Toward Communication-Efficient Decentralized Federated Graph Learning Over Non-IID Data

Shilong Wang, Jianchun Liu, *Member, IEEE*, Hongli Xu, *Member, IEEE*, Chenxia Tang, Qianpiao Ma, and Liusheng Huang, *Member, IEEE*

***Abstract*—Decentralized Federated Graph Learning (DFGL) overcomes the potential bottlenecks of the parameter server in FGL. However, extensive cross-worker communication of graph node embeddings during DFGL training introduces substantial communication costs. To improve communication efficiency, constructing sparse network topologies or applying graph sampling are potential methods. In this paper, we first reveal the bidirectional coupling between network topology construction and graph sampling, underscoring the necessity of their joint optimization. Motivated by this insight, we propose DUPLEX, a unified framework that co-optimizes these two components by explicitly modeling their interdependent relationship, thereby significantly reducing communication costs while enhancing training performance in DFGL. DUPLEX formulates the decision-making process as a coordinated configuration $\langle \mathbf{A}, \mathbf{R} \rangle$, where $\mathbf{A}$ is the adjacency matrix of the network topology and $\mathbf{R}$ denotes the set of graph sampling ratios for workers. However, determining proper coordinated configurations to achieve optimal communication efficiency and training performance (e.g., model accuracy and convergence rate) is challenging due to several practical issues, *e.g.*, statistical heterogeneity and dynamic network conditions. To overcome these challenges, DUPLEX introduces a novel learning-driven algorithm to adaptively determine optimal network topologies and graph sampling ratios for workers. Experimental results demonstrate that DUPLEX reduces completion time by 20.1%–48.8% and communication costs by 16.7%–37.6% to achieve target accuracy, while improving accuracy by 3.3%–7.9% under identical resource budgets compared to baselines.**

***Index Terms*—Edge Computing, Federated Learning, Graph Neural Network, Topology Construction, Graph Sampling.**

Received 30 April 2025; revised 1 November 2025; accepted 1 December 2025. Date of publication 9 December 2025; date of current version 6 April 2026. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62472401, Grant 62132019, and Grant 624B2136, in part by the Jiangsu Province Science Foundation for Youths under Grant BK20230275, in part by the Anhui Province Science Foundation for Youths under Grant 2408085QF185, and in part by the Fundamental Research Funds for the Central Universities under Grant WK2150110033 and Grant WK2150250044. Recommended for acceptance by X. Peng. *(Corresponding authors: Jianchun Liu; Hongli Xu.)*

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

Qianpiao Ma is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 211111, China (e-mail: maqianpiao@njust.edu.cn).

Digital Object Identifier 10.1109/TMC.2025.3641696

I. Introduction

In many real-world applications, such as e-commerce recommendation systems [1] and social network analysis [2], graph-structured data, comprising interconnected nodes and edges, has emerged as a fundamental representation for modeling complex relationships. For instance, industry leaders like Amazon and Alibaba represent user preferences for items as user-item interaction graphs to enable personalized recommendations [3], [4]. To effectively process such data, Graph Convolutional Networks (GCNs) [5], [6], [7] have been proposed, demonstrating remarkable success in graph learning tasks. Fully realizing the potential of GCNs requires training on large volumes of graph data, which are continuously generated by users across a variety of edge devices (*e.g.*, smartphones and laptops). Such data are usually privacy-sensitive, thereby raising significant concerns when collected for training purposes [8], [9].

To address these privacy issues, Federated Graph Learning (FGL) [10], [11] has emerged as the de facto paradigm for collaboratively training GCNs across geo-distributed devices (i.e., workers) without exposing raw graph data. Traditional FGL frameworks typically adopt a Parameter Server (PS) architecture [12], in which massive workers communicate directly with the PS through wide area networks (WANs) [13], [14]. As the number of workers skyrockets, the ingress bandwidth of the PS becomes a major bottleneck, as all workers must deliver their local models to the server. As an alternative, Decentralized Federated Graph Learning (DFGL) [15], [16], [17] establishes a peer-to-peer (P2P) communication network among workers, enabling them to exchange local models with their neighbors. Since there is no need to transmit models from workers to the PS, the risk of a single point failure at the PS is eliminated, and system scalability is notably improved [18].

However, the practical deployment of DFGL systems presents formidable challenges due to the unique characteristics of distributed graph learning. For example, the Amazon product co-purchasing graph [19] consists of over 3 million nodes and 60 million edges, representing products and their co-purchasing relationships, respectively. In a DFGL framework, each worker (representing a merchant on Amazon) maintains a localized subgraph that encapsulates multiple products. These subgraphs include both *internal edges*, which connect products within a single worker, and *external edges* that span across different workers, indicating that users purchase multiple products simultaneously. The presence of external edges necessitates the

exchange of hidden states of product nodes (i.e., node embeddings) among workers during local GCN training (as detailed in Section II-B). However, the stark mismatch between the enormous size of exchanged node embeddings and the limited bandwidth available on workers poses significant challenges in achieving efficient DFGL. For instance, federated training of the popular GraphSage model [6] on the Amazon co-purchasing graph incurs hundreds of gigabytes of network traffic among workers. In contrast, typical WANs provide limited bandwidth ranging from 5 to 20 Mbps for workers [20], [21], which is markedly inferior to that available in data centers (*e.g.*, over 10 Gbps [22]). Consequently, the transmission of massive embedding data among workers often leads to prohibitively long communication times, thereby hindering the overall scalability and efficiency of DFGL systems.

Some recent studies have proposed various solutions to alleviate the high communication cost in DFGL. One common approach is network topology construction [16], [17]. For example, Liu et al. [17] introduced a novel framework named S-Glint, which constructs a sparse network topology for communication by selecting several neighbors with the highest performance contributions for each worker. By reducing the total number of communication links in DFGL, the overall communication overhead is mitigated. In addition to network topology construction, graph neighbor sampling [23], [24], [25] is another approach for alleviating communication overhead. Specifically, graph neighbor sampling randomly selects a subset of nodes, rather than all nodes, in the graph data for model training, thereby reducing the number of node embeddings exchanged among workers. For instance, Du et al. [24] analyzed the optimal sampling interval (i.e., the number of rounds between two adjacent sampling actions) to achieve the optimal trade-off between convergence rate and communication time.

Intuitively, integrating network topology construction and graph neighbor sampling offers potential for further enhancing communication efficiency in DFGL. Our convergence analysis in the Appendix formally establishes that both of them are first-order factors in the convergence bound of DFGL. This motivates us to investigate the benefits of combining these two distinct techniques. However, our preliminary experiments in Section II-C3 indicate that directly integrating these techniques may lead to significant degradation in training performance and yield suboptimal communication efficiency if they are not jointly optimized. In DFGL, these techniques are inherently coupled; that is, adjustments in one directly influence the other. For instance, under-sampling graph data in sparsely connected workers may lead to under-utilized bandwidth, while over-sampling in densely connected workers can result in communication redundancy. Furthermore, graph sampling affects the quality and diversity of information shared between workers. Without considering the communication network (*e.g.*, its topology), isolated sampling strategies may fail to preserve critical information required for model convergence. To address these issues, we propose DUPLEX, a unified framework that jointly optimizes the network topology and graph sampling, thereby improving both communication efficiency and training performance in DFGL. Specifically, DUPLEX accounts for the coupling between network topology construction and graph neighbor sampling by formulating the decision-making process as a coordinated configuration $\langle \mathbf{A}, \mathbf{R} \rangle$, where $\mathbf{A}$ is the adjacency matrix of the network topology and $\mathbf{R}$ denotes the set of graph sampling ratios (i.e., the proportions of graph nodes sampled for training) for workers.

However, achieving optimal communication efficiency and training performance (*e.g.*, model accuracy and convergence rate) with DUPLEX proves challenging due to several practical issues (Section II-D). **First**, statistical heterogeneity in DFGL adversely affects the training performance of DUPLEX. In real-world scenarios, user data on workers is typically generated under different contexts, such as personal usage patterns and geographic variations. Consequently, the local graph data across workers are non-Independent and Identically Distributed (non-IID) [26], [27]. In non-IID scenarios, the local model gradients computed on individual workers constitute biased estimates of the theoretically optimal global gradient computed over the collective data, resulting in a notable deterioration of training performance. **Second**, workers in DFGL usually connect to the network via wireless links, whose quality tends to fluctuate over time due to worker mobility and link instability [28]. Besides, concurrent applications on a worker may compete for limited bandwidth, further leading to dynamic fluctuations in the available network resources [26]. Due to varying network conditions, optimal configurations for network topology and sampling ratios may evolve as training progresses, necessitating adaptive adjustments rather than static solutions. To address these challenges, DUPLEX adopts consensus distance [29], [30], a metric that quantifies the discrepancy among local model parameters, to guide the decision-making process under non-IID conditions. Moreover, DUPLEX leverages a learning-driven algorithm based on deep reinforcement learning (DRL) to adaptively determine the optimal coordinated configuration $\langle \mathbf{A}, \mathbf{R} \rangle$ in response to dynamic network conditions. Our contributions are summarized as follows:

- We propose DUPLEX, an efficient DFGL framework that focuses on the joint optimization of network topology and graph sampling, aiming to reduce communication costs and improve training performance on non-IID graph data.
- We propose an effective algorithm to adaptively determine the optimal network topology and sampling ratios for workers, simultaneously considering dynamic network conditions and heterogeneous data distributions among workers.
- The extensive experimental results demonstrate that DUPLEX can reduce the completion time by 20.1%-48.8% and communication cost by 16.7%-37.6% to reach the target accuracy, while improving model accuracy by 3.3%-7.9% under the same resource budgets, compared to the baselines.

II. BACKGROUND AND MOTIVATION

In this section, we begin by outlining the fundamentals of GCNs and DFGL. Next, we identify key bottlenecks in DFGL and focus on the impact of network topology and graph sampling

ratios on training efficiency and model accuracy. Furthermore, we demonstrate that optimizing these components in isolation leads to suboptimal performance, thereby motivating the need for a joint optimization approach. Finally, we discuss the challenges in achieving optimal performance for DUPLEX, particularly in dynamic network environments with non-IID graph data.

A. Graph Convolutional Networks

Graph-structured data (*e.g.*, a citation network) represents entities (*e.g.*, documents) and their relationships (*e.g.*, citations) in a non-Euclidean space. Formally, such data can be represented as an undirected graph $G = (V, E)$, where V is a set of nodes representing entities and E is a set of edges representing relationships between nodes. Each node $v_i \in V$ is associated with a raw feature vector x_i. Graphs are irregular and non-Euclidean structures, posing challenges for traditional neural networks designed for grid-like data (*e.g.*, images and sequences). This necessitates specialized architectures such as GCNs, which generalize deep learning to graph domains by exploiting inductive biases tied to graph topology.

GCNs operate through iterative message-passing mechanisms [5], [6], where nodes aggregate information from their neighborhoods to refine their representations. Specifically, a GCN typically consists of L graph convolutional layers. Let $h_i^l \in \mathbb{R}^{d_l}$ denote the feature vector (i.e., embedding) of node v_i at layer l. The initial node feature vector h_i^0 is typically set to x_i. Each graph convolutional layer $l \in (1, 2, \ldots, L)$ is responsible for transforming $\{h_i^{l-1} \mid v_i \in V\}$ into $\{h_i^l \mid v_i \in V\}$ through the graph convolution (GC) operation, which involves two stages: (1) Aggregation: For each node v_i, gather features from its neighbors $\mathcal{N}(v_i)$. (2) Update: Update the embedding of node v_i by combining its previous state h_i^{l-1} with the aggregated neighborhood features. Mathematically, the GC operation at layer l is formalized as follows:

$$\begin{aligned} \mathscr{E}_i^l &= \text{AGG}(\{h_j^{l-1} \mid \forall v_j \in \mathcal{N}(v_i)\}), \\ h_i^l &= \mathbf{U}^l(h_i^{l-1} \| \mathscr{E}_i^l), \end{aligned} \tag{1}$$

where $\text{AGG}(\cdot)$ is the aggregation function (*e.g.*, element-wise mean or sum of vectors), $\mathbf{U}^l(\cdot)$ is the state update function at layer l (*e.g.*, a single-layer perceptron followed by a nonlinear transformation), and $\|$ denotes concatenation. Stacking L such layers enables nodes to integrate information from L-hop neighborhoods.

After propagating through L graph convolutional layers, the GCN makes predictions for downstream tasks as follows:

$$\hat{y}_{\mathcal{S}} = \mathbf{W}(\{h_i^L \mid \forall v_i \in \mathcal{S}\}), \tag{2}$$

where $\mathbf{W}(\cdot)$ can be a concatenation function followed by a multi-layer perceptron, and $\mathcal{S} \subseteq V$ is a node set used for prediction. Given a mini-batch of nodes $\mathcal{B}$ uniformly sampled from G, the training loss of a GCN is defined as:

$$\mathcal{L}_{\mathcal{B}} = F(\omega; y_{\mathcal{B}}, \hat{y}_{\mathcal{B}}), \tag{3}$$

where ω denotes the learnable parameters in the GCN, $y_{\mathcal{B}}$ is the set of labels for nodes in $\mathcal{B}$, and $F(\cdot)$ can be any feasible loss function (*e.g.*, Cross Entropy and L2-Norm). For brevity, we denote $F(\omega; y_{\mathcal{B}}, \hat{y}_{\mathcal{B}})$ simply as $F(\omega; \mathcal{B})$ hereafter.

