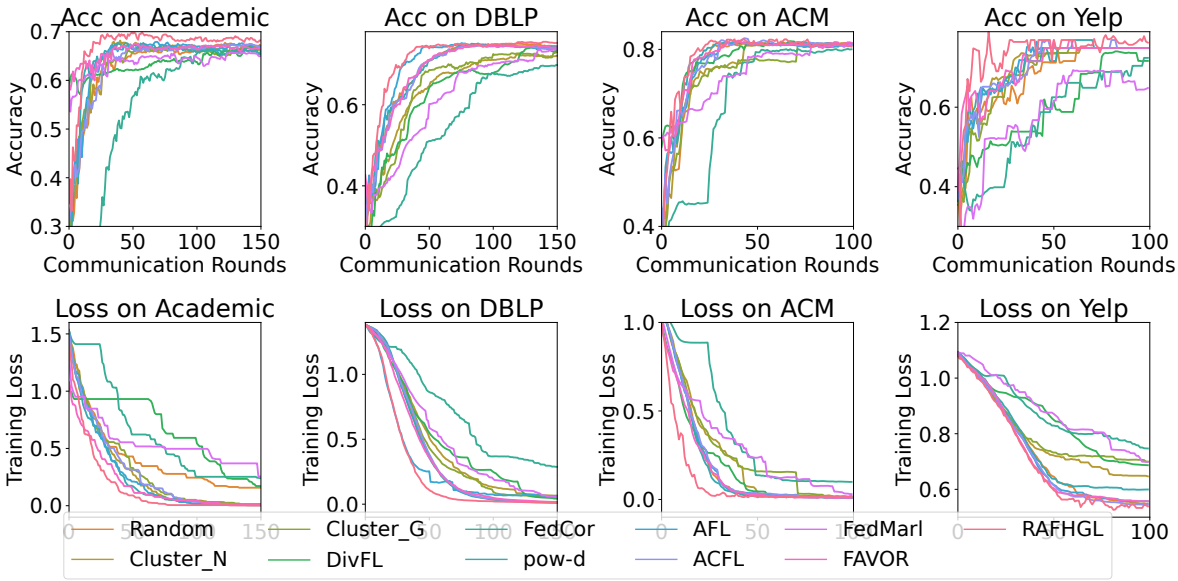


Figure 2: Comparative analysis of training curves on distinct datasets. In this context, 2 participants are selected from each of the 20 clients for training and aggregation on the Academic, while for the remaining datasets, 2 participants are chosen from a pool of 10 clients each. Local epoch for all datasets is 3, and the data partition is based on Dirichlet distribution.

Datasets	# Nodes	# Edges	# Labels
Academic(Zhang et al. 2019)	49,708	202,067	5
DBLP(Fu et al. 2020)	26,128	239,566	4
ACM(Yun et al. 2019)	11,246	34,852	3
Yelp(Hu, Fang, and Shi 2019)	3,913	77,360	3

Table 1: Description of datasets.

to represent a sub-dataset in the federated system. This ensured roughly equal node quantities and uniform label distributions across clients. *Dirichlet*: utilize a Dirichlet distribution $D(\beta)$ to partition target nodes for classification, while the remaining nodes were randomly assigned. The resulting subgraphs represented K clients with varying label distributions and node quantities. These approaches simulate different degrees of data heterogeneity among clients in the FL scenario.

Baseline. In the context of FHGL, we implemented various CS algorithms for comparative experiments to validate the performance of RAFHGL, including **Random**(Li et al. 2020), **Cluster_N**(Fraboni et al. 2021), **Cluster_G**(Fraboni et al. 2021), **DivFL**(Balakrishnan et al. 2022), **FedCor**(Tang et al. 2022), **pow-d** (Cho, Wang, and Joshi 2020), **AFL**(Goetz et al. 2019), **ACFL** (Huang et al. 2023), **FedMarl**(Zhang, Lin, and Zhang 2022), **FAVOR** (Wang et al. 2020). To ensure fairness, all methods use a HAN with 8 attention heads and a hidden layer size of 128 as the local training model, and are trained with the same set of hyperparameters in . We employ the Adam optimizer with a learning rate of [0.005, 0.002, 0.003, 0.003] across 4 datasets, and weight decay is set to 10^{-7} in all experiments.

