Federated Learning Over Coupled Graphs

Runze Lei ⬤, Pinghui Wang ⬤, *Senior Member, IEEE*, Junzhou Zhao, Lin Lan ⬤, Jing Tao, Chao Deng, Junlan Feng, *Fellow, IEEE*, Xidian Wang, and Xiaohong Guan, *Fellow, IEEE*

Abstract—**Graphs are widely used to represent the relations among entities. When one owns the complete data, an entire graph can be easily built, therefore performing analysis on the graph is straightforward. However, in many scenarios, it is impractical to centralize the data due to data privacy concerns. An organization or party only keeps a part of the whole graph data, i.e., graph data is isolated from different parties. Recently, Federated Learning (FL) has been proposed to solve the data isolation issue, mainly for Euclidean data. It is still a challenge to apply FL on graph data because graphs contain topological information which is notorious for its non-IID nature and is hard to partition. In this work, we propose a novel FL framework for graph data, FedCog, to efficiently handle coupled graphs that are a kind of distributed graph data, but widely exist in a variety of real-world applications such as mobile carriers' communication networks and banks' transaction networks. We theoretically prove the correctness and security of FedCog. Experimental results demonstrate that our method FedCog significantly outperforms traditional FL methods on graphs. Remarkably, our FedCog improves the accuracy of node classification tasks by up to 14.7%.**

Index Terms—**Distributed graph processing, federated learning, graph neural networks, privacy preservation.**

I. INTRODUCTION

GRAPHS are pervasive to represent real-world data such as social relations, traffic networks, financial transactions, and knowledge databases. To build a graph, we collect the items of interest as nodes and the relations among them as edges. Ideally, all nodes and edges in a graph are owned by a single party. Hence, the whole graph can be easily built and performing graph analysis on the graph is straightforward.

However, in many real-world scenarios, different parts of the graph data are often owned by different parties. Due to benefit conflicts, security issues, legal restrictions, privacy concerns, etc., each party's graph data should be protected and is forbidden to be accessed by other parties. For example, a mobile carrier (resp. a bank) keeps the information of its customers, along with the communication (resp. financial transaction) records among customers. Each mobile carrier's data forms an individual graph, which is highly confidential. Meanwhile, customers in one mobile carrier (resp. a bank) may have connections to customers in another mobile carrier (resp. another bank), e.g., an AT&T customer may call a T-Mobile customer. Thus, each party's graph is only a subgraph of the underlying graph, which is distributed on multiple parties, and a party cannot access other parts of the graph possessed by other parties.

In the literature, Federated Learning (FL) [1], [2] is a machine learning framework to handle the data isolation issue, which collaboratively unites a set of parties to jointly train a model while preserving each party's data privacy. Recently, FL has also been applied to handle distributed graph data [3], [4], [5]. The general structure of federated learning on distributed graph data is shown in Fig. 1. Each FL party maintains a subgraph of the global graph and trains GNN models on its local graph data. Most prior works (e.g., FedGraphNN [5]) treat the graph data located on each party independently and ignore inter-connections between different parties. In other words, these graph FL methods do not concern the *coupling* of parties' graph data. However, the coupling of graph structure widely exists. For example, different mobile carriers' communication networks are coupled by inter-phone calls (i.e., calls between users of different carriers), and different banks' transaction networks are coupled by inter-money transactions. These *inter-edges* are critical for some applications (e.g., money laundering detection), while it is challenging to exploit these edges in this FL setting.

FedSage+ [6] is proposed as a subgraph FL method based on graph generation. Generative models are trained by graph impairing and then generate the missing neighbor nodes and edges. Thus, FL parties can train GNN models on the mended local graphs and improve the performance of federated GNN models. However, the generated nodes tend to follow the local distribution, instead of representing the missing neighbors on the other subgraphs. FedSage+ resolves this problem by introducing a loss term that computes the distances between generated

Runze Lei, Pinghui Wang, Junzhou Zhao, Lin Lan, and Jing Tao are with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China (e-mail: xjtu2140506016@stu.xjtu.edu.cn; phwang@mail.xjtu.edu.cn; junzhou.zhao@xjtu.edu.cn; llan@sei.xjtu.edu.cn; jtao@mail.xjtu.edu.cn).

Chao Deng and Junlan Feng are with the China Mobile Research Institute, Beijing 100875, China (e-mail: dengchao@chinamobile.com; fengjunlan@chinamobile.com).

Xidian Wang is with the China Mobile Group Design Institute, Beijing 100080, China (e-mail: westpointwp@qq.com).

Xiaohong Guan is with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China, and also with the Center for Intelligent and Networked Systems, Tsinghua National Lab for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xhguan@mail.xjtu.edu.cn).

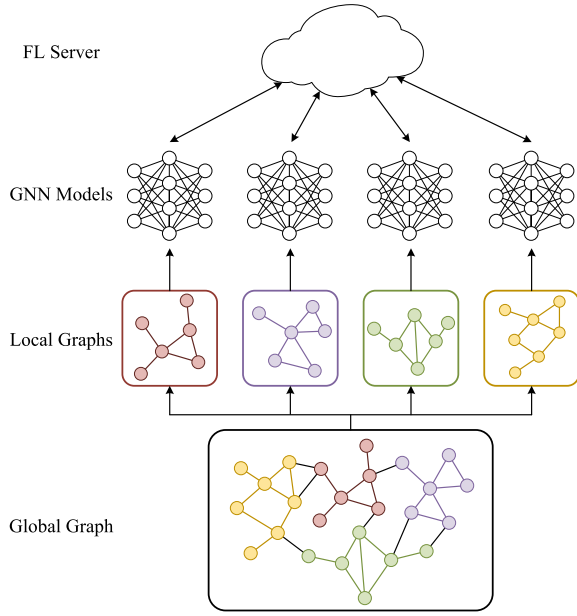Digital Object Identifier 10.1109/TPDS.2023.3240527

Fig. 1. The general structure of federated learning on distributed graph data. Each FL party possesses a subgraph of the global graph. However, a portion of topological information on the global graph is missing on the local graphs. Some edges on the global graph connect nodes held by different FL parties, so none of the parties can utilize these edges when training the GNN models.

node features and external features. This scheme increases the similarities between generated local nodes and external nodes, but is also sensitive to the initial values of the generative model and suffers from node mismatching. Therefore, FedSage+ may not work properly in highly non-IID FL tasks.

In this work, we propose a novel FL framework for graph data, *FedCog*, to efficiently handle *coupled graphs*, where the inter-edges between different parties are critical for graph learning tasks. In the setting of FedCog, graph data is distributed on several parties, and each party keeps a set of nodes as well as their features. A party's graph data not only consists of intra-edges that connect its local nodes but also inter-edges that connect its local nodes to the nodes possessed by other parties. We theoretically prove the correctness and security of FedCog. We show that performing Graph Convolutional Network (GCN) [7] on distributed graph data with FedCog is the same as performing GCN over the underlying global graph.

Our main contributions are summarized as follows.
- We formally define the problem of FL over coupled graphs. The problem is general and widely exists in many real-world applications. (Section III)
- We propose FedCog, an effective method to solve the problem of FL over coupled graphs. The method can be easily integrated with many existing FL frameworks and popular GNN models. (Section IV)
- We conduct extensive experiments on 12 real-world datasets. Compared with straightforward FL methods, Fed-Cog outperforms by up to 14.7% accuracy in node classification tasks and 0.232 AUC in link prediction tasks. (Section V)

II. PRELIMINARIES

Graph Convolutional Networks (GCNs) [7] approximate spectral graph convolutions to build GNN models. Formally, let $\mathcal{G} = (V, E, \boldsymbol{X})$ denote the graph of interest, where $V = \{v_1, \ldots, v_n\}$ is a set of nodes, E is a set of edges, and $\boldsymbol{X} \in \mathbb{R}^{n \times F}$ is the node feature matrix. Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ denote the adjacency matrix of $\mathcal{G}$, and the (u, v) entry a_{uv} indicates whether there exists an edge from node u to node v. We define the degree matrix $\boldsymbol{D} = \text{diag}(\{d_u\}_{u \in V})$ where $d_u = \sum_{v \in V} a_{uv}$ is the degree of node u. Let $\tilde{\boldsymbol{A}} \triangleq \boldsymbol{A} + \boldsymbol{I}$ denote the Self-Loop-Augmented (SLA) adjacency matrix [8], which is obtained by adding an identity matrix to the original adjacency matrix of the graph. Besides, $\tilde{\boldsymbol{D}}$ is the degree matrix of $\tilde{\boldsymbol{A}}$. Define $\boldsymbol{S}$ as the normalized adjacency matrix, i.e.,

$$\boldsymbol{S} \triangleq \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}}. \tag{1}$$

GCN computes node embeddings via layer-wise neighborhood aggregation and representation transformation, and learns node embeddings by stacking multiple layers. Specifically, GCN computes nodes' l-th layer embedding as:

$$\boldsymbol{H}^{(l+1)} = \sigma(\boldsymbol{S}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}), \quad l = 0, 1, \ldots, L-1, \tag{2}$$

where $\sigma(\cdot)$ is a non-linear activation function, $\boldsymbol{H}^{(l)}$ and $\boldsymbol{W}^{(l)}$ are the node embedding matrix and the trainable feature transformation parameters at the l-th layer, respectively. L is the number of GCN layers. GCN sets $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. It is easy to find that the above graph convolution can also be represented as a feature propagation from each node to its neighbors, that is,

$$\boldsymbol{h}_u^{(l+1)} = \sigma\left(\sum_{v \in \mathcal{N}_u} c_{uv} \boldsymbol{h}_v^{(l)} \boldsymbol{W}^{(l)}\right), \quad u \in V, \tag{3}$$

where $\boldsymbol{h}_u^{(l)}$ is the representation of node u, $\mathcal{N}_u = \{v : (u, v) \in E\} \cup \{u\}$ is is the set of node u's neighbors in the graph $\mathcal{G}$ with SLA adjacency matrix $\tilde{\boldsymbol{A}}$, and $c_{uv} = 1/\sqrt{(1 + d_u)(1 + d_v)}$.