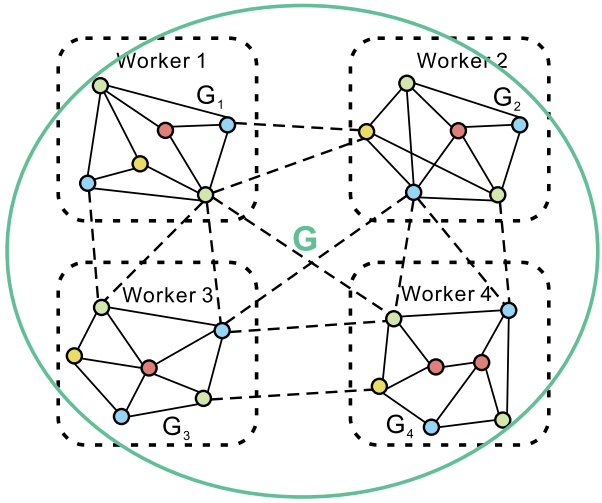


Fig. 1. Illustration of local subgraphs on workers, where the dashed lines between nodes represent the external edges across different subgraphs. The colors of nodes represent different node classes.

B. Decentralized Federated Graph Learning

In a network with a set of workers $\mathcal{M} = \{1, 2, \ldots, m\}$, each worker $i \in \mathcal{M}$ trains a local GCN model on its local subgraph G_i, which is assumed to be partitioned from an underlying global graph G, as shown in Fig. 1. Unlike other data types, such as images, that are typically independent across different workers, graph data inherently contains strong inter-dependencies manifested through external edges linking nodes across distinct subgraphs. Consequently, when performing GC operations, each worker must exchange node embeddings with others due to the existence of external edges. For instance, updating the embedding of a yellow node on worker 2 requires aggregating features from adjacent blue and green nodes that reside on worker 1. Neglecting such cross-worker interactions can lead to significant reductions in both the accuracy and convergence speed of the learned GCN models [16], [23], [24]. Moreover, in practical implementations of DFGL, the graph data across different workers is characterized by heterogeneous distributions due to factors such as varying personal usage patterns and geographic locations. For example, different companies often maintain extensive social network data that exhibits substantial heterogeneity (*e.g.*, diverse node classes and varying connection rules), stemming from different data collection methods and objectives.

Different from FGL, which relies on a PS to aggregate local models in each training round, DFGL employs a decentralized approach in which workers directly exchange their local GCN model parameters with neighboring workers after local training. This design effectively mitigates the risk of communication bottlenecks associated with a central PS. Let $f_i(\omega_i) := \mathbb{E}_{\mathcal{B}_i \sim G_i} F_i(\omega_i; \mathcal{B}_i)$ denote the local objective function of worker i, where $\mathcal{B}_i$ represents a mini-batch of nodes randomly sampled

TABLE I
RESOURCE (TIME AND NETWORK TRAFFIC) CONSUMPTION FOR LOCAL COMPUTING AND EXCHANGING LOCAL MODELS/NODE EMBEDDINGS AMONG WORKERS

Resource	Local Computing	Exchanging Local Models	Exchanging Node Embeddings
Time	313 s	115 s	3160 s
Network Traffic	N/A	0.91 GB	15.2 GB

from the local subgraph G_i, and ω_i is the local GCN model parameters of worker i. In general, the training process of DFGL is formulated as the optimization of the following objective function [31]:

$$f^* := \min_{\{\omega_i | i \in \mathcal{M}\}} \left[\frac{1}{m} \sum_{i \in \mathcal{M}} f_i(\omega_i) \right]. \tag{4}$$

This setting covers the important cases of empirical risk minimization in DFGL.

C. Motivations for Framework Design

Network topology and the number of graph node embeddings exchanged across workers during training affect both training performance and communication efficiency in DFGL. To gain a deeper understanding, we conduct several sets of experiments on 10 workers using the Reddit dataset [6], where each worker trains a two-layer GCN model [5]. Concretely, we set the number of neighbors per worker to 2 and 9 to simulate sparse and dense network topologies, respectively. We establish several configurations by combining the two network topologies (i.e., sparse and dense) with different graph sampling ratios (*e.g.*, 0.1, 0.5, and 1.0), which represent the proportions of aggregated neighboring node features for each node during the GC operation. The bandwidth for each worker fluctuates randomly between 1 and 20 Mbps [32], [33]. To simulate real-world non-IID graph data, we follow the prior work [34] and partition Reddit using the Dirichlet distribution $Dir(\alpha)$. In particular, we independently sample a subgraph $G_i \sim Dir_i(\alpha)$ from the global graph G and allocate G_i to each worker i, where α determines the degree of non-IID data. A lower value of α generates a higher label distribution shift.

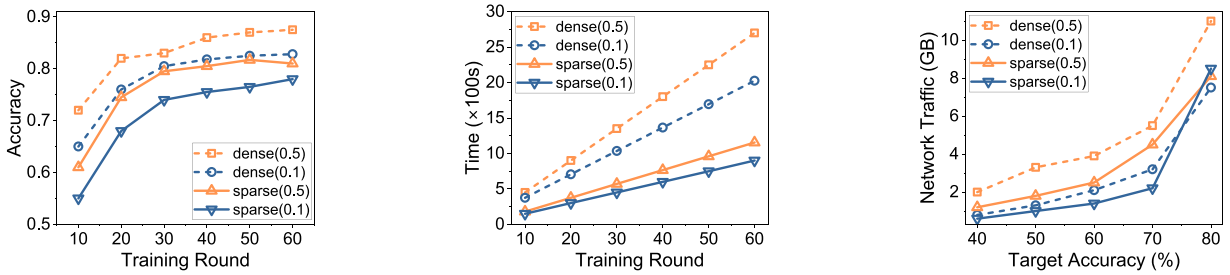
1) Communication Bottleneck in DFGL: The first set of experiments involves training the GCN model on a dense network topology with a graph sampling ratio of 1.0. We record the breakdown of total network traffic on all workers and the time required to reach a target training accuracy of 80%. The results in Table I show that the communication time for exchanging node embeddings among workers is over $10\times$ longer than the local computing time. Besides, since the size of a single GCN model is only approximately 0.5-2MB, the network traffic related to exchanging models among workers is negligible compared to that required for exchanging graph node embeddings. Therefore, it is critical to reduce both the communication time and the network traffic associated with exchanging graph node embeddings during training, as these represent major efficiency bottlenecks in DFGL.

2) Impact of Network Topology and Sampling Ratios: We further investigate the impact of network topology and graph sampling ratios on training efficiency and model accuracy. The experimental results in Fig. 2(a) indicate that a denser network topology combined with a higher graph sampling ratio improves model accuracy. For example, on the dense topology, training with a sampling ratio of 0.5 enhances the accuracy by 5.1% compared to using a ratio of 0.1. On the other hand, under the same sampling ratio (*e.g.*, 0.5), the accuracy on the dense topology increases by 6.5% compared to that on the sparse topology. However, these accuracy improvements come at a cost: resource consumption, such as training time and network traffic, increases significantly with denser network topologies and higher sampling ratios, as shown in Fig. 2(b) and (c). Therefore, it is imperative for DFGL frameworks to optimize network topologies and graph sampling strategies, as they critically mediate the trade-off between training efficiency and model performance.

3) Importance of Joint Optimization for Network Topology and Sampling Ratio: Directly combining existing techniques for network topology construction and graph neighbor sampling will lead to significant performance degradation, as the two techniques are interdependent yet not jointly optimized. In DFGL, network topology and graph sampling ratio jointly impact the volume of node embeddings exchanged across workers during training, which determines the overall communication cost. Furthermore, varying network topologies change the interconnection schemes among local subgraphs on workers, leading to distinct optimal graph sampling strategies for each configuration. For instance, a dense topology may amplify the benefits of high sampling ratios by increasing the shared information, but it incurs significant communication overhead. Besides, a sparse topology can reduce communication costs but might diminish the benefits of high sampling ratios due to reduced data diversity and quality.

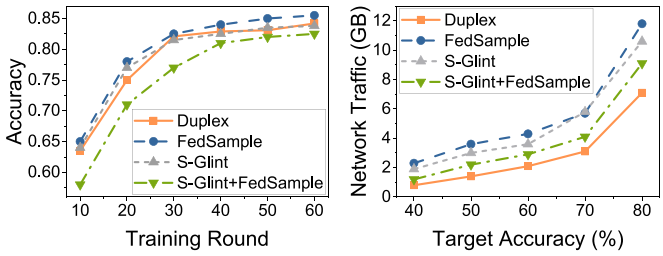
Existing works always assume that sampling strategies are agnostic to network topology, treating communication as a homogeneous process among workers. Consequently, prior methods optimize *either* network topology *or* graph sampling in isolation, with no framework jointly addressing both at the same time. However, in P2P networks, communication patterns inherently vary with topology, and sampling strategies that ignore this interdependence can result in inefficiencies and degraded training performance. Therefore, jointly optimization of network topology and graph sampling ratios is crucial yet challenging in DFGL. To validate this insight, we conduct experiments to compare the performance of four approaches: 1) S-Glint [17], which constructs a sparse network topology without graph neighbor sampling, 2) FedSample [23], which assigns appropriate graph sampling ratios to workers without considering the impact of network topology, 3) S-Glint+FedSample, which is a direct combination of S-Glint and FedSample, optimizing network topology and sampling ratios separately, and 4) DUPLEX, which coordinates topology construction and graph sampling through a unified decision-making process.

The results presented in Fig. 3(a) clearly demonstrate that DUPLEX, S-Glint, and FedSample achieve superior training



(a) Model accuracy ($\alpha = 10.0$). (b) Training Time ($\alpha = 10.0$). (c) Communication Cost ($\alpha = 10.0$).

Fig. 2. Impact of network topology/graph sampling ratio on accuracy and resource consumption of model training.



(a) Model accuracy ($\alpha = 10.0$). (b) Communication cost ($\alpha = 10.0$).

Fig. 3. Performance comparison of four methods on Reddit.

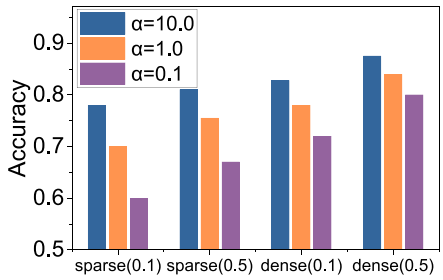


Fig. 4. Harm of Non-IID data on model accuracy.

performance compared to S-Glint+FedSample. For example, at training round 60, DUPLEX, S-Glint and FedSample improve model accuracy by 2.1%, 1.8% and 3.6%, respectively, over S-Glint+FedSample. Besides, as shown in Fig. 3(b), while S-Glint+FedSample reduces communication overhead in DFGL compared to S-Glint or FedSample, significant optimization space remains. By jointly optimizing network topology and graph sampling ratios, DUPLEX reduces network traffic by 22.1% compared to S-Glint+FedSample. These results highlight the importance of joint optimization in leveraging the benefits of both network topology construction and graph sampling in DFGL.

D. Practical Challenges in DUPLEX

In DFGL, graph data distributed across workers is usually non-IID, which significantly degrades training performance. To illustrate this challenge, we conduct experiments using three non-IID settings, i.e., $\alpha = 0.1$ (high non-IIDness), $\alpha = 1.0$ (middle non-IIDness), and $\alpha = 10.0$ (low non-IIDness). As shown in Fig. 4, model accuracy declines sharply as non-IIDness increases from low to high. For example, on the sparse topology with a graph sampling ratio of 0.5, model accuracy decreases from 81.2% to 67.1%. Furthermore, the extent of accuracy degradation varies with network topology and graph sampling ratios. For example, dense topologies with higher sampling ratios exhibit greater resilience to non-IID data compared to sparse topologies with lower ratios. These findings underscore that network topologies and sampling strategies differentially influence robustness to data heterogeneity. Therefore, optimizing both components in DUPLEX is critical to improving training performance on non-IID data.

Besides, due to workers' mobility, communication link instability, and bandwidth competition among multiple applications, the available bandwidth for each worker varies dynamically over time [26]. Consequently, the fixed network topology and graph sampling ratio may not be efficient all the time as training progresses under dynamic network conditions. Therefore, it is crucial yet challenging for DUPLEX to adaptively adjust the configurations $\langle \mathbf{A}, \mathbf{R} \rangle$ in response to time-varying communication speeds of workers.

III. FRAMEWORK DESIGN

This section outlines the design of DUPLEX and elaborates on how it enhances communication efficiency and training performance in DFGL, especially under dynamic network conditions and non-IID graph data. First, we provide an overview of DUPLEX in Section III-A. Next, we detail the methods employed for the joint optimization of network topology and graph sampling ratios (Section III-B). In addition, we describe the procedures for local GCN training (Section III-C) and model aggregation (Section III-D), both of which are executed based on the optimized network topology and sampling ratios.

A. Overview of DUPLEX

We consider a typical DFGL setting comprising a set of workers $\mathcal{M} = \{1, 2, \ldots, m\}$, where each worker $i \in \mathcal{M}$ maintains a local subgraph $G_i = (V_i, E_i)$ as part of the global graph $G = (V, E)$. Besides, a logical control worker (i.e., the coordinator) is responsible for collecting global information about training statuses and network conditions [35], [36]. Note

that the coordinator differs significantly from the PS in FGL because it does not aggregate local models, and hence will not become the communication bottleneck. Since the size of such information (*e.g.*, 100–300 KB [37]) is much smaller than that of model parameters, it is reasonable to ignore the associated costs (*e.g.*, network traffic and communication time) for information collection. Furthermore, any worker can act as the coordinator. The training procedure of DUPLEX involves multiple rounds, each encapsulating three main steps: ① **Configuration Update.** At the beginning of each round, the coordinator adaptively constructs the network topology and determines the graph sampling ratios according to the information collected from workers (*e.g.*, training statuses and network conditions). ② **Local GCN Training.** After receiving the updated configurations of topology and ratios from the coordinator, each worker iteratively updates the local GCN model over its local subgraph for τ iterations. In each iteration, workers sample a subset of graph nodes according to their respective graph sampling ratios for stochastic gradient descent, rather than using the full set of graph nodes. ③ **Model Aggregation.** Once local GCN training is completed, each worker exchanges its local GCN model parameters with its neighbors in the constructed network topology. Finally, each worker aggregates the received model parameters from its neighbors.