Implementation details. Our implementation of heterogeneous graph networks is based on the *DGL*(Wang et al. 2019) library, while the processing of heterogeneous information network data relies on the *OpenHGNN*(Hui Han 2022) library. GPU acceleration is performed on NVIDIA RTX 2080 TI. To maximize the disparity in partition results and achieve a significant difference from random partitioning, the β parameter in the $D(\beta)$ is set to 0.5. For RAFHGL, CSA is trained using Double Deep Q-Network (DDQN) (Van Hasselt, Guez, and Silver 2016), while ALA is trained using Deep Q-Network (DQN) (Mnih et al. 2015). The neural network architectures for both agents consist of a two-layer perceptron with a hidden layer size of 256. The learning rate for the agents is set to 0.001.

Performance Analysis

To validate the robustness of the RAFHGL algorithm and assess its performance across diverse federated environments, we conducted a comprehensive comparison with baseline algorithms under various federated settings, including different data partitioning methods, federated client quantities, and local training epochs. Some representative training curves are illustrated in Figure 2. It is evident that RAFHGL consistently demonstrates stable convergence and outstanding accuracy across all datasets. Baseline methods either exhibit overall inferior performance compared to RAFHGL or involve a noticeable trade-off between convergence and accuracy. Methods with faster convergence often struggle to ensure final performance, while those exhibiting better performance require a higher number of communication rounds.

Different setting of partition method. We conducted experiments on 4 datasets using different partitioning methods, and the results are presented in Table 2. RAFHGL not

	Academic		DBLP		ACM		Yelp	
	Uniform	Dirichlet	Uniform	Dirichlet	Uniform	Dirichlet	Uniform	Dirichlet
Random	56.27±4.45	66.57±8.76	67.42±5.94	73.40±8.87	85.85±4.32	81.10±12.86	70.01±5.82	74.84±8.53
Clusterd_N	56.42±4.66	66.65±8.79	67.71±6.04	73.12±9.47	86.21±4.54	81.03±13.02	69.85±4.23	74.90±8.61
Clusterd_G	55.91±4.70	65.91±9.55	67.61±5.79	73.14±10.28	84.93±4.40	81.22±12.81	67.58±6.69	74.91±14.07
DivFL	52.86±6.74	66.03±8.23	67.33±6.71	73.73±8.17	86.11±4.19	81.38±12.91	63.51±5.16	72.75±14.28
FedCor	56.13±4.71	65.46±10.20	66.09±7.20	69.35±16.48	86.23±5.14	79.62±16.01	72.27±4.09	66.88±16.81
pow-d	56.25±4.64	66.82±8.75	67.21±6.26	73.33±8.21	84.99±4.39	81.60±14.10	67.24±7.00	74.81±8.51
AFL	56.27±4.46	66.95±8.33	66.84±6.51	73.47±9.04	86.23±4.76	79.77±12.82	70.76±5.83	74.82±8.43
ACFL	56.41±4.79	66.51±9.39	67.14±6.77	73.53±9.17	86.10±5.15	80.72±12.78	70.22±6.46	74.67±7.53
FedMarl	51.75±6.29	65.66±8.66	67.04±6.05	73.47±8.79	85.72±7.32	78.52±12.95	71.31±5.84	69.14±14.69
FAVOR	55.73±4.84	66.10±8.73	67.12±7.17	73.52±9.03	85.82±4.88	80.82±12.83	61.05±10.71	66.07±6.92
RAFHGL	**56.62±3.47**	**69.01±8.16**	**67.73±5.06**	**74.68±7.30**	**86.57±4.08**	**82.24±12.71**	**72.86±3.92**	**75.07±6.86**

Table 2: Comparison of method Accuracy(%) across different data partition strategies. When sampling according to the Dirichlet distribution, similar nodes are more likely to be clustered on the same client, preserving more edge information. Therefore, the performance differences between the two partitioning methods are not solely attributed to data heterogeneity.