Simple Graph Convolution (SGC) [9] hypothesizes that the non-linearity between GCN layers is not critical and can be removed. For the l-th GCN layer, SGC converts the input node representation matrix $\boldsymbol{H}^{(l)}$ to output node representation matrix $\boldsymbol{H}^{(l+1)}$ by

$$\boldsymbol{H}^{(l+1)} = \boldsymbol{S}\boldsymbol{H}^{(l)}. \tag{4}$$

SGC iteratively operates (4) for L layers, followed by a softmax operation in the final layer to obtain probabilistic outputs. SGC with such simplification tricks achieves comparable performance as the regular GCN in many applications while greatly outperforming in efficiency.

Federated Learning aims to learn model parameters θ that minimizes the loss over all parties, where parties jointly learn a model while preserving each party's data privacy. If m parties participate in the FL task, and each party holds local data and labels $(\boldsymbol{X}_i, \boldsymbol{y}_i)$, the optimization task is

$$\min_\theta F^{\text{fed}}(\theta) = \sum_{i=1}^m p_i F_i(\theta),$$

$$F_i(\theta) = \mathcal{L}(\text{NN}(\boldsymbol{X}_i; \theta), \boldsymbol{y}_i), \tag{5}$$

where p_i is the weight for party i's local objective function. NN indicates a neural network model with parameter θ. $\mathcal{L}$ is the loss function. FedAvg [1] uses federated gradient descent to solve the optimization problem. FedAvg computes local gradients $g_i = \nabla_\theta F_i(\theta)$ on each party i, and aggregates the local gradients by weighted averaging $\bar{g} = \sum_i p_i g_i$. The average gradient is then used to update the model parameters θ via gradient descent.

III. FORMULATED PROBLEM

A. Coupled Graphs

In this work, we consider a situation that a graph $\mathcal{G} = (V, E, X)$ is distributed over m parties and each party can only access its own data. These parties want to jointly learn a model to address graph learning tasks such as node classification and link prediction. Specifically, the graph data accessible by party i is denoted by $\mathcal{G}_i = (V_i, V_i^*, E_i, \boldsymbol{X}_i)$, where $V_i \subseteq V$ is the set of *internal nodes* of which features $\boldsymbol{X}_i$ are possessed by party i and V_i^* is a set of *external border nodes* that party i's internal nodes connect to, i.e., $V_i^* = \{v^* : (v, v^*) \in E, v \in V_i, v^* \notin V_i\}$. A node's feature vector is generally private data and is only held by one party. Thus, in this paper, we assume that different parties' internal nodes have no overlapping, i.e., $V_i \cap V_j = \emptyset$, $i \neq j$. E_i is the set of edges accessible by party i, including both *intra-edges* (i.e., edges between internal nodes) and *inter-edges* (i.e., edges from internal nodes to external nodes). We refer to subgraphs $\{\mathcal{G}_i\}_{i=1}^m$ as *coupled graphs*.

Take the mobile phone network as an instance. $\mathcal{G}_i$ refers to mobile carrier i's communication network. V_i represents the set of the mobile carrier's customers, and $\boldsymbol{X}_i$ records its customers' features (e.g., age and gender). Thus, $\boldsymbol{X}_i$ is private information of party i and should never be revealed to the server or other parties during training. V_i^* consists of other carriers' customers that mobile carrier i's customers communicated with. For any customer in V_i^*, its features are *not* available for mobile carrier i. The edges represent communications between customers. The external nodes and inter-edges are essential information for the mobile carriers' service. Therefore, external nodes and inter-edges in the local graph $\mathcal{G}_i$ do not cause privacy risk. Many real-world scenarios are similar, such as financial networks between banks, document networks between academic databases, and social networks between email service providers. These scenarios can be modeled as coupled graph problems and solved by our method.

B. Federated Learning on Coupled Graphs

We aim to perform graph learning tasks such as node/graph classification and link prediction on coupled graphs $\{\mathcal{G}_i\}_{i=1}^m$ under the constraint that party i can only access its own graph data $\mathcal{G}_i$. In regular FL tasks, local losses $\mathcal{L}_i, i = 1, \ldots, m$ are independent, i.e., $\mathcal{L}_i$ and its gradient g_i could be locally computed on party i. Thus, FedAvg [1] achieves gradient descent for a global loss $\mathcal{L}$ by aggregating local gradients. For graph learning tasks, information from neighbor nodes is crucial. If we suppose that the server collects subgraph data without concern

for privacy, the ideal global model is

$$\theta^* = \underset{\theta}{\arg\min} \ \mathcal{L}(\text{GNN}(E, \boldsymbol{X}; \theta), \boldsymbol{y}), \qquad (6)$$

where E is graph $\mathcal{G}$'s edge set, and $\boldsymbol{X}, \boldsymbol{y}$ are node features in graph $\mathcal{G}$ and target labels respectively. The GNN model can utilize complete neighbor information of graph $\mathcal{G}$ in this case. However, access to the nodes and edges of other parties is restricted due to data privacy. The neighborhood and long-range relationship of nodes are damaged when ignoring these external nodes and edges. If the subgraphs are disconnected during federated GNN training, the local optimization task in (5) becomes

$$F_i(\theta) = \mathcal{L}(\text{GNN}(E_i^{\text{local}}, \boldsymbol{X}_i; \theta), \boldsymbol{y}_i), \qquad (7)$$

where $E_i^{\text{local}} = E_i \cap (V_i \times V_i)$ is the local edge set of subgraph $\mathcal{G}_i$. We observe that the federated objective shown by (5) and (7) may be distinctly different from that in (6) because a part of neighbor information is missing in the federated scheme. Our experiments later in Section V reveal that the disconnection of coupled graphs will significantly deteriorate the performance of FL models.

In this work, we propose FedCog which constructs the coupled information on parties, still without direct access to sensitive data. Our method encourages the parties to share desensitized information securely. The insensitive data $\boldsymbol{Z}$ are utilized to compose valid knowledge for other FL parties. The local task of FedCog is

$$F_i^{\text{coupled}}(\theta) = \mathcal{L}(\text{FedCog}(E_i, \boldsymbol{X}_i, \{\boldsymbol{Z}_{j \to i}\}; \theta), \boldsymbol{y}_i), \quad (8)$$

where $\boldsymbol{Z}_{j \to i}$ represents external data from party j to i. Our FedCog allows foreign data to engage in local training and addresses the missing neighbor problem. Meanwhile, data privacy is theoretically guaranteed by our method.

IV. OUR METHOD

As we mentioned, the primary difficulty of the coupled graph problem is the data dependence among parties. A party's GCN model needs representations of all neighboring nodes to achieve graph convolution (as shown in (3)). However, in the coupled graph problem, the neighboring nodes are possessed by other data owners who refuse to share their original graph data. Our method FedCog provides a surrogate approach to reduce data dependence. FedCog uses a novel graph transform operation *Graph Decoupling*. Graph Decoupling converts each original local graph into two decoupled local graphs. Accordingly, a graph convolution layer in GCN is divided into two propagation operations *Internal Propagation* and *Boarder Propagation* on the two decoupled local graphs respectively.

A. Graph Decoupling

The coupling of graphs causes data dependence among parties. That is, a party cannot run GCN locally due to the lack of graph data held by other parties. The first step of our method, FedCog, is proposed to eliminate the inter-party data dependence in GCN computation by decoupling the graphs. Specifically,
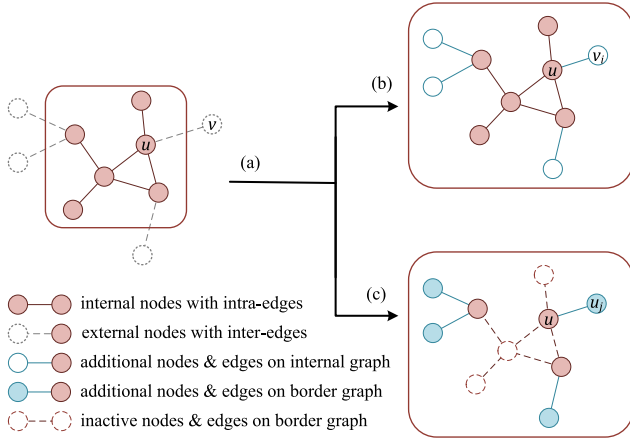
Fig. 2. Graph decoupling of FedCog. (a) is the local graph of party i. The party does not have full knowledge of the inter-edges and some neighbor nodes. (b) and (c) show the decoupled graphs of (a). (b) is the *internal graph*. The missing inter-edges and neighbor nodes are added to the local graph. (c) shows the *border graph*, which also contains extra nodes and edges, but the edges between local internal nodes are removed. Thus the border graph is a bipartite graph of local nodes and extra nodes.

we decompose global graph $\mathcal{G}$ into *internal graphs* and *border graphs* for the FL parties.

To build an *internal graph* $\mathcal{G}_i^I$, party i combines its local graph $\mathcal{G}_i$ with the inter-edges which are connected to its internal nodes, and introduces additional nodes to complete the inter-edges. For example, as shown in Fig. 2 from (a) to (b), $u \in V_i$ is party i's internal node. $v \in V_j \cap V_i^*$ is a node of party j's local graph, and also an external border node of party i, i.e., (u, v) is an inter-edge. We introduce an additional node v_i into the local graph as a substitute for node v, and connect (u, v_i). Seeing that party i does not have knowledge about node v, the additional node v_i's feature vector is initialized to zero.

