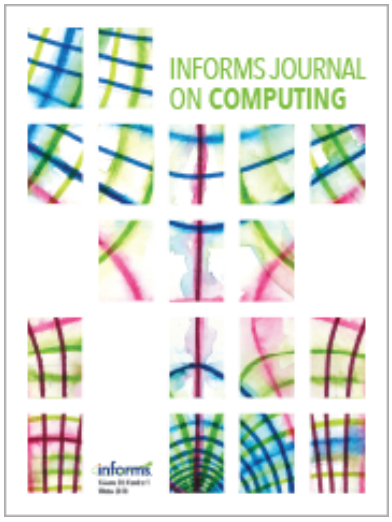


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Toward Graph Data Collaboration in a Data-Sharing-Free Manner: A Novel Privacy-Preserving Graph Pretraining Model

Jiarong Xu; , Jiaan Wang; , Zenan Zhou, Tian Lu

To cite this article:
Jiarong Xu; , Jiaan Wang; , Zenan Zhou, Tian Lu (2025) Toward Graph Data Collaboration in a Data-Sharing-Free Manner: A Novel Privacy-Preserving Graph Pretraining Model. INFORMS Journal on Computing

Published online in Articles in Advance 04 Jun 2025

. https://doi.org/10.1287/ijoc.2023.0115

https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–16
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Toward Graph Data Collaboration in a Data-Sharing-Free Manner: A Novel Privacy-Preserving Graph Pretraining Model

Jiarong Xu,[a,*] Jiaan Wang,[a] Zenan Zhou,[b] Tian Lu[b]

[a] School of Management, Fudan University, Shanghai 200433, China; [b] W. P. Carey School of Business, Arizona State University, Tempe, Arizona 85287

*Corresponding author

Contact: jiarongxu@fudan.edu.cn, https://orcid.org/0000-0003-2973-1889 (JX); jiaanwang.is@gmail.com, https://orcid.org/0000-0002-2587-7648 (JW); zzhou119@asu.edu, https://orcid.org/0000-0001-7654-9145 (ZZ); lutian@asu.edu, https://orcid.org/0000-0003-3730-1897 (TL)

Received: April 8, 2023
Revised: December 14, 2023; September 14, 2024; December 31, 2024; April 2, 2025; April 17, 2025
Accepted: April 27, 2025
Published Online in Articles in Advance: June 4, 2025

https://doi.org/10.1287/ijoc.2023.0115

Abstract. Graph data, prevalent in various domains such as telecommunication, supply chain, and social networks, holds significant potential for business, operations, and social administration. Collaborating on graph data across institutions or users can further unleash its value, making it a highly sought-after practice. However, such collaboration poses risks to information privacy and commercial confidentiality. In response, we introduce an innovative new model-sharing strategy for graph data collaboration. Here, a data owner pretrains a graph neural network (GNN) model on their private graph data and then provides model users with query access to this model. The pretrained GNN acts as an intermediary, encapsulating knowledge from the private data without exposing it directly. Two fundamental principles are essential for such a pretrained GNN model: model generalizability and privacy preservation. However, current efforts often fail to achieve both concurrently. To tackle this challenge and promote an open yet secure graph data collaboration framework, we propose a novel privacy-preserving operator. This operator integrates smoothly with graph data augmentation and graph contrastive learning, allowing the pretraining of a GNN that effectively eliminates private links at high risk of exposure while maintaining generalizability. Additionally, to improve model generalizability, we introduce a new method called generalizability learning to enhance the model's adaptability when deployed on unseen data of model user. This approach is designed to simulate diverse environments and develop representations that remain invariant across these varied environments. Extensive experiments suggest that our model surpasses existing state-of-the-art approaches in striking an effective balance between privacy preservation and generalizability.

History: Accepted by Ram Ramesh, Area Editor for Data Science and Machine Learning.
Funding: This work was supported (to J. Xu) by the National Natural Science Foundation of China [Grants 62206056, 72271059, and 72442011] and the CIPSC-SMP-Zhipu Large Model Cross-Disciplinary Fund.
Supplemental Material: The software that supports the findings of this study is available within the paper and its Supplemental Information (https://pubsonline.informs.org/doi/suppl/10.1287/ijoc.2023.0115) as well as from the IJOC GitHub software repository (https://github.com/INFORMSJoC/2023.0115). The complete IJOC Software and Data Repository is available at https://informsjoc.github.io/.