# Total	10			20			30		
# Selected	2	3	5	2	5	10	5	10	15
Random	91(1.6×)	78(1.6×)	75(2.2×)	128(1.4×)	110(1.7×)	100(1.7×)	104(1.2×)	93(1.3×)	91(1.4×)
Cluster-N	101(1.7×)	88(1.8×)	87(2.6×)	91(1×)	84(1.3×)	80(1.4×)	116(1.4×)	86(1.2×)	81(1.3×)
Cluster-G	145(2.5×)	84(1.7×)	52(1.5×)	145(1.6×)	90(1.4×)	63(1.1×)	104(1.2×)	82(1.2×)	71(1.1×)
DivFL	130(2.2×)	93(1.9×)	78(2.3×)	155(1.6×)	67(1×)	59(1×)	131(1.5×)	98(1.4×)	80(1.3×)
FedCor	132(2.3×)	95(1.9×)	84(2.5×)	127(1.7×)	82(1.2×)	79(1.4×)	123(1.4×)	96(1.4×)	82(1.3×)
pow-d	87(1.5×)	67(1.3×)	40(1.2×)	121(1.4×)	77(1.2×)	62(1.1×)	89(1×)	84(1.2×)	75(1.2×)
AFL	93(1.6×)	100(2×)	104(3×)	115(1.3×)	83(1.3×)	75(1.3×)	88(1×)	70(1×)	110(1.7×)
ACFL	105(1.8×)	99(2×)	77(2.3×)	155(1.7×)	80(1.2×)	64(1.1×)	114(1.3×)	99(1.4×)	74(1.2×)
FedMarl	141(2.4×)	130(2.6×)	76(2.2×)	122(1.4×)	99(1.5×)	103(1.8×)	115(1.4×)	92(1.3×)	86(1.4×)
FAVOR	94(1.6×)	80(1.6×)	48(1.4×)	107(1.2×)	94(1.4×)	63(1.1×)	127(1.5×)	76(1.1×)	107(1.7×)
RAFHGL	**58(1×)**	**50(1×)**	**34(1×)**	**90(1×)**	**65(1×)**	**56(1×)**	**85(1×)**	**69(1×)**	**63(1×)**

Table 3: Convergence rounds required under a distinct number of total clients and selected participation in aggregation per round on Academic. Diverse client quantities serve to simulate various federated scenarios.

only achieves high accuracy across various data partitions for each dataset but also exhibits smaller accuracy disparities among clients. The highest improvement, notably, reaches up to 11.81% and 9% on the Yelp dataset. In the case of the DBLP dataset with Dirichlet partitioning, the accuracy variance also decreases by 6.2% to 9.18%. It is worth noting that when partitioning graph data, some edges are inevitably discarded.

Different setting of client number. Diverse client quantities serve to simulate various federated scenarios. We conducted experiments with total clients set at [10, 20, 30], and the results in Table 3 clearly demonstrate that RAFHGL exhibits significant convergence speed advantages over baseline algorithms across different client quantity settings. Specifically, when selecting 2 clients from 10, 2 clients from 20, and 5 clients from 30, RAFHGL demonstrates improved convergence efficiency by 36%, 30%, and 18%, respectively, compared to the Random method.

Different setting of local training epoch. The number of local training epochs directly impacts the overall training effectiveness. RAFHGL demonstrates more pronounced performance advantages with an increase in local training rounds as shown in Figure 3. In scenarios where local training rounds are set at [3, 5, 10], RAFHGL exhibits accuracy improvements of 1.1%, 1.08%, and 1.69%, respectively. Additionally, even with lower local training rounds, RAFHGL ensures a convergence advantage, outperforming the Random method by 20 rounds when local training epochs are set at 1.