The first step to build a *border graph* $\mathcal{G}_i^B$ is similar to the internal graph. Additional nodes and edges are added to party i's local graph $\mathcal{G}_i$ according to the inter-edges. Furthermore, party i removes all intra-edges on its local graph $\mathcal{G}_i$, and then the isolated internal nodes. An instance is illustrated in Fig. 2 from (a) to (c). Since an inter-edge (u, v) connects party i's internal node u with another node v on party j's local graph $\mathcal{G}_j$, we add a node u_j as well as an edge (u, u_j) to party i's local graph. After that, all intra-edges are removed. The nodes with no inter-edges become isolated nodes on the border graph, and thus can be ignored. Therefore, a border graph is a bipartite graph. Each edge in $\mathcal{G}_i^B$ connects an internal node and an additional node. Both graphs $\mathcal{G}_i^I$ and $\mathcal{G}_i^B$ are locally held by party i and then we will see that the graph decoupling facilitates FL over coupled graphs.

B. Internal and Border Propagation

As the local graph $\mathcal{G}_i$ is decoupled into two graphs, the graph convolution in GCN is accordingly divided into two sequential steps: *Internal Propagation* and *Border Propagation*, which are modified graph convolutions.

Internal Propagation is a graph convolution implemented on $\mathcal{G}_i^I$:

$$\hat{h}_u^{(l+\frac{1}{2})} = \sum_{v \in \mathcal{N}_u^I} \beta_{uv} \hat{h}_v^{(l)} W^{(l)}, \qquad (9)$$

where u is any node on internal graph $\mathcal{G}_i^I$, including internal nodes in V_i and the additional nodes introduced during graph decoupling. The superscript $l + \frac{1}{2}$ indicates the intermediate node representation after the first half of the two-step propagation. Internal Propagation differs from the graph convolution in (3) in two aspects: 1) a linear function is used as the activation function; 2) the weighted coefficient of neighbor representations is modified to $\beta_{uv} = 1/\sqrt{1 + d_v}$ rather than c_{uv}. Note that this procedure is executed on graph $\mathcal{G}_i^I$, and so the neighbors $\mathcal{N}_u^I$ and degrees d_u are also defined on the decoupled graph $\mathcal{G}_i^I$.

After Internal Propagation, each party sends the intermediate representations of internal nodes in $\mathcal{G}_i^I$ (i.e. $h_{v_i}^{(l+\frac{1}{2})}$) to the party possessing node v. This data transmission can be implemented on a decentralized fully-connected communication network as well as with the help of a centralized server. Privacy analysis of this procedure is in Section IV-E.

Border Propagation is then executed on the bipartite local graph $\mathcal{G}_i^B$.

$$\hat{h}_u^{(l+1)} = \sigma\left(\sum_{v \in \mathcal{N}_u^B} \gamma_{uv} \hat{h}_v^{(l+\frac{1}{2})}\right), \qquad (10)$$

where $\gamma_{uv} = 1/\sqrt{1 + d_u}$. Here, d_u is the degree of node u in graph $\mathcal{G}_i^I$, and $\mathcal{N}_u^B$ is node u's neighbors in graph $\mathcal{G}_i^B$ with self-connections.

The following theorem demonstrates that our FedCog on coupled graphs $\{\mathcal{G}_i\}_{i=1}^m$ is equivalent to a graph convolution on global graph $\mathcal{G}$.

Theorem 1. Given a GCN on global graph $\mathcal{G} = (V, E, X)$, i.e.,

$$H^{(l)} = \text{GraphConv}(H^{(l-1)}; W^{(l-1)}), \quad l = 1, \ldots, L,$$

where $H^{(0)} = X$ and GraphConv is an operation of applying (3) to all nodes in V. Let $\hat{H}$ be the node representations obtained via (9) and (10). Then, we have

$$\hat{H}^{(l)} = H^{(l)}, \quad l = 1, \ldots, L.$$

Proof. Let node $u \in V_i$ be an internal node of party i, and u_j be an additional node on the border graph (see Fig. 2(c)). From (9), we have

$$\hat{h}_{u_j}^{(l+\frac{1}{2})} = \sum_{v \in \mathcal{N}_{u_j}^I} \beta_{u_j v} \hat{h}_v^{(l)} W^{(l)} \qquad (11)$$

Note B_u as the set of parties $j \neq i$ that hold node u as an external border node, i.e., $B_u = \{l | l \in \{1, \ldots, m\} \setminus \{i\} \land u \in V_l^*\}$. Thus,

$$\hat{h}_u^{(l+1)} = \sigma\left(\gamma_{uu} \hat{h}_u^{(l+\frac{1}{2})} + \sum_{j \in B_u} \gamma_{uu_j} \hat{h}_{u_j}^{(l+\frac{1}{2})}\right)$$

$$= \sigma \left(\sum_{j \in B_u} \sum_{v \in \mathcal{N}_{u_j}^I} c_{uv} \hat{\boldsymbol{h}}_v^{(l)} \boldsymbol{W}^{(l)} \right), \qquad (12)$$

where c_{uv} is defined in (3). Remember the graph decoupling method, it is easy to find that the result in (12) is the same as (3).

C. Towards Efficient FedCog

In the setting of federated learning, to train the parameters of neural networks, the communication cost for repeating FedCog in every training iteration is expensive. Moreover, the back-propagation in gradient descent also requires the cooperation of parties, aggravating the communication cost. We overcome these disadvantages by cleverly exploiting the properties of GCN variants, such as SGC [9], APPNP [10], GBP [11], GPR-GNN [12], etc.

As we mentioned in Section II, SGC is a variant of GCN which removes the linear transforms and nonlinear activation functions between graph convolution layers. Previous works suggest that SGC has a competitive performance over vanilla GCN [9]. A crucial feature of SGC when working with FedCog is the removal of trainable weights between graph convolution layers. That is, the outputs of successive SGC layers are constant during the training of model weights. Therefore, when SGC is adopted, we can compute the outputs of SGC layers only once, and then train model weights iteratively without repeating graph convolutions. This allows FedCog to run the decoupling and propagation operations only once, instead of computing FedCog propagations in every training iteration. After FedCog propagations, each party stores the embeddings of its nodes and learns the weights of neural networks via FedAvg [1]. These processes are summarized in Algorithm 1.

In Algorithm 1, lines 2-3 are the graph decoupling procedures introduced in Section IV-A. Each party applies the transform to its local graph $\mathcal{G}_i$ and obtains an internal graph $\mathcal{G}_i^I$ and a border graph $\mathcal{G}_i^B$. Lines 5-9 show the steps of internal and border propagations introduced in Section IV-B. As we change the model from vanilla GCN to SGC, the propagation operation changes from (9) and (10) into (13) and (14) accordingly

$$\hat{\boldsymbol{h}}_u^{(l+\frac{1}{2})} = \sum_{v \in \mathcal{N}_u^I} \beta_{uv} \boldsymbol{h}_v^{(l)}, \qquad (13)$$

$$\hat{\boldsymbol{h}}_u^{(l+1)} = \sum_{v \in \mathcal{N}_u^B} \gamma_{uv} \hat{\boldsymbol{h}}_v^{(l+\frac{1}{2})}. \qquad (14)$$

The correctness can be easily verified similar to Theorem 1. When FedCog is deployed with SGC models, the propagation steps are executed only once. Then in lines 11-17, we run the federated training steps, which will be explained in Section IV-D.

In addition to SGC, some more GNN models also output constant graph convolution results during model training and work efficiently with FedCog. APPNP [10] uses the predict-then-propagate framework to build neural networks, where its propagation is achieved by approximating Personalized PageRanks. The major procedures of FedCog-APPNP are similar to SGC. The only difference occurs at Border Propagation. It

---

Algorithm 1: FedCog-SGC.

Input: m FL parties with their local subgraphs $\{\mathcal{G}_i\}_{i=1}^m$, L-layer SGC model $f(\mathcal{G}; \theta), \theta = \{\boldsymbol{W}^{(0)}, \ldots, \boldsymbol{W}^{(L-1)}\}$

1: $\triangleright$ Graph Decoupling
2: **for** $i = 1$ to m **do** $\triangleright$ Execute on party i
3:     $\mathcal{G}_i^I, \mathcal{G}_i^B \leftarrow$ GraphDecoupling$(\mathcal{G}_i)$
4: $\triangleright$ Internal and Border Propagations
5: **for** $l = 0$ to $L - 1$ **do**
6:     **for** $i = 1$ to m **do** $\triangleright$ Execute on party i
7:        $\hat{\boldsymbol{h}}_u^{(l+\frac{1}{2})} \leftarrow \sum_{v \in \mathcal{N}_u^I} \beta_{uv} \boldsymbol{h}_v^{(l)}$
8:        Receive $\{\hat{\boldsymbol{h}}_{u_v}^{(l+\frac{1}{2})} : v \in V_j\}$ from party j
9:        $\hat{\boldsymbol{h}}_u^{(l+1)} \leftarrow \sum_{v \in \mathcal{N}_u^B} \gamma_{uv} \hat{\boldsymbol{h}}_v^{(l+\frac{1}{2})}$
10: $\triangleright$ Federated Training
11: **for** $t = 1$ to k **do**
12:     Sample a set of parties $\mathcal{C}_t$
13:     **for** $i \in \mathcal{C}_t$ **do** $\triangleright$ Execute on party i
14:        $\theta_i \leftarrow \theta$
15:        $\boldsymbol{g}_i \leftarrow$ GradientDescent$(\theta_i, \{\hat{\boldsymbol{h}}_u^{(L)}\}_{u \in V_i})$
16:     $\bar{\boldsymbol{g}} \leftarrow \sum_{i \in \mathcal{C}_t} p_i^{(t)} \boldsymbol{g}_i$ $\triangleright$ Execute on FL Server
17:     $\theta \leftarrow \theta - \eta \bar{\boldsymbol{g}}$

---

modifies (14) to

$$\hat{\boldsymbol{h}}_u^{(l+1)} = (1 - \alpha) \sum_{v \in \mathcal{N}_u^B} \gamma_{uv} \hat{\boldsymbol{h}}_v^{(l+\frac{1}{2})} + \alpha \hat{\boldsymbol{h}}_u^{(0)}, \qquad (15)$$

where $\alpha \in (0, 1]$ is the restart probability of Personalized PageRank.