B. Configuration Update

We first introduce the consensus distance, which is a crucial metric in the coordinator's decision-making process on non-IID graph data (Section III-B1). Then, we present the problem of joint optimization for ne**T**work t**O**pology and graph sa**M**pling r**A**tio**S**, termed TOMAS, aiming to minimize communication cost while ensuring satisfactory training performance in dynamic network environments (Section III-B2). Finally, we design an efficient algorithm to solve this problem in Sections III-B3 and III-B4.

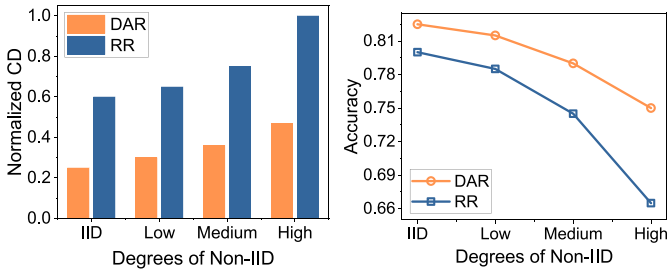
1) Consensus Distance: Since the graph data on workers is non-IID, local GCN models trained on different workers exhibit significant heterogeneity. To guide the coordinator in jointly optimizing the network topology and graph sampling ratios, we introduce the consensus distance [29], [30], a key metric in DUPLEX for quantifying the heterogeneity of data distribution through the Euclidean distance among local models. Specifically, worker i's consensus distance at round k is defined as:

$$C_i^{(k)} = \left\| \omega_i^{(k)} - \overline{\omega}^{(k)} \right\|_2, \tag{5}$$

where $\omega_i^{(k)}$ represents the local model parameters of worker i, and $\overline{\omega}^{(k)} = \frac{1}{m}\sum_{j=1}^{m} \omega_j^{(k)}$ denotes the average of all workers' model parameters. The global consensus distance at round k is then:

$$C^{(k)} = \frac{1}{m}\sum_{i=1}^{m} C_i^{(k)}. \tag{6}$$

Analogous to weight divergence in the PS architectures, consensus distance correlates with data distribution skewness [38].



(a) Global consensus distance (CD). (b) Model accuracy.

Fig. 5. Global consensus distance and model accuracy on the random ring (RR) and the distribution-aware ring (DAR) topology with different non-IID degrees.

Minimizing $C^{(k)}$ can improve training performance (*e.g.*, model accuracy and convergence rate) by reducing discrepancies between local models [39].

In DFGL with non-IID data, workers exchange *graph node embeddings* and *local GCN parameters* with their neighbors to achieve global consensus. Consequently, the global consensus distance becomes a critical metric for capturing the combined impact of network topology and graph sampling ratios. A natural method to minimize this consensus distance is to prioritize exchanges of model parameters and node embeddings between workers with significant pairwise Euclidean distances in their model parameters, thereby improving training performance. To validate this insight, we follow the experimental settings in Section II-C, training GraphSage under two distinct network topologies, i.e., random ring and distribution-aware ring, respectively. The random ring topology *randomly* arranges workers in a ring structure, with each worker assigned a fixed graph sampling ratio of 0.5. On the distribution-aware ring topology, workers are greedily connected in each round to the neighbor with the largest pairwise Euclidean distance between their local model parameters. Each worker i is dynamically assigned a sampling ratio of $\frac{0.5C_i^{(k)}}{C^{(k)}}$. We simulate non-IID data using Dirichlet distribution parameters $\alpha = \{10.0, 1.0, 0.1\}$ to represent low, medium, and high skewness levels. As shown in Fig. 5(a), the distribution-aware ring topology achieves an over 50% lower global consensus distance compared to the random ring topology. Furthermore, Fig. 5(b) demonstrates that the distribution-aware approach improves model accuracy by 3.1%–8.5% across various non-IID settings while maintaining robustness under high non-IID conditions ($\alpha = 0.1$). These results underscore the necessity of jointly optimizing network topology and sampling ratios guided by consensus distance in DFGL.

2) Problem Formulation: The P2P network topology in DUPLEX at round k is represented by a symmetric adjacency matrix $\mathbf{A}^{(k)} = [a_{i,j}^{(k)} \in \{0,1\}]_{m\times m}$, where $a_{i,j}^{(k)} = 1$ indicates that worker i and j can communicate with each other at round k, and $a_{i,j}^{(k)} = 0$ otherwise. The neighbor set of worker i at round k is defined as $\mathcal{N}_i^{(k)} = \{j \in \mathcal{M} \mid a_{i,j}^{(k)} = 1\}$. For graph data, let $\mathcal{N}(v)$ denote the one-hop neighbors of node $v \in V_i$ (where V_i is the node set of worker i), and let $\mathcal{S}(v) \subseteq \mathcal{N}(v)$ represent the subset of nodes sampled for GCN training. The graph sampling

ratio r_i for worker i is defined as the average sampling rate across all nodes in V_i:

$$r_i = \frac{1}{|V_i|} \sum_{v \in V_i} \frac{|\mathcal{S}(v)|}{|\mathcal{N}(v)|}. \tag{7}$$

In DFGL, the bandwidth available to each worker is limited and fluctuates dynamically over time due to communication capacity constraints and bandwidth competition among multiple applications. We denote the bandwidth for worker i at round k as $b_i^{(k)} = (b_i^{(k,in)}, b_i^{(k,out)})$, where $b_i^{(k,in)}$ and $b_i^{(k,out)}$ represent the inbound and outbound bandwidths, respectively. For simplicity, all communication links connected to a worker share the bandwidth equally [26]. The bandwidth of a link between worker i and worker j is the minimum of the link's inbound and outbound bandwidth:

$$b_{i,j}^{(k)} = \min \left\{ \frac{b_i^{(k,out)}}{|\mathcal{N}_i^{(k)}|}, \frac{b_j^{(k,in)}}{|\mathcal{N}_j^{(k)}|} \right\}. \tag{8}$$

Due to link asymmetry, note that the bandwidth $b_{i,j}^{(k)}$ and $b_{j,i}^{(k)}$ may be different.

When workers train GCN models synchronously, the global round time, which is the duration between two successive rounds, is given by:

$$t^{(k)} = \max_{i \in M} t_i^{(k)}, \tag{9}$$

where $t_i^{(k)}$ is the round time for worker i at round k. This time comprises $t_i^{(k)} = t_i^{(k,cp)} + t_i^{(k,com)}$, where $t_i^{(k,cp)}$ is the local computing time for GCN model updating and $t_i^{(k,com)}$ is the communication time with neighboring workers. As demonstrated in Section II-C1, round time $t_i^{(k)}$ is dominated by communication time $t_i^{(k,com)}$. In particular, $t_i^{(k,com)}$ can be calculated as follows:

$$t_i^{(k,com)} = \max_{j \in \mathcal{N}_i^{(k)}} \frac{r_i^{(k)} \cdot \mathcal{E}_{i,j}}{b_{i,j}^{(k)}} + \max_{j \in \mathcal{N}_i^{(k)}} \frac{|\omega|}{b_{i,j}^{(k)}}, \tag{10}$$

where $r_i^{(k)}$ is worker i's sampling ratio at round k, $\mathcal{E}_{i,j}$ is the total size of node embeddings sent from worker i to worker j (assuming no graph sampling), and $|\omega|$ is the size of GCN model parameters. The communication time $t_i^{(k,com)}$ depends on both the network topology $\mathbf{A}^{(k)}$ (which impacts bandwidth $b_{i,j}^{(k)}$) and the graph sampling ratio $r_i^{(k)}$. Therefore, joint optimization of these components is crucial for enhancing communication efficiency. The goal of this joint optimization is also motivated by our convergence analysis in the Appendix (Theorem 1), which identifies both the network topology and sampling strategy as first-order factors in the convergence bound. Accordingly, we formulate the TOMAS problem as follows:

$$\min_{\{\langle \mathbf{A}^{(k)}, \mathbf{R}^{(k)} \rangle | k \in [1,K]\}} \sum_{k=1}^{K} t^{(k)}$$

$$\text{s.t.} \begin{cases} C^{(k)} \le C_{\max}^{(k)}, & \forall k \\ \frac{1}{m} \sum_{i \in \mathcal{M}} f_i(\omega_i^{(K)}) \le \mathcal{F}, & \\ a_{i,j}^{(k)} \in \{0,1\}, & \forall i,j \in \mathcal{M}, \forall k \\ 0 < r_i^{(k)} \le 1, & \forall i \in \mathcal{M}, \forall k \end{cases} \tag{11}$$

where K is the total number of training rounds, and $\langle \mathbf{A}^{(k)}, \mathbf{R}^{(k)} \rangle$ denotes the coordinated configuration of the network topology and graph sampling ratios for workers at round k. The first inequality ensures the global consensus distance $C^{(k)}$ at each round k does not exceed the threshold $C_{\max}^{(k)}$ ($C_{\max}^{(k)}$ is given in Section III-B3). The second inequality guarantees the convergence of model training, where $\frac{1}{m} \sum_{i \in \mathcal{M}} f_i(\omega_i^{(K)})$ is the average training loss of all workers at the final round K and $\mathcal{F}$ is a convergence threshold of the loss value. The objective is to minimize the total completion time $\sum_{k=1}^{K} t^{(k)}$ of model training.

In fact, the problem formulated in (11) is hard to solve directly, since it is a nonlinear mixed integer programming problem [40]. Moreover, deriving precise closed-form expressions to describe the influence of network topology and graph sampling ratios on the convergence bound presented in the Appendix is intractable. Hence, instead of designing a heuristic algorithm, we resort to a learning-driven approach to jointly optimize network topology and graph sampling ratios. We believe the Deep Reinforcement Learning (DRL) based method would be effective in solving the TOMAS problem, since DRL can effectively perceive complex relationships between dynamic network states and training performance. However, due to the varied ways of DRL for different problems, it is crucial to adopt a suitable DRL method for the TOMAS problem. By carefully comparing potential DRL methods, we select Deep Deterministic Policy Gradient (DDPG) [41], as it can efficiently and effectively handle continuous action spaces, which aligns well with the continuity of the graph sampling ratio. Next, we design a customized DDPG agent tailored to TOMAS's requirements (Section III-B3) and describe the methodology of the learning-driven algorithm (Section III-B4).

3) DDPG Agent Design: The DDPG agent in DUPLEX consists of three parts, i.e., state space, action space, and reward function.

State space: The agent state at round k is defined as $s^{(k)} = (\mathbf{b}^{(k)}, \mathbf{T}^{(k)}, \mathcal{E}^{(k)}, \mathcal{C}^{(k)}, \mathbf{F}^{(k)})$. Here, $\mathbf{b}^{(k)} = \{b_i^{(k)} \mid \forall i \in \mathcal{M}\}$ and $\mathbf{T}^{(k)} = \{t_i^{(k)} \mid \forall i \in \mathcal{M}\}$ represent the available bandwidth and round time for all workers at round k, respectively. The set $\mathcal{E}^{(k)} = \{\mathcal{E}_{i,j}^{(k)} \mid \forall i,j \in \mathcal{M}\}$ denotes the sizes of node embeddings exchanged between any two workers at round k (with graph sampling). $\mathcal{C}^{(k)} = \{C_{i,j}^{(k)} \mid \forall i,j \in \mathcal{M}\}$ represents the Euclidean distances between all pairs of local model parameters at round k, where $C_{i,j}^{(k)} = \|\omega_i^{(k)} - \omega_j^{(k)}\|_2$. Finally, $\mathbf{F}^{(k)} = \{f_i(\omega_i^{(k)}) \mid \forall i \in \mathcal{M}\}$ denotes the local training losses on all workers at round k.

Action space: The action space at round k can be represented as the coordinated configuration $\sigma^{(k)} = \langle \mathbf{A}^{(k)}, \mathbf{R}^{(k)} \rangle$, where $\mathbf{A}^{(k)}$ denotes the adjacency matrix of the network topology and $\mathbf{R}^{(k)} = \{r_i^{(k)} \mid \forall i \in \mathcal{M}\}$ denotes the set of graph sampling ratios for all workers at round k.

Reward function: At round k, the coordinator computes a reward $u^{(k)}$ after executing the action $\sigma^{(k)}$. The reward should be positively correlated with the system objective [42]. That is, with less round time, smaller consensus distance, and lower training loss, the reward $u^{(k)}$ gotten would be greater. We define the reward function as:

$$u^{(k)} = -\chi\left(\frac{t^{(k)}}{\bar{t}^{(k-1)}} - 1\right) + \varrho\left(C_{\max}^{(k)} - C^{(k)}\right) + \varphi^{\left(\mathcal{F}-\bar{f}^{(k)}\right)}, \tag{12}$$

where χ, ϱ, φ are three positive constants. $\bar{t}^{(k)}$ is the moving average time used to alleviate the impact of data jitter:

$$\bar{t}^{(k)} = \Upsilon t^{(k)} + (1-\Upsilon)\bar{t}^{(k-1)}, \tag{13}$$

where $\Upsilon \in (0,1)$. $\bar{f}^{(k)}$ denotes the average local training loss of workers at round k. $C_{\max}^{(k)}$ is the exponential moving average of the gradient norm [29]:

$$C_{\max}^{(k)} = (1-\beta)C_{\max}^{(k-1)} + \frac{\beta}{m}\sum_{i=1}^{m}\left\|g_i^{(k)}\right\|_2, \tag{14}$$

where $\frac{1}{m}\sum_{i=1}^{m}\|g_i^{(k)}\|_2$ denotes the average gradient norm of local model updates at round k, and $\beta \in [0,1]$. The first term of the reward function encourages fast training. The longer the training time the round k takes, the less reward $u^{(k)}$ will be obtained. The second and third terms evaluate the descent of consensus distance and training loss, respectively. The closer to the given threshold of consensus distance $C_{\max}^{(k)}$ and loss value $\mathcal{F}$, the more reward will be received. It is worth noting that the actual $\overline{\omega}^{(k)}$ in (5) is not available in practice. Thus, we follow [26] to estimate the global consensus distance $C^{(k)}$ as follows:

$$\hat{C}^{(k)} = \frac{1}{m^2}\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}}\left(1-a_{i,j}^{(k)}\right)\hat{C}_{i,j}^{(k)}, \tag{15}$$

where $\hat{C}_{i,j}^{(k)} = \min_{q\in\mathcal{M}\setminus\{i,j\}}(C_{i,q}^{(k)} + C_{j,q}^{(k)})$ is the estimated Euclidean distance between worker i and j's local model parameters.