Parameter Analysis

RAFHGL's hyperparameters consist of common hyperparameters shared among algorithms and hyperparameters unique to RAFHGL. Common hyperparameters like model learning rate and weight decay, which are adjusted and selected through simple tuning and shared across all algorithms. RAFHGL's unique hyperparameters like the loss weight λ, the learning rates and the length of the hidden layer of the 2 agents. It is satisfying to note that most of these unique hyperparameters demonstrate considerable robustness, as variations across datasets have minimal impact on model performance. While parameters like the strategy probability ϵ exert significant influence on model performance as shown in Figures 4, but this influence follows consistent patterns across different datasets. For unique hyperparameter that exhibit minimal variation across datasets, we conduct parameter tuning on Academic dataset and apply the same settings across all datasets. A notably unique

hyperparameter is the quantiles n_q, which yields significant differences in results across different datasets, as illustrated in Figure 5 for the 4 datasets.

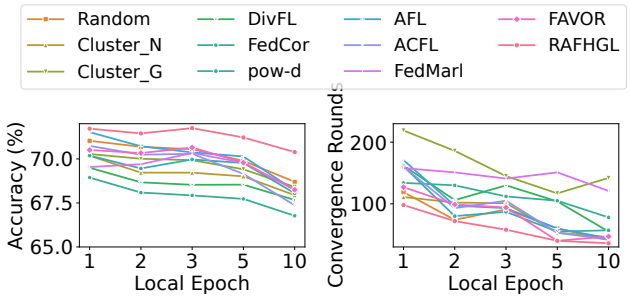
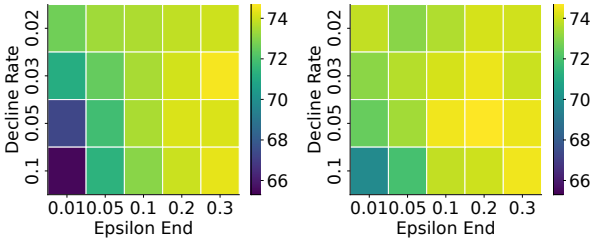


Figure 3: Performance comparison under various local epochs. More local training epochs can accelerate convergence but may intensify the client drift effect, consequently diminishing overall performance.

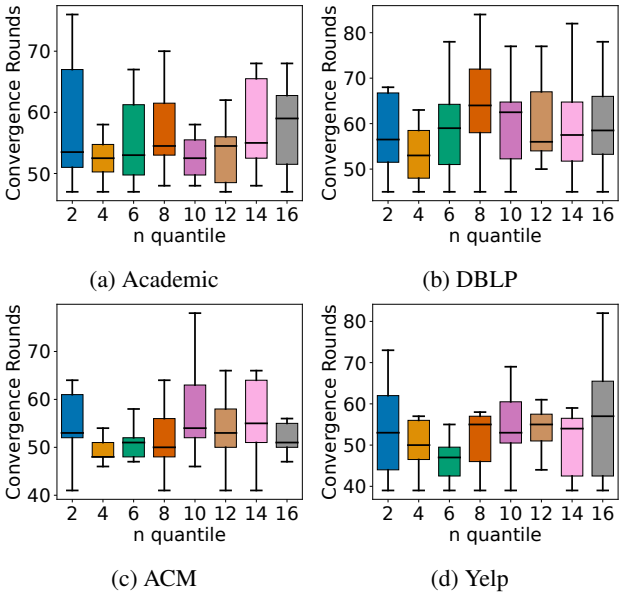


(a) (b)

Figure 4: Parameter sensitivity under different ϵ descent behavior. Subfigure (a) depicts the scenario with an initial ϵ value of 1, while Subfigure (b) illustrates the situation with an initial ϵ value of 0.8.