GBP [11] extends the feature propagating method in SGC to Generalized PageRank (GPR) [13]. The propagation is computed through $\boldsymbol{H}^{(l+1)} = \tilde{\boldsymbol{D}}^{r-1} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-r} \boldsymbol{H}^{(l)}, r \in [0, 1]$. Therefore, to apply our method to GPR, we modify the coefficients in (13) and (14) to

$$\beta_{uv} = \frac{1}{(1 + d_v)^{1-r}}, \qquad (16)$$

$$\gamma_{uv} = \frac{1}{(1 + d_u)^r}. \qquad (17)$$

GPR-GNN [12] is also a GNN model motivated by GPR. From the computation perspective, GPR-GNN can be viewed as applying a weighted sum to the hidden features of SGC. Therefore, our method is also feasible for GPR-GNN.

We take the SGC model as an example to analyze the computation and communication complexities of our method FedCog. Suppose there are m parties collaborating to train the FL model. Each party keeps the graph data with n nodes, e_I intra-edges and e_E inter-edges. The node features in the graphs are embedded in the F-dim space. When we use a GNN model with L graph convolutional layers and apply k iterations of federated training, the computation and communication costs are summarized in Table I.

From the aspect of computation cost, FedCog utilizes the information of inter-edges, which is not involved in original

TABLE I
COSTS OF GRAPH FL METHODS

Method	Computation	Communication
GCN	$O(kmLe_IF + kmLnF^2)$	$O(kmLF^2)$
SGC	$O(mLe_IF + kmnF^2)$	$O(kmF^2)$
FedCog-SGC	$O(mL(e_I + e_E)F + kmnF^2)$	$O(mnF + kmF^2)$

SGC-based Federated Learning. The computation cost of Fed-Cog is larger than SGC during propagation, but FedCog keeps the same efficiency as SGC in training steps, which is faster than GCN. As for the communication cost, FedCog makes a coupled feature propagation and leads to a higher cost than SGC. The communication cost of FedCog is in proportion to the size of graph n, so it will be expensive to apply FedCog to large graphs. Despite this, our later experiments show that such costs are not extremely expensive, which is acceptable in practice.

D. Federated Training

FedCog is designed as a substitute propagation method for GNN models and does not affect the model parameters' local and global update steps during federated training. Therefore, FedCog can be implemented as a plug-in algorithm for various FL methods. For instance, GNN parameters can be trained with FedAvg [1]. The procedure is shown in lines 11-17 in Algorithm 1. We can also apply FedCog to other FL training frameworks, such as FedDyn [14] and FedOpt [15] to improve the model convergence, especially when statistical heterogeneity occurs in parties' local graph data.

E. Privacy Preservation

The coupled graph problems often arise among company or organization parties. The parties are generally honest and trusted. However, the risk of malicious adversarial parties should never be neglected. The propagation and federated training may cause privacy leakage to the adversarial server and parties. We analyze the privacy security of our FedCog in this section.

• *Privacy Security of Propagation.* During the propagation procedures, FedCog requires the FL parties to exchange some intermediate computation results. In Section IV-B, we mentioned two types of information exchange: a decentralized peer-to-peer exchange, and a centralized exchange through a server. Next, we will first introduce the weakness of FedCog with the decentralized propagation, and propose a method, named *LNNC*, to address this problem. Then, we will show that LNNC also solves the privacy problem in the centralized manner.

In practical coupled graph scenarios, a node's feature vector is usually sensitive information. For example, a feature vector may represent attributes of a user on an online social network. Therefore, FedCog must ensure that a malicious party or server cannot obtain any node feature vectors during the propagation. That is, our goal is to guarantee that any entry of party i's node features $\boldsymbol{X}_i$ will never be revealed by the server and other parties during FedCog.

In decentralized propagation, we consider a set of parties $\mathcal{C}_A$ as malicious adversaries. In our threat model, we assume that all adversarial parties are semi-honest and they may collude with each other. An adversarial party i has prior knowledge $\{\mathcal{G}_i, \mathcal{H}_j\}$, where $\mathcal{G}_i = (V_i, V_i^*, E_i, \boldsymbol{X}_i)$ is party i's local graph data, and $\mathcal{H}_j = \{\hat{\boldsymbol{h}}_{u_j}^{(l+\frac{1}{2})} | u \in V_i \cap V_j^*, l = 0, 1, \dots\}$ is the intermediate results received from a victim party j.

Adversarial party i's goal is to infer any node feature vector $\boldsymbol{x}_v, v \in V_j$ on the victim party j. Thus, the privacy scheme must avoid the leakage of party j's any node feature. We propose to protect the confidentiality of individual node features based on the following theorem.

Theorem 2. Given a victim party j's internal node $v \in V_j$. The necessary condition of collusive adversaries $\mathcal{C}_A$ obtaining the value of $\boldsymbol{x}_v$ is $\exists u \in V_j$, s.t. $\mathring{\mathcal{N}}_u \subseteq V_A$, where $V_A = \bigcup_{i \in \mathcal{C}_A} V_i$ is the union of all adversaries' internal nodes, and $\mathring{\mathcal{N}}_u = \mathcal{N}_u \setminus \{u\}$ is the set of node u's neighbors on graph G without self-connections.

Proof.

Consider the worst situation of Theorem 2, i.e., all parties except a victim party are adversaries and collusive. Then, we regard the union of all adversarial parties $\mathcal{C}_A$ as one party and simplify the case to a two-party scenario: Party A is the malicious party and Party B is the victim party. Each Party i holding a local graph $\mathcal{G}_i = (V_i, V_i^*, E_i, \boldsymbol{X}_i), i = A, B$.

(1) In SGC layer 1: For any external node $u \in V_B^*$ of the victim, we have

$$\hat{\boldsymbol{h}}_{u_B}^{(\frac{1}{2})} = \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} \beta_{u_B v} \boldsymbol{x}_v. \tag{18}$$

The adversary has no knowledge of the victim's intra-edges, therefore variable $\beta_{u_B v}$ is unknown to the adversary. Let F be the number of node feature attributes. There are F equations in (18), which might be utilized by the adversary to infer sensitive information about the victim, but the number of unknown variables $\beta_{u_B v}, \boldsymbol{x}_v, \forall v \in \mathring{\mathcal{N}}_{u_B}^I$ is $|\mathring{\mathcal{N}}_{u_B}^I|(F + 1) \geq F$. Furthermore, when selecting any r equations from the F equations in (18), there are $|\mathring{\mathcal{N}}_{u_B}^I|(r + 1) > r$ independent variables. Therefore, the adversary fails to infer any entries in $\boldsymbol{x}_v$ via $\hat{\boldsymbol{h}}_{u_B}^{(\frac{1}{2})}$.

(2) In SGC layer 2: For any external node $u \in V_B^*$ of the victim, we have

$$\hat{\boldsymbol{h}}_{u_B}^{(1+\frac{1}{2})} = \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} \beta_{u_B v} \hat{\boldsymbol{h}}_v^{(1)}$$

$$= \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} \beta_{u_B v} \gamma_{vv} \sum_{w \in \mathring{\mathcal{N}}_v \cap V_B} \beta_{vw} \boldsymbol{x}_w$$

$$+ \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} \beta_{u_B v} \gamma_{vv} \beta_{vv} \boldsymbol{x}_v + \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} \beta_{u_B v} \gamma_{vv_A} \hat{\boldsymbol{h}}_{v_A}^{(\frac{1}{2})}. \tag{19}$$

Combining (18) and (19), the adversary has $2F$ equations, and the total number of independent unknown variables is

$$T = |\mathring{\mathcal{N}}_{u_B}^I|(F + 2) + \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} |\mathring{\mathcal{N}}_v \cap V_B|(F + 1). \tag{20}$$

If the necessary condition in Theorem 2 does not hold, i.e., $|\mathring{\mathcal{N}}_v \cap V_B| \geq 1, \forall v \in V_B$, then we have

$$T = |\mathring{\mathcal{N}}_{u_2}^I|(F+2) + \sum_{v \in \mathring{\mathcal{N}}_{u_2}^I} |\mathring{\mathcal{N}}_v \cap V_2|(F+1)$$

$$\geq |\mathring{\mathcal{N}}_{u_2}^I|(F+2) + \sum_{v \in \mathring{\mathcal{N}}_{u_2}^I} (F+1) > 2F. \quad (21)$$

If we select any r equations from (18) and s equations from (19), the number of independent variables in these selected equations is $|\mathring{\mathcal{N}}_{u_B}^I|(\max(r,s)+2) + \sum_{v \in \mathring{\mathcal{N}}_{u_B}^I} |\mathring{\mathcal{N}}_v \cap V_B|(s+1) > r + s$. Therefore, the adversary still fails to infer any node features $\boldsymbol{x}_v, v \in V_B$ from $\hat{\boldsymbol{h}}_{u_B}^{(\frac{1}{2})}$ and $\hat{\boldsymbol{h}}_{u_B}^{(1+\frac{1}{2})}$.

(3) In SGC layer $L \geq 3$: The adversary has LF equations with respect to $\{\boldsymbol{x}_v\}_{v \in V_B}$. However, similar to the steps in (2), we have at least $LF + 2L - 1$ independent variables to be solved when the condition in Theorem 2 does not hold. Therefore, the node features possessed by the victim Party B cannot be inferred by the adversary Party A.

Theorem 2 reveals that the intermediate results $\mathcal{H}_j$ may be insecure in some scenarios. In addition, it also inspires us to guarantee privacy security by breaking the necessary condition. Theorem 2 indicates that the adversary can obtain party j's private information only when there are nodes in V_j which have no internal neighbors, i.e. no intra-edges, on the local graph $\mathcal{G}_j$. Thus, we propose *Local Nearest Neighbor Connection* (LNNC) to protect privacy security by adding intra-edges to the nodes without internal neighbors.