4) Learning-Driven Algorithm: The basic idea of our algorithm is to maintain an actor network $\pi(s|\theta^{\pi})$ and a critic network $Q(s,\sigma|\theta^{Q})$ with parameters θ^{π} and θ^{Q}, respectively. Meanwhile, we maintain two target networks $\pi'(s|\theta^{\pi'})$ and $Q'(s,\sigma|\theta^{Q'})$, which have the same structures as $\pi(s|\theta^{\pi})$ and $Q(s,\sigma|\theta^{Q})$. The actor network and the critic network can be implemented using a Deep Neural Network and a Deep Q-Network [43], respectively. Additionally, the replay buffer B is employed to store historical transitions defined as $(s^{(k)},\sigma^{(k)},u^{(k)},s^{(k+1)})$. We update the actor network and the critic network using a mini-batch of transitions sampled from the replay buffer. Given a state $s^{(k)}$, the actor network generates an action:

$$\sigma^{(k)} = \pi\left(s^{(k)}|\theta^{\pi}\right). \tag{16}$$

Then, the critic network would return a Q value $Q(s^{(k)},\sigma^{(k)}|\theta^{Q})$ when provided with a state $s^{(k)}$ and an action $\sigma^{(k)}$ as input. In order to update the critic network, we should also calculate the target Q value:

$$y^{(k)} = u^{(k)} + \gamma Q'\left(s^{(k+1)}, \pi'\left(s^{(k+1)}|\theta^{\pi'}\right)|\theta^{Q'}\right), \tag{17}$$

where γ is the discount factor for the future rewards. Then, the Temporal-Difference error (TD-error) is obtained:

$$\delta^{(k)} = y^{(k)} - Q\left(s^{(k)},\sigma^{(k)}|\theta^{Q}\right). \tag{18}$$

The critic network can be updated using the TD-error through gradient descent. Besides, to update the actor network, we should obtain the policy gradient as:

$$\nabla_{\theta^{\pi}}Q(s^{(k)},\sigma^{(k)}) = \nabla_{\sigma}Q(s,\sigma|\theta^{Q})|_{u^{(k)}}^{s^{(k)}} \cdot \nabla_{\theta^{\pi}}\pi(s|\theta^{\pi})|_{s^{(k)}}. \tag{19}$$

Accordingly, the actor network can be updated using the policy gradient with gradient ascent.

In each round, the coordinator samples transitions $(s^{(k)},\sigma^{(k)},u^{(k)},s^{(k+1)})$ from the buffer to update the actor network and the critic network. The accumulated weight change for the actor network and the critic network is:

$$\begin{cases}\Delta_{\theta^{\pi}} := \Delta_{\theta^{\pi}} + \frac{1}{|B|}\cdot\nabla_{\theta^{\pi}}Q(s^{(k)},\sigma^{(k)}),\\ \Delta_{\theta^{Q}} := \Delta_{\theta^{Q}} + \frac{1}{|B|}\cdot\delta^{(k)}\cdot\nabla_{\theta^{Q}}Q(s^{(k)},\sigma^{(k)}),\end{cases} \tag{20}$$

where $\nabla_{\theta^{\pi}}Q(s^{(k)},\sigma^{(k)})$ is policy gradient defined in (19) and $\delta^{(k)}$ is TD-error defined in (18). The adaptive control algorithm in DUPLEX is presented in Algorithm 1. At the beginning of the algorithm, the coordinator first initializes the parameters of the actor network and the critic network. To apply an off-policy training method, the target networks $Q'(s,\sigma|\theta^{Q'})$, $\pi'(s|\theta^{\pi'})$, and a replay buffer are employed to improve the training speed. Subsequently, the coordinator awaits the reception of the initial state from workers (Line 2). At each training round k, the coordinator firstly obtains the action (i.e., sampling ratios and

Algorithm 1: Control Algorithm on Coordinator.

1: Initialize actor network π, critic network Q, target networks π' and Q' and replay buffer B;
2: Receive initial state $s^{(1)}$ from workers;
3: **for** $k = 1$ to K **do**
4: Obtain action $\sigma^{(k)}$ by (16);
5: Send action $\sigma^{(k)}$ to workers;
6: Receive $s^{(k+1)}$ from workers;
7: Obtain reward $u^{(k)}$ by (12);
8: Put transition $(s^{(k)},\sigma^{(k)},u^{(k)},s^{(k+1)})$ into B;
9: **for** $t = 1$ to N **do**
10: Sample a mini-batch of transitions from B;
11: Compute target Q value by (17);
12: Compute TD-error by (18);
13: Compute policy gradient by (19);
14: Accumulate weight-change for actor network and critic network by Eq.(20);
15: **end**
16: Update the actor network and the critic network;
17: Update the target networks by (21);
18: **end**

Algorithm 2: Procedure at Worker i.

1: **for** $k = 1$ to K **do**
2: Receive $\mathcal{N}_i^{(k)}$, $r_i^{(k)}$ from the coordinator;
3: Perform local training by `LocalTraining()`;
4: Send (receive) models to (from) neighbors $\mathcal{N}_i^{(k)}$;
5: Aggregate models and obtain $\omega_i^{(k+1)}$ by (23);
6: Compute consensus distance $C_{i,j}^{(k)}, \forall j \in \mathcal{N}_i^{(k)}$;
7: Send consensus distance and loss to the coordinator;
8: **end**
9: FnFunction: `LocalTraining ()` **for** each training iteration **do**
10: Randomly sample a mini-batch of nodes $\mathcal{B}_i \subset V_i$;
11: $\{V_i^l\}, \{\mathcal{S}^l(v)\}$=`GraphSampling` $(\mathcal{B}_i, r_i^{(k)})$;
12: **for** $l = 1$ to L **do**
13: **for** each node $v \in V_i^l$ **do**
14: $\mathscr{o}_v^l = \text{AGG}(\{h_u^{l-1} \mid u \in \mathcal{S}^l(v)\})$;
15: $h_v^l = \mathbf{U}^l(h_v^{l-1} \| \mathscr{o}_v^l)$;
16: **end**
17: **end**
18: Compute training loss $\mathcal{L}_{\mathcal{B}_i}$ by (2) and (3);
19: Compute model gradient and update GCN model parameters by gradient descent;
20: **end**
21: FnFunction: `GraphSampling` $(\mathcal{B}_i, r_i)$ Initialize node set $V_i^L = \mathcal{B}_i$;
22: **for** $l = L$ to 1 **do**
23: Initialize node set $V_i^{l-1} = V_i^l$;
24: **for** each node $v \in V_i^l$ **do**
25: Randomly sample a subset of nodes $\mathcal{S}^l(v)$ from v's neighbors $\mathcal{N}(v)$ based on $r_i^{(k)}$;
26: Update node set $V_i^{l-1} = V_i^{l-1} \cup \mathcal{S}^l(v)$;
27: **end**
28: **end**
29: **return** $\{V_i^l \mid l \in [1, L]\}, \{\mathcal{S}^l(v) \mid l \in [1, L]\}$;

network topology) $\sigma^{(k)}$ by (16) (Line 4) and sends the action to workers (Line 5). Based on the received action, each worker trains the local GCN model (Section III-C) and aggregates models from neighbors (Section III-D). Next, a new state $s^{(k+1)}$ is sent from workers to the coordinator (Line 6), and the reward associated with the action is obtained (Line 7). Following this, the coordinator samples the transition data from the replay buffer to train the actor and critic networks (Lines 8-16). Specifically, the target Q value and TD-error are computed by (17) and (18), respectively (Lines 11-12). Then, the policy gradient is calculated by the chain rule (Line 13). Concurrently, the weight changes are accumulated for updating the actor network and the critic network (Lines 14-16). Last, the target actor network and critic network are updated as follows (Line 17):

$$\begin{cases} \theta^{\pi'} = \xi\theta^{\pi} + (1-\xi)\theta^{\pi'}, \\ \theta^{Q'} = \xi\theta^{Q} + (1-\xi)\theta^{Q'}, \end{cases} \tag{21}$$

where $\xi \in (0, 1]$ is the update coefficient of target networks.

C. Local GCN Training

After receiving updated configurations of network topology and graph sampling ratios from the coordinator, each worker $i \in \mathcal{M}$ iteratively updates its local GCN model on its local subgraph via stochastic gradient descent (SGD) for τ iterations. The local GCN training procedure for each worker i is detailed in Algorithm 2.

Within each iteration, worker i first randomly samples a mini-batch of graph nodes $\mathcal{B}_i$ from its local subgraph G_i (Line 10). It then performs graph sampling to randomly select a subset for L-hop neighbors (for an L-layer GCN) for each node in $\mathcal{B}_i$, guided by the sampling ratio $r_i^{(k)}$ (Lines 11, 8-29). These sampled nodes are used for training the local GCN model. Fig. 6(a)–(c) illustrate this graph sampling process for various sampling ratios under isolated conditions (*e.g.*, no inter-worker communication). Specifically, given an L-layer GCN, all nodes in $\mathcal{B}_i$ constitute the node set at layer L (Line 8). From layer L to layer 1, a subset of neighbors for each node in the current layer is randomly sampled to populate the node set for the preceding layer (Lines 22-26). However, due to the existence of external graph edges connecting nodes across workers (*e.g.*, black dotted lines in Fig. 7), the sampled graph nodes by worker i may reside on other workers. As shown in Fig. 7, considering a system with three workers, Worker 1 and Worker 2 are neighbors in the network topology while Worker 3 is isolated. Worker 1 initially samples nodes B and C (connected to node A) during graph sampling. Subsequently, the 2-hop neighbors of node A (i.e., nodes D, E, and F on Worker 2) are further sampled. Since Worker 3 is non-adjacent to Worker 1, it contributes no nodes to this sampling process.

Upon completing the graph sampling, each worker iteratively transforms low-level node embeddings into higher-level ones from layer 1 to layer L using GC operations defined in (1) (Lines 12-17). Since the sampled node set $\mathcal{S}^l(v)$ may include nodes from worker i's neighboring workers, worker i should request corresponding node embeddings from those workers. After performing the GC operation on the top layer L, the training loss $\mathcal{B}_i$ for the mini-batch $\mathcal{B}_i$ is computed using (2) and (3) (Line 18). Finally, the local GCN model parameters $\omega_i^{(k)}$ are updated through gradient descent (Line 19):

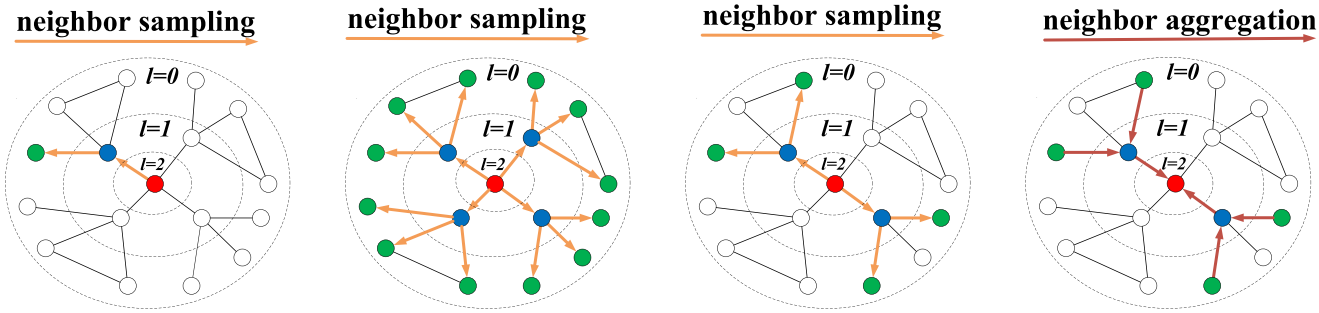
$$\omega_i^{(k)} = \omega_i^{(k)} - \eta \cdot \nabla_{\omega_i^{(k)}} \mathcal{L}_{\mathcal{B}_i}, \tag{22}$$

where η is the learning rate, and $\nabla_{\omega_i^{(k)}} \mathcal{L}_{\mathcal{B}_i}$ denotes the gradient of the loss with respect to the model parameters.

D. Model Aggregation

Upon performing local GCN training, workers send (receive) their local GCN parameters to (from) neighbors in the network topology (Line 4). Each worker then aggregates the received parameters to update its local model (Line 5). Let $\mathbf{P}^{(k)} = [\mathbf{P}_{i,j}^{(k)}]_{m \times m}$ denote the mixing weight matrix for model aggregation at round k. The update rule for worker i is as follows:

$$\omega_i^{(k+1)} = \omega_i^{(k)} + \sum_{j \in \mathcal{N}_i^{(k)}} \mathbf{P}_{i,j}^{(k)} \left(\omega_j^{(k)} - \omega_i^{(k)}\right). \tag{23}$$



(a) Graph sampling ($r_i = 0.25$). (b) Graph sampling ($r_i = 1.0$). (c) Graph sampling ($r_i = 0.5$). (d) Graph aggregation ($r_i = 0.5$).