Ablation Study

To validate the effectiveness of the RAFHGL modules, we designed eight different ablation experiment methods. (1) *P*: Data prototypes are introduced to alleviate client drift stemming from random client selection. (2) *RA*: Built upon random client selection, it involves local training node filtration through ALA during training. (3) *RA+P*: Introduces ALA for local node selection while mitigating data distribution drift based on data prototypes, while excluding CSA from RAFHGL and randomly selecting clients. (4) *RA+CS*: Utilizes ALA to assess data node quality and perform selection, with CSA employed for server-side client selection, but without utilizing prototype-based loss correction to mitigate data distribution drift. (5) *LC*, (6) *E*, and (7) *MS* each incorporate active selection algorithms based on least confidence, information entropy, and marginal confidence, respectively, as alternatives to ALA within the RAFHGL framework. (8) *agg*, building upon RAFHGL, aggregates ALA while also aggregating local HGNN. Experimental results, as presented in Table 4, confirm significant performance advantages of



(a) Academic (b) DBLP

(c) ACM (d) Yelp

Figure 5: Performance on different quantiles n_q. Excessively small n_q hinder the comprehensive representation of client data quality, while excessively large n_q lead to redundancy, making it difficult to distinguish between different clients. The optimal n_q values are concentrated around 4 or 6.

RAFHGL compared to degradation methods, thereby validating the effectiveness of its combined component usage.

method	Acc	Rounds	method	Acc	Rounds
RAFHGL	**75.19**	**52**	RAFHGL	**75.19**	52
P	72.20	95	LC	73.83	**50**
RA	73.03	59	E	72.56	53
RA+P	74.03	65	MS	73.66	53
RA+CS	73.99	**52**	agg	74.21	84

Table 4: Ablation study of RAFHGL in terms of Accuracy (%) and convergence Rounds on DBLP.

Conclusion

In conclusion, this paper presents RAFHGL, a novel method aimed at overcoming challenges in CS for FHGL. By integrating RL and AL, RAFHGL dynamically adapts CS strategies during the FL process. The RL component CSA enables intelligent decision-making by considering delayed rewards. Concurrently, the introduction of AL on clients prioritizes nodes with higher perplexity in local training, allowing for a nuanced evaluation of client data quality by assessing the distribution of effective samples. Furthermore, we propose a local training correction algorithm based on data prototypes, which prevents node embeddings from deviating significantly from the prototype set. Experimental results across diverse federated learning scenarios highlight the effectiveness of RAFHGL. The method not only enhances model performance but also respects privacy constraints inherent in federated learning.

Acknowledgments

This work is supported in part by the National Key Research and Development Program of China (2023YFF0725103), National Natural Science Foundation of China(62192784,U23A20319,62422202,62272054), the 8th Young Elite Scientists Sponsorship Program by CAST (2022QNRC001).

References

An, W.; Tian, F.; Chen, P.; and Zheng, Q. 2022. Aspect-based sentiment analysis with heterogeneous graph neural network. *IEEE Transactions on Computational Social Systems*, 10(1): 403–412.

Balakrishnan, R.; Li, T.; Zhou, T.; Himayat, N.; Smith, V.; and Bilmes, J. 2022. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*.

Chai, Z.; Ali, A.; Zawad, S.; Truex, S.; Anwar, A.; Baracaldo, N.; Zhou, Y.; Ludwig, H.; Yan, F.; and Cheng, Y. 2020. Tifl: A tier-based federated learning system. In *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*, 125–136.

Chen, M.; Zhang, W.; Yuan, Z.; Jia, Y.; and Chen, H. 2021. Fede: Embedding knowledge graphs in federated setting. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs*, 80–88.

Chen, X.; Yu, G.; Wang, J.; Domeniconi, C.; Li, Z.; and Zhang, X. 2019. Activehne: Active heterogeneous network embedding. *arXiv preprint arXiv:1905.05659*.

Cho, Y. J.; Wang, J.; and Joshi, G. 2020. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*.

Dennis, D. K.; Li, T.; and Smith, V. 2021. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning*, 2611–2620. PMLR.

Fraboni, Y.; Vidal, R.; Kameni, L.; and Lorenzi, M. 2021. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *International Conference on Machine Learning*, 3407–3416. PMLR.