The procedure of LNNC is as follows. Before party j starts FedCog propagations, it retrieves its internal nodes in V_j which satisfy $\mathring{\mathcal{N}}_u \subseteq V_j^*$. Note that $V_j^* \subseteq V_A$. For each retrieved node u, party j selects another internal node $u_N \in V_j \setminus \{u\}$ and adds an edge (u, u_N) to graph $\mathcal{G}_j$. We notice that propagation is a weighted sum operation. Intuitively, adding an additional edge which is connected to a node with similar features will have a slight influence on the propagation results. Therefore, we select the node u_N as the nearest node in the feature space, i.e., $u_N = \arg\min_{v \in V_j \setminus \{u\}} D(\boldsymbol{x}_u, \boldsymbol{x}_v)$, where $D(\cdot, \cdot)$ is a distance metric. In this work, we use the angular distance, i.e., $D(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\pi} \cos^{-1}(\frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|})$. Using LNNC, we guarantee that each internal node of party j has at least one intra-edge. Therefore, we broke a necessary condition of privacy leakage. After using LNNC, when we start the propagating and training procedures of FedCog, the adversarial parties will not get the sensitive data.

Besides the decentralized transmission, we also consider a centralized form to exchange intermediate data. A server receives $\hat{\boldsymbol{h}}_{v_i}^{(l+\frac{1}{2})}$ and forwards it to party i. In general, the server cannot access parties' local data, so its prior knowledge is generally less than the adversarial parties. In the worst situation, the server may collude with adversarial parties, but it still cannot get more prior knowledge than the adversarial parties. Thus, privacy is preserved from the server as long as from the parties. In centralized transmission, the adversarial parties have the same prior knowledge as the decentralized scenario,

TABLE II
SUMMARY OF REAL-WORLD DATASETS

dataset	#nodes	#edges	#classes	#attributes
CORA [20]	2,708	5,429	7	1,433
CiteSeer [21]	3,327	4,732	6	3,703
PubMed [21]	19,717	44,338	3	500
DBLP [22]	17,716	52,867	4	1,639
Coauthor-CS [23]	18,333	81,894	15	6,805
Coauthor-Phy [23]	34,493	247,962	5	8,415
ACM [24]	3,025	13,128	3	1,870
BlogCatalog [25]	5,196	171,743	6	8,189
Flickr [25]	7,575	239,738	9	12,047
Amazon-CS [26]	13,752	287,209	10	767
Amazon-Photo [26]	7,650	143,663	8	745
UAI [27]	3,067	28,311	19	4,973

so the LNNC method is still effective to resolve privacy risks. Therefore, LNNC also guarantees privacy security in centralized transmission.

Notice that the correctness of Theorem 2 is established on the secrecy of β_{uv} in (9), which depends on the GNN models. Thus, LNNC cannot ensure data privacy for some GNN models, e.g., GraphSage with a mean (β_{uv} is constant) or pooling ($\beta_{uv} \in \{0, 1\}$) aggregators [16], because the attacker may obtain β_{uv} without prior knowledge of the victim's local graph structure. This is a limitation of FedCog and LNNC, while LNNC can still provide a privacy guarantee for a large number of widely used GNN models, such as GCN [7], SGC [9], APPNP [10], GBP [11], GPR-GNN [12], etc.

• *Privacy Security of Federated Training.* In the training steps, the FL server collects gradients $\boldsymbol{g}_i^{(t)}$ from parties, which may also increase the risk of privacy leakage. To address this, homomorphic encryption such as the Paillier cryptosystem [17] can be applied to achieve secure computations on the untrusted server, which prevents the untrusted server from accessing private information. In addition, previous works, such as [18], [19] providing secure aggregation methods can also be used to preserve privacy in the training steps of our FedCog.

V. EVALUATION

In this section, we evaluate our method FedCog on node classification and link prediction tasks. We show empirical results of model performance and efficiency. In our experiment, all FL servers and parties were simulated by processes on a physical device with Intel Xeon Silver 4316 CPU and NVIDIA Tesla V100S GPU.

A. Experimental Setup

• *Datasets.* Table II summarizes the 12 real-world datasets from 5 domains used for evaluating the performance of our method. CORA [20], CiteSeer, PubMed [21], and DBLP [22] are citation networks. Coauthor-CS, Coauthor-Phy [23], and ACM [24] represent co-authorship relations between authors and papers. BlogCatalog and Flickr [25] are social networks. Amazon-CS and Amazon-Photo [26] are graphs of goods that are usually bought together. UAI [27] is a graph of Internet document network including the reference relations. To simulate the situation of coupled graphs, we partitioned the datasets into

subgraphs as the preprocess. Each graph was partitioned into $\{2, 5, 10, 20, 50, 100\}$ parts, and each part was distributed to one party.

We adopted two partitioning methods to evaluate FedCog's performances in different local data distributions comprehensively: *topological clustering* and *feature clustering*. We use METIS [28] algorithm to partition graphs according to their topological structures. Feature clustering is to cluster the nodes into groups by applying K-Means to node features. Both partitioning methods generate subgraphs with no nodes in common. An important difference between the two partitioning methods is the number of intra-edges in the local graphs $\mathcal{G}_i$. The topological clustering algorithm will try to keep high edge density in the partitioned subgraphs. Thus the local graphs will have more intra-edges than local graphs partitioned by other methods. However, the feature clustering method will ignore the topological structure during partitioning. Feature clustering may lead to sparse local subgraphs. For example, when we used METIS to cut CORA and CiteSeer graphs into 100 parts, original graphs' 54.62% and 79.62% edges became subgraphs' intra-edges respectively. Our LNNC method only added 0.21% and 0.02% edges compared with the original graphs. In the case of K-Means for 100-subgraph partitioning, only 27.32% and 36.23% of edges were intra-edges in CORA and CiteSeer datasets respectively. Our LNNC method added 9.87% and 9.26% edges to satisfy privacy conditions.

The difference in local data distributions is also reflected in the unbalanced labels of local data. In general, the neighbor nodes on a graph likely have the same label, and so do the nodes with similar features. Therefore, both partitioning methods tend to cluster the nodes with the same labels into one party. This will cause unbalanced label distributions among parties. We use average earth mover distance (EMD) [29] of label distributions to measure the unbalance. EMD is in the range of $[0, 2]$, where a larger EMD means more unbalanced. If each dataset is partitioned into 100 parts, the average EMD over 12 datasets is 1.29 for METIS and 1.46 for K-Means, which shows a degree of unbalance. In particular, the average EMD of CORA, Flickr and UAI datasets under K-Means partitioning is 1.69, 1.70 and 1.71, respectively. It indicates that these datasets have highly unbalanced local data.

● *Baselines.* To show the effectiveness of FedCog, we compare the performance of FedCog with the regular FedAvg, FedSage, and FedSage+[6]. The regular FedAvg method is a simple combination of SGC [9] and Federated Learning method based on [1]. In our experiments, FedAvg and FedSage regard each party's local subgraph as an individual graph, ignoring the existence of inter-edges. FedSage+ trains a neighbor generator for each party and complements local graphs. FedSage+ utilizes both local and foreign data, so it is an available solution for coupled graph problems. As for our method FedCog, we adopt SGC as the GNN model for node classification and link prediction tasks. SGC models in the experiments contain 2 simplified graph convolutional layers. In addition, we use a centralized SGC model as a reference method. Centralized SGC collects local data from all parties and trains the model on a server without regard to data privacy, which is an ideal but impractical

setting for learning tasks. For all methods, we tune the learning rate parameter in the range of $[0.001, 0.1]$ during training.

We also conduct experiments to show FedCog's applicability to different federated training algorithms. FedAvg [1], FedDyn [14], FedAdagrad and FedAdam [15] are selected as baselines. FedCog is integrated into the four FL training algorithms and compared with the FedCog-free versions. In this experiment, the settings of SGC models are the same as the above. During the federated training of all baselines and our method, we iteratively apply 1 epoch of local model training and one round of global model update. For all four methods, learning rates are tuned in $[0.001, 0.1]$. FedDyn's parameter α is selected in $[0.001, 0.1]$. For FedAdagrad and FedAdam, we set parameter $\tau \in [10^{-5}, 10^{-1}]$. As [15] suggested, we set $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for FedAdam.

● *Metrics.* In node classification tasks, we use micro-average precision of predictions to evaluate the accuracy performance of the methods. Larger accuracy values are better results. As for link prediction tasks, we use the area under the ROC curve (AUC) to show the results as a measurement metric. ROC is the TPR-FPR curve of a binary classifier, where $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$, $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$. Larger AUC scores indicate the model has better performance.

B. Results of Node Classification

We examine the node classification performance in this experiment. For each dataset, we sample 30 nodes per class as the training set, and then uniformly sample 1,000 nodes other than training samples as the test set. METIS and K-Means are used to partition the graphs into at most 100 parts as local graph data. We achieve federated node classification tasks by baselines FedAvg, FedSage, FedSage+, and our method FedCog. In the experiments for FedSage and FedSage+, we follow the settings explained in [6]. For the FedAvg method, the GNN model contains 2 layers of Simplified Graph Convolution followed by a fully-connected layer. As for FedCog, model settings are similar to that in FedAvg. The difference is that we repeat the propagations of FedCog-SGC 2 times to simulate a 2-layer SGC on the global graph.

Fig. 3 illustrates node classification accuracy when the graphs are partitioned by K-Means node feature clustering. *FedAvg*, *FedSage* and *FedSage+* show the baseline results respectively, and *FedCog with LNNC* indicates the results of our method. To show the influence of adding intra-edges during LNNC, we also illustrate the results of *FedCog w/o LNNC* as a reference. Besides, we also conduct experiments on the entire graphs without partitioning, which is noted as *Centralized* in the results.