Fig. 6. Illustration of graph sampling and aggregation of a 2-layer GCN on a single worker. The green and blue nodes are in the 0-th layer and 1-th layer, respectively, which are sampled by the red node.

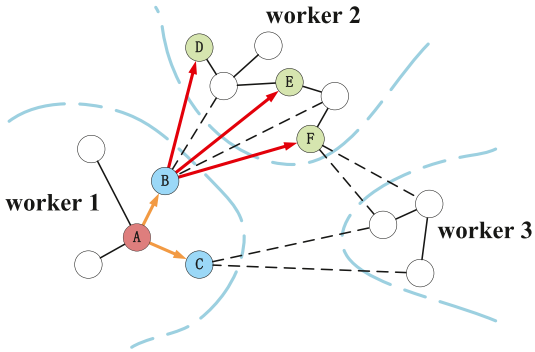


Fig. 7. Illustration of graph sampling in DUPLEX with sampling ratio $r_1 = 0.5$. The black dotted lines indicate external edges among different subgraphs, while the orange and red lines represent internal and external graph neighbor sampling, respectively.

By carefully setting $\mathbf{P}_{i,j}^{(k)}$ according to the network topology $\mathbf{A}^{(k)} = [a_{i,j}^{(k)}]_{m\times m}$, the local models on workers will converge to a stationary point [36]. Specifically, the degree matrix $\mathbf{D}^{(k)} = [\mathbf{D}_{i,j}^{(k)}]_{m\times m}$ is a diagonal matrix, with $\mathbf{D}_{i,i}^{(k)}$ equal to the row sum of $\mathbf{A}^{(k)}$. The Laplacian matrix is defined as $\mathbf{L}^{(k)} = \mathbf{A}^{(k)} - \mathbf{D}^{(k)}$. Boyd et al. [44] proved that the optimal convergence time for a given network topology can be derived by setting $\mathbf{P}_{i,j}^{(k)}$ as follows:

$$\mathbf{P}_{i,j}^{(k)} = \begin{cases} \frac{2}{\lambda_2(\mathbf{L}^{(k)}) + \lambda_m(\mathbf{L}^{(k)})}, & a_{i,j}^{(k)} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

where $\lambda_m(\mathbf{L}^{(k)})$ denotes the m-th smallest eigenvalue of matrix $\mathbf{L}^{(k)}$.

Additionally, based on the received model parameters from neighbors, each worker i computes the consensus distance $C_{i,j}^{(k)}$ between its model and those of its neighbors:

$$C_{i,j}^{(k)} = \|\omega_i^{(k)} - \omega_j^{(k)}\|_2, \forall j \in \mathcal{N}_i^{(k)}. \tag{25}$$

Each worker sends these consensus distances along with the training loss to the coordinator. These training statuses help the coordinator to adjust the network topology and graph sampling ratios for the next round.

E. Privacy Protection

Due to external graph edges connecting nodes across workers, sampled graph nodes may require cross-worker node embedding exchanges during GC operations, raising privacy concerns for raw node features. To prevent the leakage of local raw node features (i.e., $h_v^{(0)}$), DUPLEX restricts cross-worker GC operations to layers 2 through L and enforces intra-worker GC operations on layer 1. Specifically, on the first layer, each worker i aggregates only local raw node features within its subgraph G_i. For node v in G_i, the first-layer embedding h_v^1 is computed as:

$$\begin{aligned} \mathscr{E}_v^1 &= \text{AGG}(\{h_u^0 \mid u \in \mathcal{S}^0(v) \cap G_i\}), \\ h_v^1 &= \mathbf{U}^1(h_v^0 \| \mathscr{E}_v^0), \end{aligned} \tag{26}$$

where $\mathcal{S}^0(v)$ denotes the sampled neighbors of node v. No cross-worker communication occurs here. Starting from layer 2, workers exchange embeddings h_u^l ($l \geq 1$) with neighbors. For exampple, if worker i requires h_u^l (from worker j) to compute h_v^{l+1}, worker j sends h_u^l to worker i. This method can protect the raw feature data of node u. Specifically, it is infeasible for worker i to further infer the local raw node feature h_u^0 from the embedding h_u^l ($l > 0$). This is because the aggregation function AGG($\cdot$) (*e.g.*, element-wise mean or sum of multiple vectors) in each layer, which is an irreversible process, destroys information about individual h_u^0. Besides, each layer applies nonlinear transformations (*e.g.*, ReLU) in the GC operation, further obfuscating h_u^0. This analysis is consistent with privacy arguments in previous federated graph learning literature [23]. Therefore, DUPLEX ensures privacy-preserving collaboration by sharing only post-aggregation embeddings while maintaining model efficacy.

IV. PERFORMANCE EVALUATION

A. Experimental Setup

Experimental Platform: Our experiments are conducted on a physical platform including 50 NVIDIA Jetson devices [45], which comprise 20 Jetson TX2s, 25 Jetson NXs, and 5 Jetson

TABLE II
DEVELOPMENT BOARDS USED IN EXPERIMENTS

Device	GPU	CPU	Performance
TX2	256-core Pascal (8GB)	2-core Denver 2 (64 b)	2 TFLOPS
NX	384-core Volta (16GB)	6-core Carmel (64 b)	21 TOPS
AGX	512-core Volta (32GB)	8-core Carmel (64 b)	32 TOPS

TABLE III
SUMMARY OF THE GRAPH DATA STATISTICS

Datasets	#Nodes	#Edges	Features	#Classes
ogbn-arxiv	169,343	1,166,243	128	40
ogbn-products	2,449,029	61,859,140	100	47
Reddit	232,965	114,615,892	602	41

AGXs. The detailed technical specifications of these devices are listed in Table II. Specifically, the TX2 and NX devices can work in one of four computation modes, while the AGX devices have eight modes. Devices (i.e., workers) working in different modes exhibit diverse computing capabilities. On this platform, we build the software infrastructure based on Docker Swarm [46] and the PyTorch deep learning library [47]. Docker Swarm effectively builds a decentralized system and helps detect the status of each device. Besides, communication between devices is established using the socket library [48], which enables efficient data transfer through a set of sending and receiving functions. To accurately capture the heterogeneous bandwidths among workers, we follow prior works [49], [50] by connecting all devices via wireless links and arranging them at different locations within a building of 400 m^2 . Due to different distances between the devices and routers, as well as the random channel noise, the measured bandwidth of the devices fluctuates between 5 to 20 Mbps.

Datasets and Models: We evaluate the performance of DUPLEX and baselines on three widely-used datasets: ogbn-arxiv, ogbn-products [19], and Reddit [6], which are summarized in Table III.

- The ogbn-arxiv dataset captures the citation network between Computer Science papers on arXiv. Our aim is to categorize each paper node's subject area of its corresponding arXiv paper.
- The ogbn-products dataset, which aims to predict the category of a product, represents a network of products sold on Amazon, with nodes representing individual products and edges indicating frequent co-purchases.
- The Reddit dataset, which aims to predict the community to which a post belongs, consists of a large collection of user-generated content from the social website.

We perform node classification tasks using two well-known deep graph learning models: a two-layer GCN model [5] on ogbn-arxiv and Reddit, and a two-layer GraphSAGE model [6] on ogbn-products. Specifically, GCN is a scalable model for learning on graph data. Besides, GraphSAGE is a general inductive model that builds on GCN to generate node embeddings efficiently [6].

Non-IID Data Partitioning: To simulate the non-IID graph data in the real world, we follow the prior work [34] to partition the original training set into 50 parts (i.e., local training sets), each of which is assigned to a specific worker. The graph edges (including internal and external edges) are maintained according to the global graph in the original training set. Specifically, we assume that for every worker, the classes of graph nodes in the local training set follow the Dirichlet distribution. Accordingly, we sample the local training set $\mathcal{D}_i \sim Dir_i(\alpha)$ from the original training set and allocate $\mathcal{D}_i$ to each worker i. α determines the degree of non-IID, and a lower value of α generates a high node class distribution shift. Unless otherwise specified, we maintain a default setting of $\alpha = 10.0$ throughout our experiments.

Baselines: We present a comparative evaluation of DUPLEX against the following methods:

- **S-Glint** [17] is a DFGL framework that effectively constructs a fixed and sparse network topology by evaluating the convergence contributions of workers.
- **TDGE** [51] presents a hypercube graph construction method to reduce data heterogeneity by carefully selecting neighbors of each device. In addition, TDGE explores the communication patterns in hypercube topology and proposes a sequential synchronization scheme to reduce communication cost.
- **D-FedPNS** is an alternative of FedPNS [24], which is an FGL framework conducting the periodic graph neighbor node sampling and embedding synchronization of cross-worker neighbors. We extend FedPNS to D-FedPNS by forcing P2P communication between workers without changing the periodic sampling strategy in FedPNS.
- **D-FedGraph** is an alternative to FedGraph [23], which is a FGL framework intelligently sampling graph nodes for training by automatically adjusting sampling policies based on deep reinforcement learning. We extend FedGraph to D-FedGraph by forcing P2P communication between workers without changing the sampling policies.
- **D-FedGNN** [52] proposes a dual-topology adaptive communication mechanism that leverages the unique topological features of each worker's local subgraph to dynamically construct and optimize the inter-worker communication topology. This allows D-FedGNN to guide model aggregation efficiently in the face of data heterogeneity.

To provide a comprehensive comparison between the above methods and DUPLEX, we train D-FedGraph and D-FedPNS on three network topologies: ring topology, sparse topology (each worker has 10 neighbors), and dense topology (each worker has 25 neighbors). We formulate various baselines by training D-FedGraph and D-FedPNS on different network topologies, such as D-FedGraph(dense) and D-FedPNS(sparse), representing D-FedGraph is trained based on the dense and D-FedPNS is trained based on the sparse network topology, respectively. For S-Glint, TDGE, and D-FedGNN, we assign different graph sampling ratios for workers to represent several baselines. For example, Glint(0.1) and TDGE(0.1) represent that each worker in S-Glint and TDEG has a sampling ratio of 0.1.

Performance Metrics: We adopt three metrics to evaluate the performance of different algorithms in our experiments. ① *Test accuracy:* During each round, the coordinator averages the test accuracy of all local models to compute the overall test accuracy.

TABLE IV
FINAL TEST ACCURACY (%) OF DUPLEX AND THE BASELINES ON THE THREE DATASETS

Datasets	DUPLEX	D-FedGraph			D-FedPNS			Glint				TDGE				D-FedGNN			
		ring	sparse	dense	ring	sparse	dense	0.1	0.3	0.5	0.7	0.1	0.3	0.5	0.7	0.1	0.3	0.5	0.7
ogbn-arxiv	**53.21**	42.26	49.78	**52.97**	43.52	50.29	**53.47**	45.33	50.98	52.91	**53.59**	43.73	50.21	**53.11**	53.74	45.69	51.61	**53.72**	54.26
Reddit	**88.23**	79.14	83.51	**88.19**	80.85	83.86	**88.36**	83.31	86.65	**88.32**	89.27	82.65	84.94	87.82	**88.33**	83.23	85.91	87.51	**88.54**
ogbn-products	**65.57**	57.53	62.19	**65.78**	58.33	62.74	**65.42**	58.37	62.55	64.23	**65.59**	57.42	62.13	63.87	**65.35**	59.65	63.11	**65.47**	66.15

TABLE V
TEST ACCURACY (%) OF DUPLEX AND THE BASELINES ON OGBN-ARXIV, REDDIT, AND OGBN-PRODUCTS UNDER THE COMMUNICATION RESOURCE CONSTRAINT OF 1.5 GB, 10 GB, AND 20 GB, RESPECTIVELY

Datasets	ogbn-arxiv	Reddit	ogbn-products
D-FedGraph	49.32	83.98	61.37
D-FedPNS	49.89	84.32	62.19
Glint	51.53	85.72	62.91
TDGE	50.36	85.52	61.73
D-FedGNN	51.79	84.92	63.65
DUPLEX	**53.23**	**88.25**	**65.58**

② *Training time:* For fairness, we compare the end-to-end training time of different algorithms to reach the same target test accuracy. ③ *Communication cost:* We compare the total communication cost for exchanging graph node embeddings and local models of all workers while achieving the same target test accuracy.

Parameter Settings: We use a hidden feature size of 128 for GCN and 256 for GraphSage. Besides, we adopt the Adam optimizer [53] to optimize the models, with an initial learning rate of 0.01 and the weight decay of $3e^{-4}$. The number of local updates and the training batch size are fixed to 5 and 64 for Reddit, and 10 and 128 for ogbn-arxiv and ogbn-products, respectively. We train the models for a specific number of rounds to ensure convergence: 200 rounds for GCN on ogbn-arxiv, 100 rounds for GCN on Reddit, and 150 rounds for GraphSage on ogbn-products. For the reward function of (12), we set χ, ϱ and φ as 2, 1 and 10, respectively.