Fu, X.; and King, I. 2023. FedHGN: A Federated Framework for Heterogeneous Graph Neural Networks. *arXiv preprint arXiv:2305.09729*.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, 2331–2341.

Goetz, J.; Malik, K.; Bui, D.; Moon, S.; Liu, H.; and Kumar, A. 2019. Active federated learning. *arXiv preprint arXiv:1909.12641*.

Guan, Z.; Li, Y.; Pan, Z.; Liu, Y.; and Xue, Z. 2023. Rfdg: Reinforcement federated domain generalization. *IEEE Transactions on Knowledge and Data Engineering*.

Hu, B.; Fang, Y.; and Shi, C. 2019. Adversarial learning on heterogeneous information networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 120–129.

Hu, K.; Wu, J.; Li, Y.; Lu, M.; Weng, L.; and Xia, M. 2022. Fedgcn: Federated learning-based graph convolutional networks for non-euclidean spatial data. *Mathematics*, 10(6): 1000.

Huang, H.; Shi, W.; Feng, Y.; Niu, C.; Cheng, G.; Huang, J.; and Liu, Z. 2023. Active Client Selection for Clustered Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Hui Han, C. Y. H. Z. Y. L. X. W. C. S., Tianyu Zhao. 2022. OpenHGNN: An Open Source Toolkit for Heterogeneous Graph Neural Network. In *CIKM*.

Jiang, J.; Burkhalter, L.; Fu, F.; Ding, B.; Du, B.; Hithnawi, A.; Li, B.; and Zhang, C. 2022. Vf-ps: How to select important participants in vertical federated learning, efficiently and securely? *Advances in Neural Information Processing Systems*, 35: 2088–2101.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Lin, Y.; Chen, C.; Chen, C.; and Wang, L. 2020. Improving federated relational data modeling via basis alignment and weight penalty. *arXiv preprint arXiv:2011.11369*.

Liu, P.; Wang, L.; Ranjan, R.; He, G.; and Zhao, L. 2022a. A survey on active deep learning: from model driven to data driven. *ACM Computing Surveys (CSUR)*, 54(10s): 1–34.

Liu, R.; Xing, P.; Deng, Z.; Li, A.; Guan, C.; and Yu, H. 2022b. Federated graph neural networks: Overview, techniques and challenges. *arXiv preprint arXiv:2202.07256*.

Liu, Y.; Zhang, H.; Yang, C.; Li, A.; Ji, Y.; Zhang, L.; Li, T.; Yang, J.; Zhao, T.; Yang, J.; et al. 2023. Datasets and Interfaces for Benchmarking Heterogeneous Graph Neural Networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 5346–5350.

Long, Y.; Xue, Z.; Chu, L.; Zhang, T.; Wu, J.; Zang, Y.; and Du, J. 2023. Fedcd: A classifier debiased federated learning framework for non-iid data. In *Proceedings of the 31st ACM International Conference on Multimedia*, 8994–9002.

Lu, G.; Li, J.; and Wei, J. 2022. Aspect sentiment analysis with heterogeneous graph neural networks. *Information Processing & Management*, 59(4): 102953.

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

Mei, X.; Cai, X.; Yang, L.; and Wang, N. 2022. Relation-aware heterogeneous graph transformer based drug repurposing. *Expert Systems with Applications*, 190: 116165.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control

through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Paul, S.; Goyal, P.; and Ghosh, S. 2022. LeSICiN: A heterogeneous graph-based approach for automatic legal statute identification from Indian legal documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 11139–11146.

Peng, H.; Li, H.; Song, Y.; Zheng, V.; and Li, J. 2021. Differentially private federated knowledge graphs embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 1416–1425.

Qu, Z.; Duan, R.; Chen, L.; Xu, J.; Lu, Z.; and Liu, Y. 2022. Context-aware online client selection for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(12): 4353–4367.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B. B.; Chen, X.; and Wang, X. 2021. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9): 1–40.