From the results of baseline FedAvg, we observe that on most datasets, such as CORA, DBLP, and Amazon-CS, FedAvg has a significant accuracy decrease as the number of parties increases. As a comparison, our FedCog method has better performance than FedAvg, and even maintains almost the same accuracy as no partitioning on some datasets, such as CORA and DBLP. When there are 100 parties, our FedCog's accuracy is 14.7%, 12.8%, and 10.9% higher than FedAvg on CORA, DBLP, and Amazon-CS respectively. Although on PubMed and
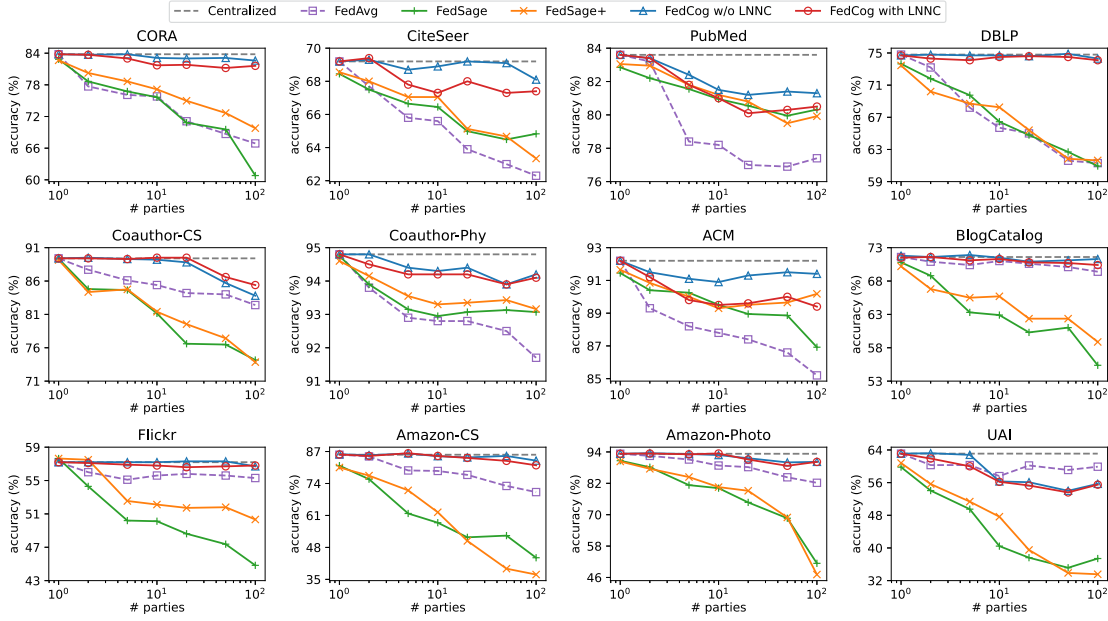
Fig. 3. Node classification accuracy when the global graphs are partitioned by K-Means.
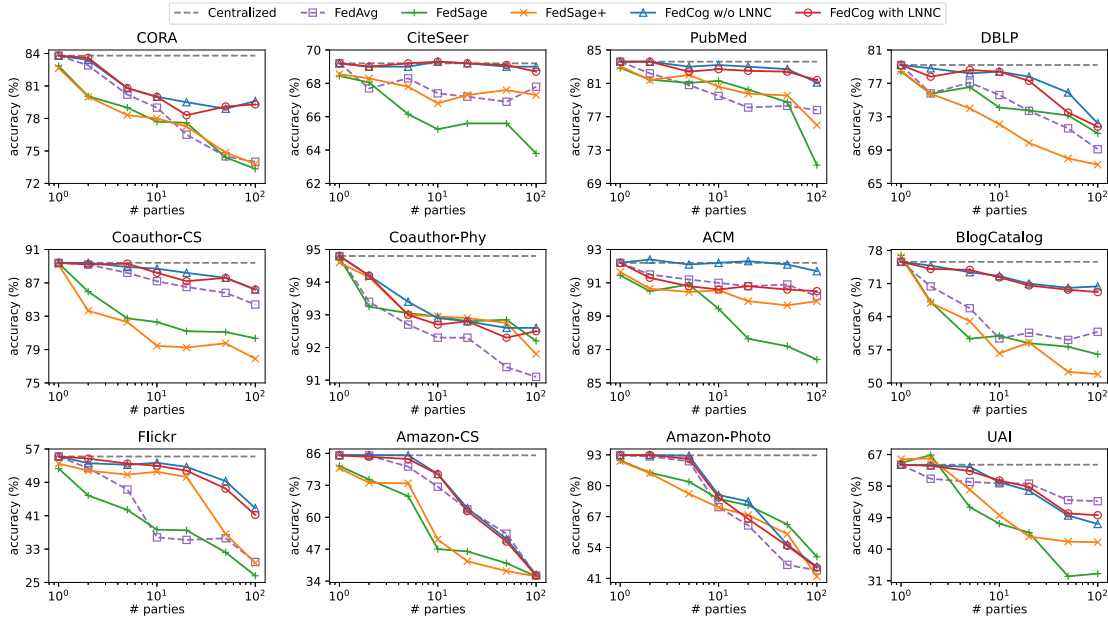

Fig. 4. Node classification accuracy when the global graphs are partitioned by METIS.

ACM datasets, FedCog's performance also declines along with party number growth, our method still outperforms FedAvg. Compared with FedSage and FedSage+ baselines, FedCog's accuracy is similar to FedSage+ on PubMed and ACM datasets. FedCog outperforms FedSage+ on the other 10 datasets.

When we compare FedCog with its LNNC-free version, we find that LNNC slightly weakens classification accuracy. In our experiments, the maximum accuracy decrease caused by LNNC occurs on the 100-party ACM dataset. LNNC causes an accuracy decrease by 2.0%, which is a tolerable cost to guarantee privacy security.

Fig. 4 shows node classification accuracy when the graphs are partitioned by METIS. In these experiments, our FedCog also outperforms FedAvg when the graphs are partitioned into various parties. On CORA, BlogCatalog, and Flickr datasets, the results of FedCog are 5.3%, 8.4%, and 11.3% better than FedAvg at 100 parties. FedAvg performs better than FedCog on the UAI dataset, but such results do not appear on the rest 11 datasets. When we compare FedCog with FedSage and FedSage+, we observe that our method has comparable results on some datasets, such as Coauthor-Phy, ACM, and Amazon-Photo. There are also some datasets where our method FedCog performs better,
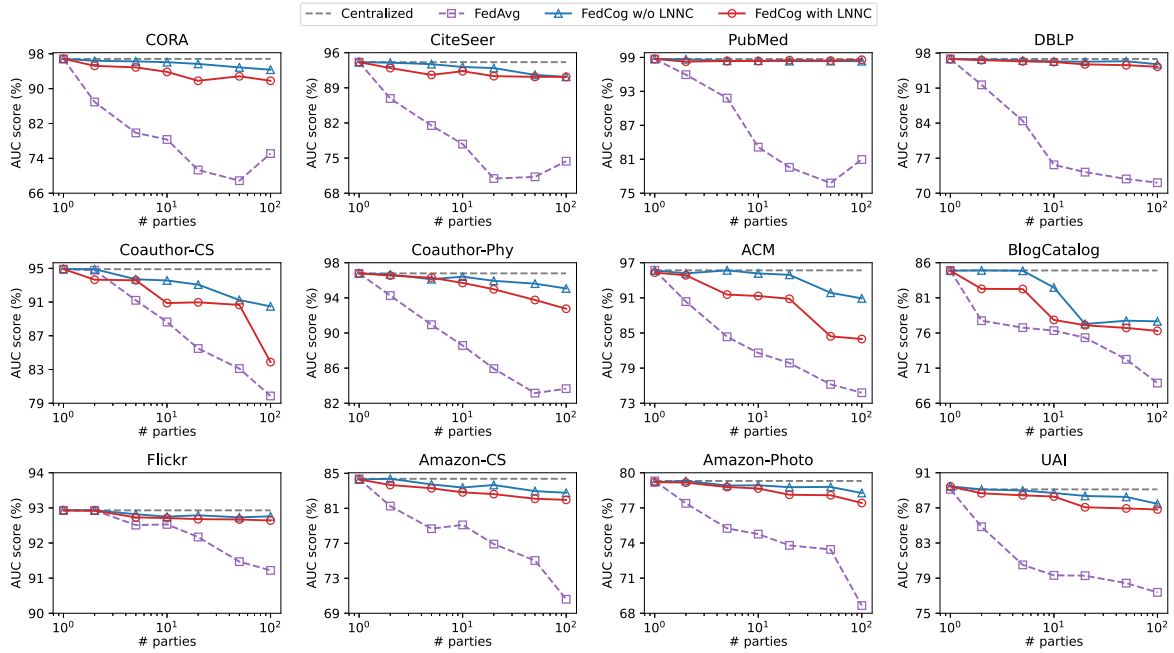
Fig. 5. AUC scores of link prediction tasks when the global graphs are partitioned by K-Means.

such as CORA and PubMed. LNNC has no significant influence on the classification results over most datasets except ACM. LNNC causes a 1.2% accuracy decrease on the ACM dataset, and consequently, FedCog does not outperform FedAvg on this dataset. However, on all of the other datasets, LNNC does not influence the performance significantly. Thus LNNC is an acceptable solution for privacy preservation.

In addition, we would like to point out that graph partitioning methods have a significant impact on classification accuracy even if the original dataset is the same. For example, if we change the partitioning method from METIS to K-Means for 100 parties, the accuracy changes of FedAvg are -7.1% on Cora and -5.5% on CiteSeer. These results show that when graphs are cut by different partitioning methods (or say when the distributions of local subgraphs are different), FedAvg will be significantly affected. However, the corresponding results of FedCog are +2.3% on Cora and -1.3% on CiteSeer respectively. The change is much smaller than FedAvg, which suggests that our FedCog method would perform better in the highly non-IID federated tasks of real-world scenarios.