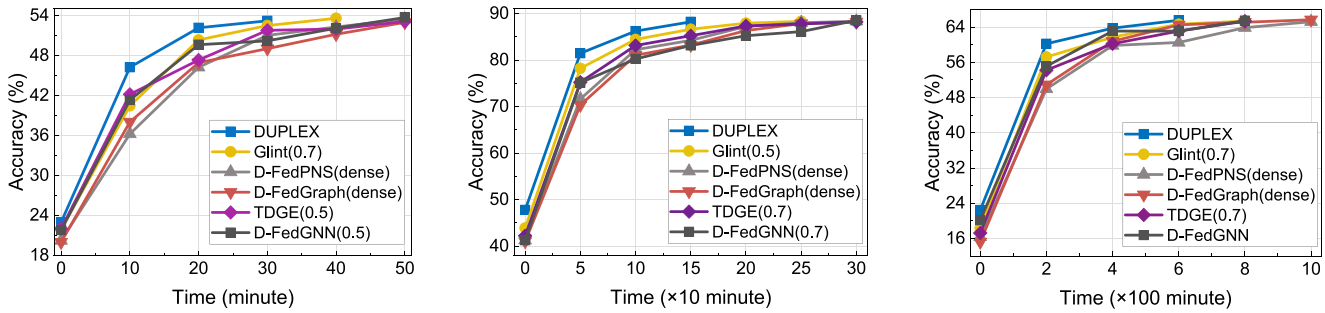
B. Overall Performance

Test Accuracy: The test accuracy of DUPLEX and the baselines on the three datasets is listed in Table IV. The baselines are labeled with different graph neighbor sampling ratios and network topologies in the shadow rows. On ogbn-arxiv, since D-FedGraph(dense), D-FedPNS(dense), Glint(0.7), TDGE(0.5), and D-FedGNN(0.5) achieve comparable test accuracy to DUPLEX, they are selected for subsequent experiments to compare training time and communication cost with DUPLEX. For Reddit, we consider D-FedGraph(sparse), D-FedPNS(dense), Glint(0.5), TDGE(0.7), and D-FedGNN(0.7) for further comparison with DUPLEX. Lastly, D-FedGraph(dense), D-FedPNS(dense), Glint(0.7), TDGE(0.7), and D-FedGNN(0.5) are chosen on ogbn-products for the same reason. These selections are made based on the similar test accuracy achieved by the baselines compared to DUPLEX, ensuring a fair evaluation of our proposed algorithm on resource consumption, such as training time and communication cost, to reach the target test accuracy.

The time-to-accuracy performance of DUPLEX and the selected baselines on the three datasets is presented in Fig. 8. We observe that the test accuracy of DUPLEX first steadily increases and then stabilizes on the three datasets. It indicates that the model training of DUPLEX will converge to a stationary point. Impressively, DUPLEX always achieves the highest accuracy compared to these baselines within the same training time. For instance, in Fig. 8(c), DUPLEX outperforms Glint(0.7), D-FedGraph(dense), D-FedPNS(dense), TDGE(0.7), and D-FedGNN(0.5) in accuracy by 3.1%, 10.2%, 9.7%, 6.3%, and 5.1% respectively, after 200 minutes of training on ogbn-products. Moreover, DUPLEX always converges faster than the baselines to achieve the same accuracy. This is because the suboptimal network topology or graph sampling strategy of the baselines cannot fully handle the heterogeneous data distributions in DFGL. In contrast, DUPLEX achieves a faster convergence rate and higher test accuracy with adaptive topology construction and proper sampling ratio assignment.

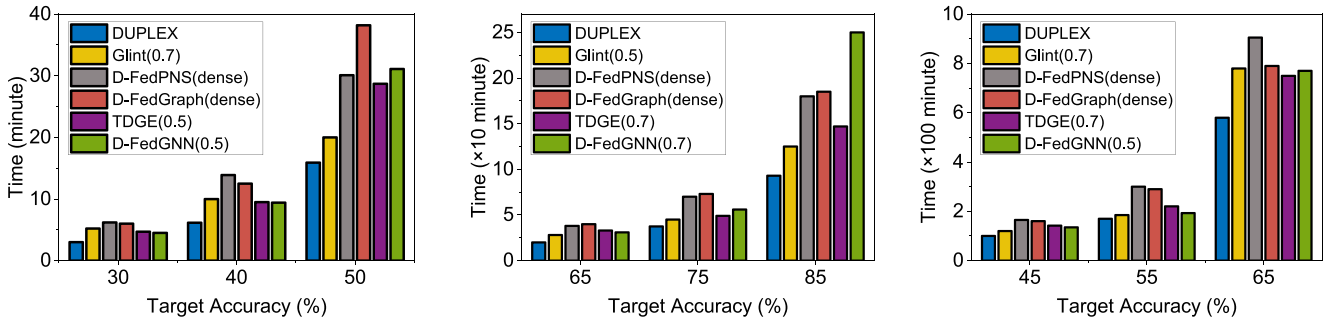
To further evaluate the model performance of DUPLEX and the baselines, we train models under the communication resource constraint of 1.5 GB, 10 GB, and 20 GB on ogbn-arxiv, Reddit, and ogbn-products, respectively. As shown in Table V, DUPLEX can achieve the highest accuracy. For example, DUPLEX achieves the test accuracy improvement by about 4.2%, 3.6%, 2.7%, 3.9%, and 1.9% on ogbn-products, compared with D-FedGraph, D-FedPNS, Glint, TDGE, and D-FedGNN, respectively.

Time Cost: In order to gain a deeper understanding of resource consumption associated with the baselines and DUPLEX, we conduct a set of experiments to record the completion time when they attain the same target accuracy on the three datasets. The simulation results are depicted in Fig. 9, revealing that DUPLEX outperforms the baselines in terms of reducing the completion time required to attain the target accuracy. For instance, as illustrated in Fig. 9(b), DUPLEX takes 93 minutes to achieve the target accuracy of 85% for GCN on Reddit, while D-FedGraph(dense), D-FedPNS(dense), Glint(0.5), TDGE(0.7), and D-FedGNN(0.7) take 185, 180, 121, 147, and 250 minutes, respectively. Besides, for GraphSage on ogbn-products, as shown in Fig. 9(c), DUPLEX speeds up training by about 1.4×, 1.3×, 1.6×, 1.3×, and 1.4× compared with Glint, D-FedGraph(dense), D-FedPNS(dense), TDGE(0.7), and D-FedGNN(0.5), respectively. These improvements highlight the efficiency gains enabled by DUPLEX in terms of



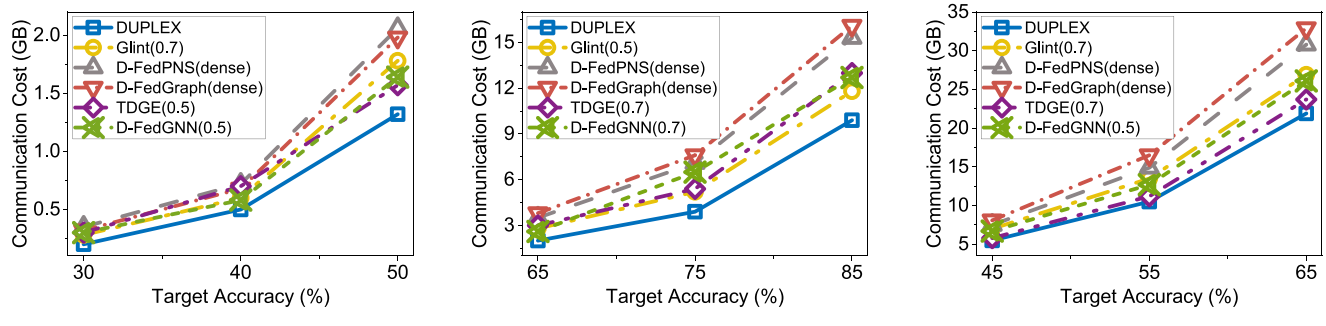
(a) ogbn-arxiv (b) Reddit (c) ogbn-products

Fig. 8. Time-to-Accuracy of DUPLEX and the baselines on the three datasets. For a fair comparison, the baselines are selected based on the results in Table IV, where their final test accuracy (bold) is close to that of DUPLEX.



(a) ogbn-arxiv (b) Reddit (c) ogbn-products

Fig. 9. Training time of DUPLEX and the selected baselines to reach the target accuracy on the three datasets.



(a) ogbn-arxiv (b) Reddit (c) ogbn-products

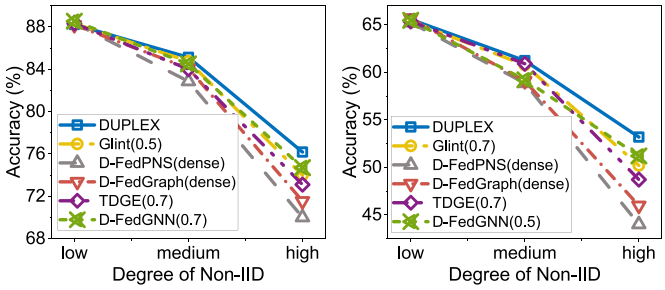
Fig. 10. Communication cost of DUPLEX and the baselines to reach the target accuracy on the three datasets.

accelerating the training process of DFGL by optimizing the network topology and the graph sampling ratio.

Communication Cost: We measure the required communication cost of DUPLEX and the baselines to attain the same target accuracy on the three datasets. The results in Fig. 10 show that DUPLEX outperforms the baselines in terms of communication cost. Specifically, by Fig. 10(a), DUPLEX reduces the communication cost by 26%, 33%, 36%, 16%, and 20%, respectively, compared with Glint(0.7), D-FedGraph(dense), D-FedPNS(dense), TDGE(0.5), and D-FedGNN(0.5) on ogbn-arxiv to reach the accuracy of 50%. Besides, by Fig. 10(b), DUPLEX reduces the communication cost on Reddit by 16%, 39%, 35%, 24%, and 22%, respectively, compared with Glint(0.5), D-FedGraph(dense), D-FedPNS(dense), TDGE(0.7), and D-FedGNN(0.7) to reach the accuracy of 85%. These results indicate that DUPLEX is highly scalable to effectively reduce communication cost across a variety of different datasets.

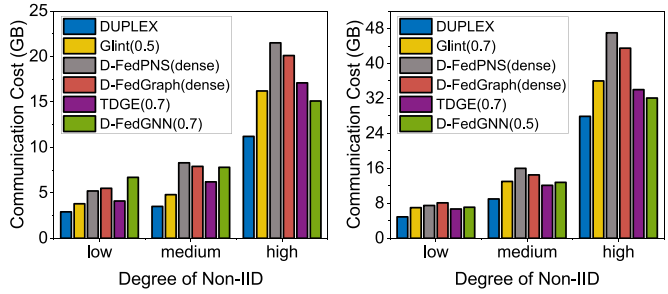
C. Impact of non-IID Data

In this section, we evaluate the performance of DUPLEX and the baselines under different degrees of data heterogeneity. Specifically, we train models on three Dirichlet distributions with $\alpha = \{10.0, 1.0, 0.1\}$, where a smaller α denotes a higher



(a) Reddit. (b) ogbn-products.

Fig. 11. Impact of non-IID data on model performance.



(a) Reddit. (b) ogbn-products.

Fig. 12. Impact of non-IID data on communication cost.

non-IID degree. As shown in Fig. 11, DUPLEX achieves more robust performance on non-IID data compared to the baselines. Although all algorithms suffer from model accuracy degradation with a higher degree of data heterogeneity, DUPLEX performs significantly more stably than the baselines. For example, as the non-IID degree increases (with α decreasing from 10.0 to 0.1) on ogbn-products, the model accuracy of Glint(0.7), D-FedPNS(dense), D-FedGraph(dense), TDGE(0.7), and D-FedGNN(0.5) decreases dramatically by 15.2%, 21.2%, 19.7%, 16.6%, and 14.3%, respectively, while that of DUPLEX only degrades by 11.4%.

Besides, we observe that DUPLEX also improves resource efficiency on non-IID data. Fig. 12 shows the communication costs for model training to reach the target accuracy of 70% and 45% on Reddit and ogbn-products, respectively. With an increase in non-IID degrees, all algorithms incur higher communication costs. However, the cost of DUPLEX grows much more slowly than that of the baselines. Specifically, under different non-IID degrees, DUPLEX saves about 21.2%-40.6% in communication costs compared to the baselines. These results suggest that the joint optimization of network topology and graph sampling, guided by consensus distance and training loss, enables DUPLEX to effectively alleviate the negative impact of non-IID data.

D. Ablation Study

In this section, we implement several breakdown versions of DUPLEX to evaluate the effectiveness of topology construction and graph neighbor sampling in DUPLEX, respectively.

TABLE VI
TEST ACCURACY (%) OF NATIVE DUPLEX AND THE BREAKDOWN VERSIONS ON THE THREE DATASETS

Datasets	DUPLEX	breakdown versions of DUPLEX					
		ring	sparse	dense	0.3	0.5	0.7
ogbn-arxiv	**53.21**	43.45	48.97	**52.81**	48.23	51.87	**53.25**
Reddit	**88.23**	77.84	82.19	**87.96**	84.63	87.07	**88.86**
ogbn-products	**65.57**	55.26	61.96	**65.49**	63.23	**65.23**	66.23

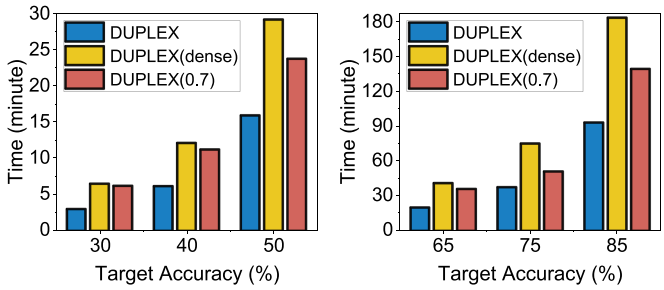
TABLE VII
TEST ACCURACY (%) OF DIFFERENT VERSIONS OF DUPLEX ON OGBN-ARXIV AND REDDIT UNDER THE COMMUNICATION RESOURCE CONSTRAINT OF 1.5 GB AND 10 GB, RESPECTIVELY

Datasets	DUPLEX	DUPLEX(dense)	DUPLEX(0.7)
ogbn-arxiv	**53.25**	48.01	49.78
Reddit	**88.25**	83.58	86.12

- *DUPLEX with fixed network topology:* We disable the adaptive network topology construction in DUPLEX, in which workers communicate with others based on the fixed network topology without changing the graph sampling ratios in native DUPLEX. We formulate various breakdown versions of DUPLEX, i.e., DUPLEX(ring), DUPLEX(sparse), and DUPLEX(dense), by training DUPLEX on different network topologies.
- *DUPLEX with fixed graph sampling ratio:* We disable the adaptive graph neighbor sampling in DUPLEX, in which workers conduct graph neighbor sampling according to the fixed sampling ratio and communicate with others based on the network topology output by deep reinforcement learning. We formulate various breakdown versions of DUPLEX, i.e., DUPLEX(0.3), DUPLEX(0.5) and DUPLEX(0.7), by training DUPLEX according to different sampling ratios.