Ribero, M.; and Vikalo, H. 2020. Communication-efficient federated learning via optimal client sampling. *arXiv preprint arXiv:2007.15197*.

Rjoub, G.; Wahab, O. A.; Bentahar, J.; Cohen, R.; and Bataineh, A. S. 2022. Trust-augmented deep reinforcement learning for federated learning client selection. *Information Systems Frontiers*, 1–18.

Shi, C.; and Shen, C. 2021. Federated multi-armed bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9603–9611.

Song, D.; Shen, G.; Gao, D.; Yang, L.; Zhou, X.; Pan, S.; Lou, W.; and Zhou, F. 2023. Fast heterogeneous federated learning with hybrid client selection. In *Uncertainty in Artificial Intelligence*, 2006–2015. PMLR.

Tan, Y.; Liu, Y.; Long, G.; Jiang, J.; Lu, Q.; and Zhang, C. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 9953–9961.

Tan, Y.; Long, G.; Liu, L.; Zhou, T.; Lu, Q.; Jiang, J.; and Zhang, C. 2022. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8432–8440.

Tang, M.; Ning, X.; Wang, Y.; Sun, J.; Wang, Y.; Li, H.; and Chen, Y. 2022. FedCor: Correlation-based active client selection strategy for heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10102–10111.

Tian, Y.; Dong, K.; Zhang, C.; Zhang, C.; and Chawla, N. V. 2023. Heterogeneous graph masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 9997–10005.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Wang, H.; Kaplan, Z.; Niu, D.; and Li, B. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, 1698–1707. IEEE.

Wang, L.; Guo, Y.; Lin, T.; and Tang, X. 2024. Delta: Diverse client sampling for fasting federated learning. *Advances in Neural Information Processing Systems*, 36.

Wang, M.; Zheng, D.; Ye, Z.; Gan, Q.; Li, M.; Song, X.; Zhou, J.; Ma, C.; Yu, L.; Gai, Y.; Xiao, T.; He, T.; Karypis, G.; Li, J.; and Zhang, Z. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315*.

Wang, X.; Bo, D.; Shi, C.; Fan, S.; Ye, Y.; and Philip, S. Y. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 9(2): 415–436.

Yan, B.; Cao, Y.; Wang, H.; Yang, W.; Du, J.; and Shi, C. 2023. Federated Heterogeneous Graph Neural Network for Privacy-preserving Recommendation. *arXiv preprint arXiv:2310.11730*.

Yang, J.; Liu, Y.; and Kassab, R. 2023. Client selection for federated bayesian learning. *IEEE Journal on Selected Areas in Communications*, 41(4): 915–928.

Yang, X.; Yan, M.; Pan, S.; Ye, X.; and Fan, D. 2023. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 10816–10824.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.

Zhan, Y.; Li, P.; and Guo, S. 2020. Experience-driven computational resource allocation of federated learning by deep reinforcement learning. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 234–243. IEEE.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 793–803.

Zhang, H.; Shen, T.; Wu, F.; Yin, M.; Yang, H.; and Wu, C. 2021. Federated graph learning–a position paper. *arXiv preprint arXiv:2105.11099*.

Zhang, K.; Wang, Y.; Wang, H.; Huang, L.; Yang, C.; Chen, X.; and Sun, L. 2022a. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. *arXiv preprint arXiv:2203.09553*.

Zhang, S. Q.; Lin, J.; and Zhang, Q. 2022. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9091–9099.

Zhang, Y.; Tong, H.; Xia, Y.; Zhu, Y.; Chi, Y.; and Ying, L. 2022b. Batch active learning with graph neural networks via multi-agent deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9118–9126.

Zhang, Y.; Xia, Y.; Zhu, Y.; Chi, Y.; Ying, L.; and Tong, H. 2022c. Active Heterogeneous Graph Neural Networks with Per-step Meta-Q-Learning. In *2022 IEEE International Conference on Data Mining (ICDM)*, 1329–1334. IEEE.