C. Results of Link Prediction

We also conduct experiments on link prediction tasks. We use K-Means to partition the graphs into at most 100 parts as local graph data. For each dataset, we sample two sets of edges as the training and test sets successively. In both sets, the proportions of positive and negative samples are 1:1. In our experiment, the neural network model is an SGC model followed by a dot predictor. The 2-layer SGC outputs a 100-dim embedding for each node. Then the dot predictor takes a pair of node embeddings as its input and outputs a predictive score by computing the inner product of the two embedding vectors. We

compute AUC scores based on the predictive scores to evaluate each method's performance.

The results of link prediction are shown in Fig. 5. The finding is similar to node classification: the performance of FedAvg decreases along with the increase of parties, while the performance of our FedCog changes slightly. The results of FedCog are better than the baselines when the graphs are cut into a large number of parties. When the number of parties is 100, FedCog outperforms FedAvg by 0.168, 0.176, and 0.232 on CiteSeer, PubMed, and DBLP datasets respectively. These results show that our FedCog method is better than the FedAvg baseline in link prediction tasks. LNNC scheme causes performance degradation on some datasets, such as Coauthor-CS and ACM. The AUC scores decrease by 0.066 and 0.069 on Coauthor-CS and ACM datasets respectively. However, FedCog's results are still better than the baseline methods. Therefore, FedCog is an effective approach to improve link prediction results on coupled graphs.

D. Results of Efficiency

We test the computation and communication efficiency of FedCog on 4 large-scale datasets from different domains: DBLP, Coauthor-Phy, Flickr, and Amazon-CS. The settings of GNN models and FL training algorithms are the same as the node classification experiments in Section V-B. FedCog only changes the propagation procedure of SGC, so we simply compare the computation and communication costs during the propagation phase (parameter updates are excluded) for FedCog and all baselines.

● *Computation Efficiency*. Table III shows the run-time of FedCog, FedAvg, and the centralized method during SGC's propagation phase. For the federated methods (i.e., FedCog

TABLE III
RUN-TIMES ON 100 FL PARTIES

	FedCog (ms)	FedAvg (ms)	Centralized (ms)	Param. Update (ms/round)	$\frac{\text{FedCog}}{\text{Centralized}}$	$\frac{\text{FedCog}}{\text{Update}}$	Edge Density $(\times 10^{-4})$
DBLP	1470	522	2144	938	0.69	1.57	3.37
Coauthor-Phy	9746	2168	13183	1040	0.74	9.37	4.17
Flickr	4178	449	1996	878	2.09	4.76	83.57
Amazon-CS	1436	283	825	431	1.74	3.33	26.00

and FedAvg), we show the sum of all parties' computing time. FedAvg is faster than the centralized method because the graph is cut into small pieces and some edges are missing. Also as expected, FedCog takes more computation resources than FedAvg because FedCog introduces extra nodes and edges to the local graph. Comparing FedCog with the centralized method, we find that the sum of run-times over all FedCog parties is smaller than the centralized baselines DBLP and Coauthor-Phy datasets, but larger on Flickr and Amazon-CS. The difference between FedCog and the centralized method in efficiency is caused by two main factors: (1) Computing on graphs generally costs $O(|V|^2)$ time. Each party i of FedCog works on a small graph with $|V_i|$ nodes, while the centralized method directly computes on a large graph with $|V|$ nodes, so FedCog will benefit from the small local data. (2) FedCog adds extra nodes and edges to the local graphs, so the total number of nodes and edges is larger than the global graph in the centralized method, which slows down FedCog. Combining the two factors with the theoretical efficiency analysis in Section IV-C and Table I, the number of inter-edges is supposed to be the reason for different efficiency results on different datasets. The experimental results support our theory. We list the edge density of each dataset

$$\text{density} = \frac{2|E|}{|V|(|V|-1)}, \tag{22}$$

in Table III. DBLP and Coauthor-Phy have small edge density, thus the number of inter-edges between parties is also small. FedCog does not need to add too many nodes and edges to the local graphs and presents high efficiency. On the contrary, Flickr and Amazon-CS have much larger edge densities, so FedCog is slow. Notice that on the extremely dense graph Flickr, the total computation cost of FedCog is $2.09\times$ that of the centralized method and $0.02\times$ for a single party on average, respectively. Such computation cost is acceptable in practice. We also show the computation cost of federated training in Table III. In the worst case, the computing time of FedCog is approximately 9.37 rounds of model parameter updates on the Coauthor-Phy dataset. In our experiments, FedAvg generally needs 100~200 rounds to reach the best model performance, which means FedCog only consumes at most 10% of computing resources during the entire federated learning process.

● *Communication Efficiency*. Table IV shows the communication cost of FedCog. The FedCog-free or centralized method takes no communication resources during the propagation phase of SGC, so they are not listed in Table IV. We compare FedCog's communication cost with the global parameter updates during federated training. The worst results show that FedCog's communication cost is equivalent to 7.81 rounds of global updates

TABLE IV
COMMUNICATION COSTS ON 100 FL PARTIES

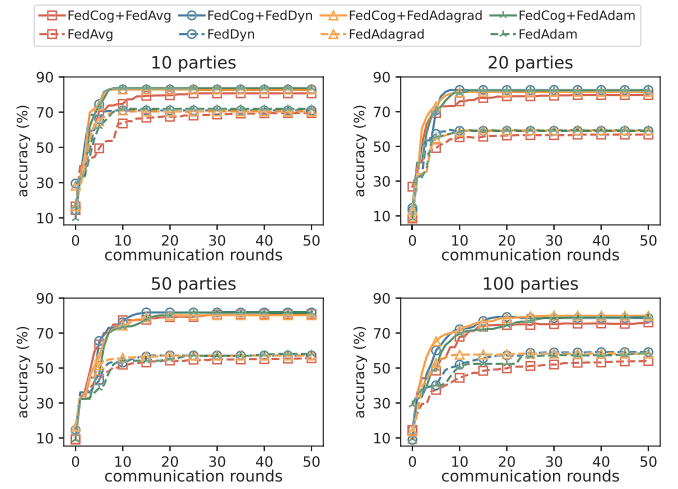|	FedCog (MB)	Global Update (MB/round)	$\frac{\text{FedCog}}{\text{Update}}$
DBLP	8.3	125.4	0.07
Coauthor-Phy	253.5	642.5	0.39
Flickr	4064.6	919.9	4.42
Amazon-CS	463.5	59.4	7.81

Fig. 6. Node classification accuracy on CORA dataset partitioned into $\{10, 20, 50, 100\}$ parts with K-Means. Solid lines show the results of FedCog, and dashed lines are for SGC without regard to coupled graphs.

on the Amazon-CS dataset, which is a small part of the total communication during federated learning.

E. Results With More Federated Training Algorithms

All model parameters are trained by FedAvg [1] (i.e., Algorithm 1) during experiments in Sections V-B, V-C, and V-D. In this section, we show the results of training the SGC model with FedDyn [14], FedAdagrad, and FedAdam [15], which are designed to improve FL convergence, especially for non-IID local data. Fig. 6 shows the node classification accuracy over the CORA dataset during the first 50 communication rounds of federated training. FedDyn, FedAdagrad, and FedAdam with FedCog (resp. without FedCog) provide better convergence rates and model performance than FedAvg with FedCog (resp. without FedCog). However, the improved FL training algorithms cannot fill the gaps between performance with and without FedCog. For example, the accuracy of FedAvg is 54.0% at the 50-th round in the 100-party scenario. The results of FedAdagrad and FedCog+FedAvg are 59.1% and 76.1%, respectively. Such results suggest that the coupled graph problem is not merely

caused by local data's non-IID distribution. The lack of topological information in graph federated learning cannot be offset by simply enhancing the performance of model parameter update algorithms. Methods for dealing with inter-party graph data (such as FedCog) are essential for graph federated learning.

VI. RELATED WORK

Federated Learning (FL) [1], [2] is a machine learning framework where several parties (or, clients) jointly train a model while preserving the data privacy of each party. There are three classes of FL based on the partition pattern of data, i.e., horizontal FL, vertical FL, and transfer FL. Horizontal FL is designed for the situation where each party shares the same attribute and label space, while the data samples are partitioned. Vertical FL is the opposite of horizontal FL and it considers that parties have the same samples but the attribute or label spaces are different. Transfer FL allows the parties to have few samples or attributes and uses the transfer learning methods to build a federated model. Recent works in the FL area concentrate on communication efficiency [30], [31], [32], [33], privacy preservation [34], [35], [36], [37], and robustness on not independent and identically distributed (Non-IID) data [14], [15], [29], [38].

Graph Neural Networks (GNN) [39], [40] are deep learning methods to solve the graph learning tasks such as node classification and link prediction. Graph Convolutional Network (GCN) [7] is a GNN model with graph convolution operators. From the topological perspective, GCN can be interpreted as feature propagation operations, i.e., each node aggregates the features of its neighbors. SGC [9] adopts straightforward low-pass filters to simplify GCN, which smooths the neighbor features using normalized adjacency matrices. GNN models, especially GCN and its variants, have been successfully applied to recommendation systems [41], [42], [43], social networks mining [44], [45], natural language processing [46], [47], [48], and biochemistry [49].

The most related line of research to our work is the *FL over Graphs*. It is non-trivial to perform FL over graphs because graphs are non-Euclidean data and particularly they are given in a variety of forms for real-world applications. To address this challenge, recently, several methods have been proposed and they can be summarized as follows.

● The *horizontal graph FL* mainly has three classes, i.e., *graph level* FL, *subgraph level* FL, and *node level* FL. For graph level FL, each party has a set of graphs. Take FedGraphNN [5] as an instance. FedGraphNN exploits small graphs (e.g., chemical and biological molecular graphs) distributed on multiple parties and uses FL to achieve high performance of graph classification and regression. Subgraph level FL considers each party as a collection of nodes and edges from a graph. For example, each mobile user holds a subgraph consisting of edges to its friends or its installed mobile applications. Mobile users can jointly perform FL on their personal data to build an effective system for recommending mobile applications. To the best of our knowledge, FedGNN [50] is the first subgraph level FL. GraphFL [51] provides a generalized method for subgraph level FL, and ASFGNN [4] addresses the problem of non-IID node

features in subgraph level FL. FedGraphNN [5] also discusses the settings of partitioned subgraphs and nodes, but it does not involve the data coupling among parties, and only fits some specific cases (i.e., it is mainly designed for bipartite graphs). For node level FL, each party processes only one node and all of its connections in the graph of interest. The practical cases include online social networks and spatial-temporal trajectory graphs [52].