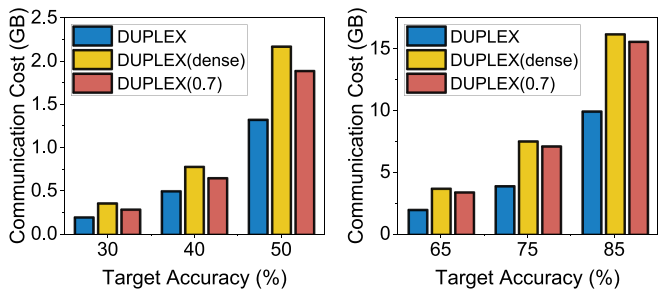
Test Accuracy: Table VI shows the test accuracy of native DUPLEX (i.e., DUPLEX with optimized network topology construction and graph neighbor sampling) and the breakdown versions on the three datasets. Without adaptive network topology construction or graph sampling, the test accuracy decreases significantly on all datasets. For example, on Reddit, DUPLEX with the sparse topology and DUPLEX with a fixed graph sampling ratio of 0.3 achieve 6.04% and 3.60% lower final accuracy than that of native DUPLEX. Besides, as shown in Table VII, under the same communication cost constraint (1.5 GB on ogbn-arxiv and 10 GB on ogbn-products), native DUPLEX is able to visibly improve test accuracy by 2.13%-5.24% on ogbn-arxiv and Reddit, compared with the breakdown versions of DUPLEX, i.e., DUPLEX(dense) and DUPLEX(0.7). These results indicate that the joint optimization of network topology and graph sampling strategy contributes to the accuracy improvements of DUPLEX.

Training Time and Communication Cost: As illustrated in Fig. 13, the training time for breakdown versions of DUPLEX to reach the target accuracy on ogbn-arxiv (50%) and Reddit (85%) increases 49.1%-96.8%, compared to native DUPLEX. Besides,



(a) ogbn-arxiv. (b) Reddit.

Fig. 13. Training time of different versions of DUPLEX to reach the target accuracy. The breakdown versions of DUPLEX are selected from Table VI, where their final test accuracy (bold) is close to that of native DUPLEX.



(a) ogbn-arxiv. (b) Reddit.

Fig. 14. Communication cost of different versions of DUPLEX to reach the target accuracy.

Fig. 14 evaluates the effect of the network topology construction and graph sampling on the communication cost. For instance, compared to native DUPLEX, DUPLEX(dense) and DUPLEX(0.7) need 63.6% and 42.4% more communication cost to reach the target accuracy of 50% on ogbn-arxiv. By jointly constructing the dynamic network topology and assigning proper graph sampling ratios to workers, DUPLEX not only effectively accelerates training but also reduces communication costs.

Analysis: The performance degradation caused by fixed network topologies or sampling ratios stems from three key limitations. First, fixed topologies or sampling ratios cannot adapt to fluctuating network bandwidth across workers. For example, slow or congested workers become bottlenecks, delaying synchronization and model training. Besides, dynamic topologies and sampling ratios enable broader exchanges of model parameters and node embeddings by periodically reconnecting workers, fostering model diversity and avoiding local optima. Fixed topologies and sampling ratios restrict information flow, leading to suboptimal convergence and model accuracy on non-IID data. In addition, fixed ratios and topologies under-share data points in early training phases and over-share instances in later phases.

E. Sensitivity Analysis

To evaluate the impact of the parameters χ, ϱ, and φ in the reward function, we vary each parameter independently while fixing the others to default values. χ penalizes the reward as the round time becomes longer. A larger χ provides a more severe penalty for an increase in round time. ϱ prioritizes the descent of consensus distance. A larger ϱ promotes the reduction of discrepancies between workers' local models, improving the training performance on non-IID data. φ prioritizes the minimization of the training loss. A larger φ promotes the convergence of local models. Fig. 15 shows that DUPLEX outperforms baselines consistently in terms of completion time and network traffic to reach the target accuracy across different χ (*e.g.*, 1, 2, 3), ϱ (*e.g.*, 0.5, 1, 1.5), and φ (*e.g.*, 5, 10, 15). However, some extreme values of these parameters, *e.g.*, 10 for χ and ϱ, 50 for φ, lead to significant performance degradation. Therefore, we recommend a conservative parameter setting (*e.g.*, $\chi = 2, \varrho = 1, \varphi = 10$) for most scenarios.

F. Scalability Evaluation

To evaluate the scalability of DUPLEX in complex graph processing, we train the GraphSAGE model on a complex graph dataset, i.e., ogbn-mag[1] [19], which is a large-scale and heterogeneous academic graph derived from a subset of the Microsoft Academic Graph (MAG). We test the completion time and network traffic required for different methods to reach the same target accuracy (i.e., 42%) at different worker scales (*e.g.*, 50, 100, 200, 300, 400, 500).

As shown in Fig. 16, the results show that the completion time and network traffic required for DUPLEX increase much slowly with the increasing scale of workers, compared to baselines. For instance, when the worker scale increases from 50 to 500, the completion time/network traffic growth rate of DUPLEX is 22.1×/28.3×, while the growth rates of Glint, D-FedGraph, TDGE, and D-FedPNS are 25.9×/31.1×, 29.8×/40.2×, 26.1×/40.5×, and 30.2×/34.4×, respectively. These results highlight the scalability of DUPLEX in complex graph processing with large-scale workers.

V. RELATED WORKS

Graph Neural Networks (GNNs) [54], [55] have revolutionized the processing of graph-structured data, where data samples are interconnected through graph topologies. Unlike traditional neural networks such as Convolutional Neural Networks [56], which are tailored for grid-like data structures, GNNs excel at extracting neighborhood information from the underlying graph topology. This capability enables them to efficiently capture and learn the intricate, structured knowledge inherent in graph data. However, training high-quality GNNs necessitates access to vast amounts of graph data, which in real-world scenarios is often distributed across numerous edge devices [57]. Privacy concerns

[1]Ogbn-mag is *a heterogeneous academic network* comprising four types of entities, i.e., papers (736,389 nodes), authors (1.13 million nodes), institutions (8,740 nodes), and fields of study (59,965 nodes), interconnected by four types of directed relationships, i.e., authors affiliating with institutions, authors writing papers, papers citing other papers, and papers belonging to research fields. The primary task is a 349-class classification, where the goal is to predict the venue of each paper based on its content, citations, authors, and institutional affiliations. With over 21 million edges and 1.93 million nodes, ogbn-mag poses challenges in handling heterogeneous information fusion, scalability, and long-range dependencies in academic networks.

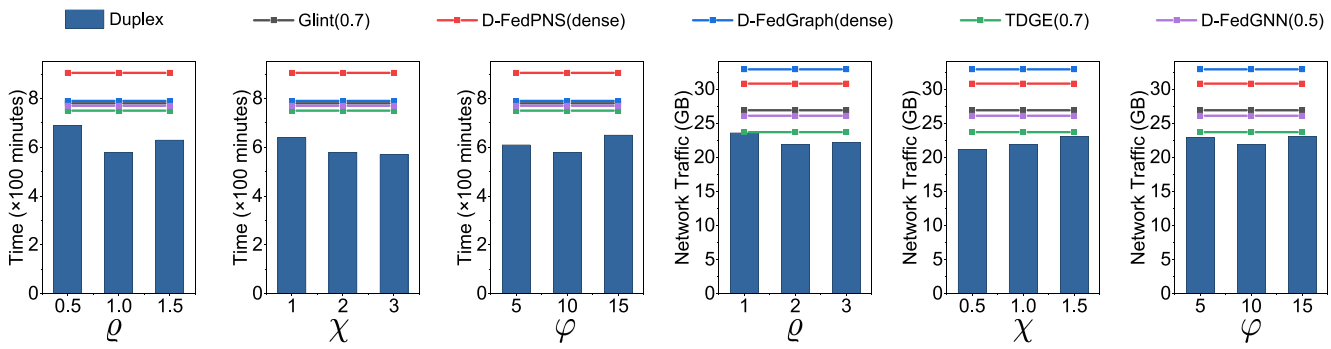
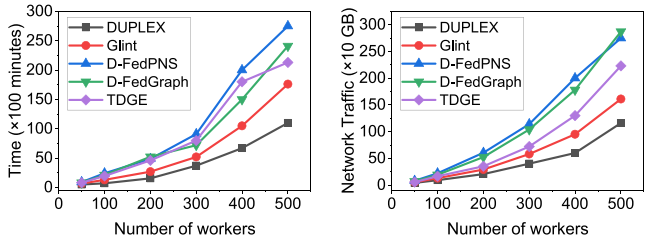


Fig. 15. Parameter sensitivity analysis of DUPLEX on ogbn-products with $\alpha = 10.0$.



(a) Completion time vs. Number of workers. (b) Network traffic vs. Number of workers.

Fig. 16. Completion time and network traffic on different worker scales.

pose significant barriers to aggregating this data centrally, as data holders are reluctant to share their private information.

Federated Graph Learning: FGL [12], [57], [58] has emerged as a promising solution to this challenge through a distributed and collaborative training paradigm where each data holder (i.e., the worker) participates in the training process using their local graph data. This approach not only preserves data privacy but also enhances security by sharing only model parameters rather than raw data. Current research in FGL can be broadly classified into three scenarios: 1) each data holder owns only a single node within the global graph [59]; 2) each data holder has a subgraph of a global graph [60], [61], [62]; and 3) each data holder possesses multiple independent graphs [63]. The second scenario is the most prevalent in real-world applications and has become the mainstream focus in FGL [57]. The optimization efforts in this article are also based on this scenario. In this context, while each data holder manages their own subgraph, edges exist that connect different subgraphs, representing structural relationships between data holders. The primary objective is for each data holder to leverage their subgraph data to collaboratively train the global GNN model, enhancing overall performance through shared learning while maintaining data privacy.

Decentralized Federated Graph Learning: One of the primary concerns in traditional worker-server-based FGL is the single point of failure and limited scalability [64]. To address this, Pei et al. [15] proposed a new decentralized federated graph learning framework (D-FedGNN), which allows multiple data holders to train GNN models in a peer-to-peer network without a centralized server. Another method, SpreadGNN [65], extends FGL to realistic serverless and multi-task settings, and utilizes a novel optimization algorithm to ensure the convergence of GNN models. Moreover, DCI-PFGL [66] introduces a cross-institutional framework for IoT service recommendation, employing anonymized graph feature embeddings and smart contracts on a blockchain to maintain privacy while addressing data heterogeneity. Efficient communication is crucial for scaling decentralized FGL, with recent studies emphasizing communication cost reduction. For example, Glint [16] and S-Glint [17] are novel DFGL frameworks that leverage traffic throttling and flow scheduling to alleviate communication bottlenecks in DFGL tasks, leading to enhanced service quality for graph learning.

Graph Neighbor Sampling: The recent advancements in graph neighbor sampling within FGL address several pressing challenges, including computational efficiency, communication overhead, and representation accuracy. For example, FedGraph [23] uses deep reinforcement learning to dynamically assign a distinct neighbor sampling ratio for each worker in large-scale federated graphs, addressing issues such as privacy leakage and training overhead. This method dynamically balances training accuracy and speed, making it viable for privacy-sensitive and resource-intensive applications. Du et al. [24] optimized neighbor sampling intervals to balance convergence accuracy with runtime efficiency. By periodically sampling neighbors, this algorithm mitigates resource constraints without compromising model accuracy. FedAAS [25], developed by Li et al., utilizes historical embeddings to enhance sampling efficiency by focusing on nodes with significant impact, reducing communication costs while maintaining model accuracy in FGL. This method addresses scalability concerns inherent in large graph networks. Additionally, FedAIS [67] focuses sampling efforts on high-importance nodes in the graph to adapt to structural heterogeneity. This importance-based sampling method improves both the efficiency and scalability of FGL by concentrating on influential nodes. However, these works mainly focus on PS-based FGL, which suffers from the single point of failure problem as the PS may be down.

Topology Construction: A key objective in decentralized federated learning (DFL) is achieving efficient communication and high model performance without a parameter server, where the topology of the peer-to-peer network plays a pivotal role. For example, cluster topologies are advantageous in leveraging

data distribution similarities, which help reduce communication costs while maintaining performance comparable to centralized models. Wireless ring networks present another approach, using carefully designed consensus coefficients and network coding to balance learning efficacy with communication latency. DFGL, a special variant of DFL, shares the same goals as DFL but faces unique challenges. Specifically, DFGL requires extensive communication of graph node embeddings among workers during training, leading to high communication overhead. To address this, Liu et al. [17] proposed constructing a fixed network topology for worker communication, selectively transmitting only critical information and thus significantly reducing network traffic. Krasanakis et al. [68] also advanced DFGL by developing a peer-to-peer GNN model for decentralized node classification. Their methods integrate asynchronous diffusion and graph-based communication mechanisms, reducing dependency on centralized data exchanges while sustaining model accuracy.