● The *vertical graph FL* studies different node data partition scenarios. VFGNN [53] focuses on the scenario where node features are vertically separated into different parties. SGNN [3] performs the vertical graph FL when each party has all nodes in the graph but partial content, structural and label information. FedGL [54] further considers the situation where the nodes on parties are not completely overlapped.

In the previous graph FL researches, the partition of node features and edge features are studied. Horizontal graph FL regards the graph data on parties as individual graphs or topologically independent subgraphs. Vertical graph FL focuses on the division of attribute space, where the nodes and edges are shared among parties. Therefore, a majority of previous works on graph FL do not consider the partition of graph topological structures. Topological structures are essential components for graphs and play important roles in graph learning. In our work, we consider not only the partition of node samples, but also the partition of the graph's topological structures. We propose *coupled graphs* to represent the distributed graph data. A recent work, FedGCN [55], also focuses on a similar problem as our work. It addresses the cross-party edge problem in graph federated learning. In FedGCN, each federated learning party receives node features from other parties and stores them on its local device. Thus, the locally stored graph data from other parties can participate in local computation. However, it faces some challenges. The direct transmission of node features lacks privacy guarantees. There are risks of sensitive data leakage, e.g., edges and node features. In addition, the communication and storage costs of FedGCN are not minor, especially influenced by the depth of the GNN model. The expensive cost of multi-layer models is also a problem of FedGCN. In our work, we propose a graph decoupling method to deal with the edges between federated parties. Our method also provides a guarantee for the privacy of node features and has practical communication costs.

VII. CONCLUSION

In this work, we formally define the coupled graph learning problem in the setting of Federated Learning, and propose a framework FedCog to address it. To improve the performance of federated models, FedCog enables the federated parties to share the coupled information in a privacy-preserving way. The experiments demonstrate that our FedCog increases the performance of federated graph learning by up to 14.7% accuracy in node classification tasks compared with existing baselines, and up to 0.232 AUC score in link prediction tasks. In the future, we will work towards communication efficiency and privacy preservation strategies for more GNN models.

REFERENCES

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

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

[3] G. Mei, Z. Guo, S. Liu, and L. Pan, "SGNN: A graph neural network based federated learning approach by hiding structure," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 2560–2568.

[4] L. Zheng, J. Zhou, C. Chen, B. Wu, L. Wang, and B. Zhang, "ASFGNN: Automated separated-federated graph neural network," *Peer Peer Netw. Appl.*, vol. 14, no. 3, pp. 1692–1704, 2021.

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

[6] K. Zhang, C. Yang, X. Li, L. Sun, and S.-M. Yiu, "Subgraph federated learning with missing neighbor generation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 6671–6682.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[8] A. Arenas, A. Fernandez, and S. Gomez, "Analysis of the structure of complex networks at different resolution levels," *New J. Phys.*, vol. 10, 2008, Art. no. 053039.

[9] F. Wu, A. H. Souza Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.

[10] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. Int. Conf. Learn. Representations*, 2019.

[11] M. Chen et al., "Scalable graph neural networks via bidirectional propagation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 14556–14566.

[12] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *Proc. Int. Conf. Learn. Representations*, 2021.

[13] P. Li, I. Chien, and O. Milenkovic, "Optimizing generalized pagerank methods for seed-expansion community detection," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11705–11716.

[14] D. A. Emre Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. Int. Conf. Learn. Representations*, 2021.

[15] S. J. Reddi et al., "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Representations*, 2021.

[16] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

[18] K. A. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[19] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning," 2020, *arXiv:2009.11248*.

[20] Q. Lu and L. Getoor, "Link-based classification," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 496–503.

[21] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–106, 2008.

[22] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1895–1901.

[23] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, *arXiv:1811.05868*.

[24] X. Wang et al., "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032.

[25] J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proc. Conf. Inf. Knowl. Manage.*, 2015.

[26] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 43–52.

[27] W. Wang, X. Liu, P. Jiao, X. Chen, and D. Jin, "A unified weakly supervised framework for community detection and semantic matching," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2018, pp. 218–230.

[28] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, 1998.

[29] T.-M. Harry Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 76–92.

[30] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statst.*, 2020, pp. 2021–2031.

[31] D. Rothchild et al., "FetchSGD: Communication-efficient federated learning with sketching," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8253–8265.

[32] Y. Zhou, Q. Ye, and J. Lv, "Communication-efficient federated learning with compensated overlap-FedAvg," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 192–205, Jan. 2022.

[33] M. Tang et al., "FedCor: Correlation-based active client selection strategy for heterogeneous federated learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10092–10101.

[34] Y. Wu et al., "FedCG: Leverage conditional GAN for protecting privacy and maintaining competitive performance in federated learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2334–2340.

[35] X. Zhang, X. Chen, M. Hong, S. Wu, and J. Yi, "Understanding clipping for federated learning: Convergence and client-level differential privacy," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 26048–26067.

[36] W.-N. Chen, A. Özgür, and P. Kairouz, "The poisson binomial mechanism for unbiased federated learning with secure aggregation," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 3490–3506.

[37] X. Gong et al., "Preserving privacy in federated learning with ensemble cross-domain knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 11891–11899.

[38] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-iid federated learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10164–10173.

[39] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2005, pp. 729–734.

[40] F. Scarselli, M. Gori, Ah C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

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

[42] S. Wu, Y. Zhang, C. Gao, K. Bian, and B. Cui, "GARG: Anonymous recommendation of point-of-interest in mobile networks by graph convolution network," *Data Sci. Eng.*, vol. 5, no. 4, pp. 433–447, 2020.

[43] Y. Zheng, C. Gao, L. Chen, D. Jin, and Y. Li, "DGCN: Diversified recommendation with graph convolutional networks," in *Proc. World Wide Web Conf.*, 2021, pp. 401–412.

[44] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with markov random field reasoning for social spammer detection," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 1054–1061.

[45] W. Lin, Z. Gao, and B. Li, "Guardian: Evaluating trust in online social networks with graph convolutional networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 914–923.

[46] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1506–1515.

[47] C. Li and D. Goldwasser, "Encoding social information with graph convolutional networks forpolitical perspective detection in news media," in *Proc. Annu. Meeting Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 2594–2604.

[48] Q. Dong, S. Niu, T. Yuan, and Y. Li, "Disentangled graph recurrent network for document ranking," *Data Sci. Eng.*, vol. 7, no. 1, pp. 30–43, 2022.

[49] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6533–6542.

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

[51] B. Wang, A. Li, H. Li, and Y. Chen, "GraphFL: A federated learning framework for semi-supervised node classification on graphs," 2020, *arXiv:2012.04187.*

[52] C. Meng, S. Rambhatla, and Y. Liu, "Cross-node federated graph neural network for spatio-temporal data modeling," 2021, *arXiv:2106.0522.*

[53] J. Zhou et al., "Vertically federated graph neural network for privacy-preserving node classification," 2020, *arXiv:2005.11903.*

[54] C. Chen, W. Hu, Z. Xu, and Z. Zheng, "FedGL: Federated graph learning framework with global self-supervision," 2021, *arXiv:2105.03170.*

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

Runze Lei received the BS degree in automation engineering from Xi'an Jiaotong University, Xi'an, China, in 2020. He is currently working toward the graduate degree with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University. His research interests include federated learning and stream data mining.

Pinghui Wang (Senior Member, IEEE) is currently a professor with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China, and also with the Shenzhen Research Institute, Xi'an Jiaotong University, Shenzhen, China. His research interests include internet traffic measurement and modeling, traffic classification, abnormal detection, and online social network measurement

Junzhou Zhao received BS and PhD degrees in control science and engineering from Xi'an Jiaotong University, in 2008 and 2015, respectively. He is currently an associate professor with the School of Cyber Science and Engineering, Xi'an Jiaotong University. His research interests include graph data mining and streaming data processing.

Lin Lan received the BS degree in automation engineering from Xi'an Jiaotong University, Xi'an, China, in 2016. He is currently working toward the PhD degree with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University. His research interests include network analysis, network representation learning, and deep learning and its applications.

Jing Tao received the BS and MS degrees in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2006, respectively. He is currently a teacher with Xi'an Jiaotong University and an on-the-job PhD candidate with the Systems Engineering Institute. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and botnet.

Chao Deng received the MS and PhD degrees from the Harbin Institute of Technology, Harbin, China, in 2003 and 2009, respectively. He is currently a deputy general Manager with AI Center of China Mobile Research Institute. His research interests include artificial intelligence for ICT operations.

Junlan Feng (Fellow, IEEE) received the PhD degree in acoustics from the Chinese Academy of Sciences, in 2001. She was a principal technical staff member with AT&T Labs Research. She is currently the chief scientist with China Mobile Research. Her research interests include several technical areas including Web mining, natural language understanding, machine learning, and speech processing.

Xidian Wang received BS and MS degrees from Xidian University, Xi'an, China, in 2002 and 2005, respectively. He is currently the deputy chief engineer with China Mobile Group Design Institute and the chief engineer with R&D Center for Network Planning and Optimization. His research interests include artificial intelligence for communication networks and intelligent network systems.

Xiaohong Guan (Fellow, IEEE) received the PhD degree in electrical engineering from the University of Connecticut, Storrs, in 1993. Since 1995, he has been with the Department of Automation, Tsinghua National Laboratory for Information Science and Technology, and the Center for Intelligent and Networked Systems, Tsinghua University. He is currently with the MOE Key Laboratory for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China, where he is also the dean of the Faculty of Electronic and Information Engineering. He is an academician of the Chinese Academy of Sciences.