Non-IID Handling Strategies: The non-IID problem in FGL manifests in two principal dimensions, i.e., data distribution heterogeneity and graph topology heterogeneity. The former encompasses heterogeneity in node/edge attributes (*e.g.*, feature skews in social network graphs) and label distributions (*e.g.*, class imbalance in molecular property prediction tasks). The latter arises from structural divergences, including variations in node degrees, edge weights, or global connectivity patterns. Most FGL research focuses on data distribution heterogeneity [57], [69], providing some personalized FL-inspired solutions such as regularized local loss, knowledge distillation, clustering, and personalized aggregation. For instance, FedAlign [70] introduces an optimal transport distance-based regularization term between local and global models to minimize model divergence. InfoFedSage [71] employs a PS-side generative module coupled with worker-side information bottleneck regularization to alleviate the non-IID problem. GCFL [10] partitions workers into clusters using Graph Isomorphism Network (GIN) [72], enabling cluster-specific model aggregation. However, existing works' solutions primarily adopt the PS architecture and fail to address the high communication overhead in DFGL. Our framework bridges these gaps through consensus distance-guided joint optimization of network topology and graph sampling, improving both training performance and communication efficiency in non-IID DFGL environments.

VI. Potential Limitations and Future Work

First, DUPLEX relies on a logical coordinator to collect global states and compute configurations. While the coordinator is fully different from the PS that needs to aggregate local model parameters and update the global model, its transient failure could temporarily disrupt adaptation of network topology and graph sampling. To eliminate single-point dependencies, we are integrating a leader election protocol (*e.g.*, Raft consensus [73]) among workers to dynamically re-elect a coordinator during failures. Further, we are exploring decentralized consensus algorithms (*e.g.*, blockchain-inspired Proof-of-Stake) to distribute coordinator responsibilities across workers. This aligns with our goal of minimizing centralization while retaining the efficiency benefits of coordinator-guided topology and sampling optimization. In addition, the synchronous training paradigm in DUPLEX simplifies deployment but may delay aggregation in heterogeneous edge environments due to stragglers. To address this, we plan to extend DUPLEX with asynchronous aggregation protocols and staleness-aware model exchange strategies to accommodate heterogeneous worker capabilities. Specifically, stragglers with stale model updates (beyond a staleness threshold) will asynchronously trigger local aggregation with neighbors, decoupling them from the global synchronization barrier. Besides, the coordinator will prioritize workers with lower staleness for configuration updates, ensuring timely adaptation of network topology and sampling ratios while accommodating slower workers.

VII. Conclusion

In this work, we propose an efficient and effective DFGL framework, named DUPLEX, which integrates network topology construction and graph neighbor sampling to address the challenge of non-IID graph data and reduce communication costs. We also propose an efficient learning-driven algorithm to adaptively determine the optimal topology and sampling ratios for workers, considering resource budgets and data distributions, simultaneously. We build a simulated DFGL environment to evaluate the performance of DUPLEX via extensive experiments. The results significantly demonstrate the effectiveness and efficiency of DUPLEX.

References

[1] J.-C. Zhang, A. M. Zain, K.-Q. Zhou, X. Chen, and R.-M. Zhang, "A review of recommender systems based on knowledge graph embedding," *Expert Syst. Appl.*, vol. 250, 2024, Art. no. 123876.
[2] K. Yokotani, M. Takano, N. Abe, and T. A. Kato, "Predicting social anxiety disorder based on communication logs and social network data from a massively multiplayer online game: Using a graph neural network," *Psychiatry Clin. Neurosciences*, vol. 79, pp. 274–281, 2025.
[3] H. Chen et al., "Macro graph neural networks for online billion-scale recommender systems," in *Proc. ACM Web Conf.*, 2024, pp. 3598–3608.
[4] Z. Wu, "Optimizing e-commerce recommender systems: A comprehensive review of techniques and future directions," *Appl. Comput. Eng.*, vol. 97, pp. 96–101, 2024.
[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[6] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[7] W. Ren, N. Jin, and L. OuYang, "Phase space graph convolutional network for chaotic time series learning," *IEEE Trans. Ind. Informat.*, vol. 20, no. 5, pp. 7576–7584, May 2024.
[8] J. Huang et al., "KeystrokeSniffer: An off-the-shelf smartphone can eavesdrop on your privacy from anywhere," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 6840–6855, 2024.
[9] J. Liu et al., "Adaptive local update and neural composition for accelerating federated learning in heterogeneous edge networks," *IEEE Trans. Netw.*, vol. 33, no. 4, pp. 1438–1453, Aug. 2025.
[10] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-iid graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 18839–18852.
[11] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 6671–6682.
[12] P. Sun et al., "Pain-FL: Personalized privacy-preserving incentive for federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3805–3820, Dec. 2021.

[13] J. Liu, J. Yan, H. Xu, Z. Wang, J. Huang, and Y. Xu, "Finch: Enhancing federated learning with hierarchical neural architecture search," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 6012–6026, May 2024.

[14] J. Liu et al., "Enhancing semi-supervised federated learning with progressive training in heterogeneous edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 3, pp. 2315–2330, Mar. 2025.

[15] Y. Pei et al., "Decentralized federated graph neural networks," in *Proc. Int. Workshop Federated Transfer Learn. Data Sparsity Confidentiality Conjunction IJCAI*, 2021.

[16] T. Liu, P. Li, and Y. Gu, "Glint: Decentralized federated graph learning with traffic throttling and flow scheduling," in *Proc. 2021 IEEE/ACM 29th Int. Symp. Qual. Service*, 2021, pp. 1–10.

[17] T. Liu, P. Li, Y. Gu, and Z. Su, "S-Glint: Secure federated graph learning with traffic throttling and flow scheduling," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 894–903, Jun. 2023.

[18] J. Liu et al., "Accelerating decentralized federated learning with probabilistic communication in heterogeneous edge computing," *IEEE Trans. Netw.*, early access, Aug. 25, 2025, doi: 10.1109/TON.2025.3600015.

[19] W. Hu et al., "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 22118–22133.

[20] M. Satyanarayanan, T. Eiszler, J. Harkes, H. Turki, and Z. Feng, "Edge computing for legacy applications," *IEEE Pervasive Comput.*, vol. 19, no. 4, pp. 19–28, Oct.–Dec. 2020.

[21] H. Zhou, M. Li, P. Sun, B. Guo, and Z. Yu, "Accelerating federated learning via parameter selection and pre-synchronization in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 11, pp. 10313–10328, Nov. 2024.

[22] P. Goyal et al., "Accurate, large minibatch SGD: Training imagenet in 1 hour," 2017, *arXiv:1706.02677*.

[23] F. Chen, P. Li, T. Miyazaki, and C. Wu, "FedGraph: Federated graph learning with intelligent sampling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1775–1786, Aug. 2022.

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

[25] A. Li et al., "Historical embedding-guided efficient large-scale federated graph learning," *Proc. ACM Manage. Data*, vol. 2, no. 3, pp. 1–24, 2024.

[26] L. Wang, Y. Xu, H. Xu, M. Chen, and L. Huang, "Accelerating decentralized federated learning in heterogeneous edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5001–5016, Sep. 2023.

[27] J. Liu, Y. Xu, H. Xu, Y. Liao, Z. Wang, and H. Huang, "Enhancing federated learning with intelligent model migration in heterogeneous edge computing," in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 1586–1597.

[28] M. Waqas, Y. Niu, M. Ahmed, Y. Li, D. Jin, and Z. Han, "Mobility-aware fog computing in dynamic environments: Understandings and implementation," *IEEE Access*, vol. 7, pp. 38867–38879, 2019.

[29] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. Stich, "Consensus control for decentralized deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5686–5696.

[30] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *Proc. 8th Int. Conf. Learn. Representations*, 2019.

[31] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," in *Proc. 6th Indian Control Conf.*, 2019, pp. 299–300.

[32] Y. Liao, Y. Xu, H. Xu, L. Wang, Z. Yao, and C. Qiao, "MergeSFL: Split federated learning with feature merging and batch size regulation," in *Proc. IEEE 40th Int. Conf. Data Eng.*, 2024, pp. 2054–2067.

[33] Y. Liao, Y. Xu, H. Xu, Z. Yao, L. Huang, and C. Qiao, "ParallelSFL: A novel split federated learning framework tackling heterogeneity issues," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, 2024, pp. 845–860.

[34] C. He et al., "Fedgraphnn: A federated learning system and benchmark for graph neural networks," 2021, *arXiv:2104.07145*.

[35] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, "Communication-efficient decentralized machine learning over heterogeneous networks," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 384–395.

[36] H. Xu, M. Chen, Z. Meng, Y. Xu, L. Wang, and C. Qiao, "Decentralized machine learning through experience-driven method in edge networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 515–531, Feb. 2022.

[37] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2393–2406, Oct. 2019.

[38] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2023, pp. 1–10.

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

[40] C. H. Papadimitriou and M. Yannakakis, "The complexity of facets (and some facets of complexity)," in *Proc. 14th Annu. ACM Symp. Theory Comput.*, 1982, pp. 255–260.

[41] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[42] Z. Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 1871–1879.

[43] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[44] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.

[45] "Nvidia jetson devices," 2024. [Online]. Available: https://docs.nvidia.com/jetson/

[46] D. Merkel et al., "Docker: Lightweight linux containers for consistent development and deployment," *Linux j*, vol. 239, no. 2, 2014, Art. no. 2.

[47] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 8026–8037.

[48] "socket library," 2024. [Online]. Available: https://github.com/python/cpython/blob/3.10/Lib/socket.py

[49] Z. Jiang, Y. Xu, H. Xu, Z. Wang, J. Liu, and C. Qiao, "Semi-supervised decentralized machine learning with device-to-device cooperation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9757–9771, Oct. 2024.

[50] J. Liu, J. Liu, H. Xu, Y. Liao, Z. Wang, and Q. Ma, "YOGA: Adaptive layer-wise model aggregation for decentralized federated learning," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1768–1780, Apr. 2024.

[51] Y. Duan, X. Li, and J. Wu, "Topology design and graph embedding for decentralized federated learning," *Intell. Converged Netw.*, vol. 5, no. 2, pp. 100–115, Jun. 2024.

[52] L. Guo, Z. Yuan, X. Li, Y. Zhu, M. Qu, and W. Wang, "DFed-SST: Building semantic-and structure-aware topologies for decentralized federated graph learning," 2025, *arXiv:2508.11530*.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[54] G. Corso, H. Stark, S. Jegelka, T. Jaakkola, and R. Barzilay, "Graph neural networks," *Nature Rev. Methods Primers*, vol. 4, no. 1, 2024, Art. no. 17.

[55] K. Sharma et al., "A survey of graph neural networks for social recommender systems," *ACM Comput. Surv.*, vol. 56, no. 10, pp. 1–34, 2024.

[56] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artif. Intell. Rev.*, vol. 57, no. 4, 2024, Art. no. 99.

[57] R. Liu, P. Xing, Z. Deng, A. Li, C. Guan, and H. Yu, "Federated graph neural networks: Overview, techniques, and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 3, pp. 4279–4295, Mar. 2025.

[58] W. Huang, G. Wan, M. Ye, and B. Du, "Federated graph semantic and structural learning," in *Proc. Thirty-Second Int. Joint Conf. Artif. Intell.*, 2023, pp. 3830–3838.

[59] Y. He, D. Yan, and F. Chen, "Hierarchical federated learning with local model embedding," *Eng. Appl. Artif. Intell.*, vol. 123, 2023, Art. no. 106148.

[60] C. Chen, Z. Xu, W. Hu, Z. Zheng, and J. Zhang, "Fedgl: Federated graph learning framework with global self-supervision," *Inf. Sci.*, vol. 657, 2024, Art. no. 119976.

[61] G. Wan, W. Huang, and M. Ye, "Federated graph learning under domain shift with generalizable prototypes," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 14, pp. 15429–15 437.

[62] R. Lei et al., "Federated learning over coupled graphs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1159–1172, Apr. 2023.

[63] J. Feng, Z. Wang, Z. Wei, Y. Li, B. Ding, and H. Xu, "Federated heterogeneous contrastive distillation for molecular representation learning," in *Proc. 33 rd ACM Int. Conf. Inf. Knowl. Manage.*, 2024, pp. 1038–1048.

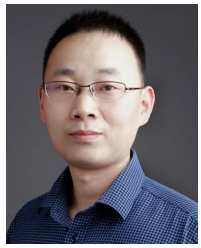
[64] E. T. M. Beltrán et al., "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2983–3013, Fourthquarter 2023.

[65] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "SpreadGNN: Decentralized multi-task federated learning for graph neural networks on molecular data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 6, pp. 6865–6873.

[66] B. Xie, C. Hu, H. Huang, J. Yu, and H. Xia, "DCI-PFGL: Decentralized cross-institutional personalized federated graph learning for IoT service recommendation," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13837–13850, Apr. 2024.
[67] A. Li et al., "Federated graph learning with adaptive importance-based sampling," 2024, *arXiv:2409.14655*.
[68] E. Krasanakis, S. Papadopoulos, and I. Kompatsiaris, "p2pGNN: A decentralized graph neural network for node classification in peer-to-peer networks," *IEEE Access*, vol. 10, pp. 34755–34 765, 2022.
[69] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023.
[70] Y. Lin, C. Chen, C. Chen, and L. Wang, "Improving federated relational data modeling via basis alignment and weight penalty," 2020, *arXiv:2011.11369*.
[71] J. Guo, S. Li, and Y. Zhang, "An information theoretic perspective for heterogeneous subgraph federated learning," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2023, pp. 745–760.
[72] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, *arXiv:1810.00826*.
[73] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
[74] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized sgd with changing topology and local updates," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5381–5393.
[75] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 01, pp. 5693–5700.

Shilong Wang received the BS degree from Sichuan University in 2022. He is currently working toward the doctor's degree with the School of Computer Science, University of Science and Technology of China (USTC). His research interests include edge computing, deep learning, and federated learning.

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



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

Chenxia Tang received the BS degree in computer science and technology in 2023 from the University of Science and Technology of China, where he is currently working toward the master's degree in computer science. His research interests include federated learning and efficient machine learning.

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

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