Keywords: data collaboration • model sharing • pretraining model • privacy preservation

1. Introduction

Graph data, also known as graph-structured data or network data, are ubiquitous data structures used for modeling the interactions (i.e., edges) between objects (i.e., nodes), providing rich topological information and generic connectivity patterns (Barabási 2013). Examples of graph data include social, telecommunication, and supply chain networks (Zhou et al. 2020). Recent computational technology advancements (e.g., cloud computing, deep learning) have facilitated the use of graph data in decision support and production process optimization.[1] As a result, the global market size of the graph database is rapidly growing, and it is expected to reach USD 4.5 billion by 2026.[2]

The escalating demand for graph data are particularly notable. Consider a scenario where Foursquare is supposed to collaborate with Twitter.[3] Twitter is one of the largest social media platforms, with over 500 million monthly active users in 2024, constituting an extensive private social network. In contrast, Foursquare, famous as a location-sharing-featured social media platform, maintains a much smaller monthly active user base of 55 million. Foursquare may seek to enhance downstream tasks such as friend recommendation, influencer

identification, and fake account detection by leveraging Twitter's rich network information as valuable supplementary knowledge.

However, it is highly improbable for any single entity to practically possess and sustain diverse graph data resources. As a result, there is a growing demand for sharing or collaborating on graph data with partners and external institutions to augment in-house data. Typically, companies actively seek collaborations with external data partners who share a portion of their user base (or user data commonality) but possess larger and higher-quality data sets to enhance their operational task performance (Bauer et al. 2020), as illustrated by the Twitter–Foursquare collaboration. Nevertheless, because of concerns related to commercial confidentiality and information privacy issues (e.g., refer to General Data Protection Regulation and California Consumer Privacy Act), sharing graph data directly is typically challenging or not permitted. Consequently, although graph data represent a concealed gold mine in the big data era, the majority of valuable graph data, in reality, remain inaccessible.

To navigate challenges in graph data collaboration, a viable solution involves facilitating data collaboration in a data-sharing-free manner, where knowledge is shared instead of raw data. Recent advances in artificial intelligence (AI) techniques, such as pretraining strategy (Han et al. 2021), have made it feasible for data owners to train an AI model on their local data and share only the pretrained AI models (also known as models or pretraining models) with their partners. This provides model users flexibility to obtain knowledge by querying the pretrained AI model with their own data. We conceptualize this data-sharing-free method as *model sharing*, which enables the goal of information sharing while minimizing the risk of privacy breaches.

In this study, we specifically address two primary objectives for an ideal graph model in the context of model sharing: *model generalizability* and *privacy preservation*. We believe these dual imperatives underpin the fundamental principles of a collaborative model. Specifically, model generalizability requires the models for collaboration be capable of learning universal and transferable knowledge from the data owner's training graph, ensuring that the knowledge embedded in the pretrained models can be easily adapted to unseen graphs of model users. This objective necessitates the practical usefulness of models used for information sharing, as distorted information sharing renders data collaboration meaningless. Moreover, privacy preservation requires that adversaries accessing the pretrained model cannot reveal the private information of the training graph (i.e., defend against information inference attacks). This specifies that qualified models must ensure data privacy to adhere to the information interests of data owners and comply with regulations.

However, it is worth noting that it is always technically challenging to balance the trade-off between privacy preservation and model generalizability, because implementing strong privacy preservation approaches often hinders the model's ability to accurately discern and learn from the transferable pattern of the data (Li and Li 2009). Indeed, existing research mainly falls into two lines, yet each has inherent limitations for the desired graph model for model sharing. One line introduces some strategies of pretraining graph neural networks (GNNs) to ensure model generalizability (Qiu et al. 2020, Xu et al. 2024), but many pretrained GNN models pose risks of privacy leakage. The other line focuses on the privacy-preserving graph models, although most of them cannot be generalized to unseen data with different tasks (Liao et al. 2021, Wang et al. 2021, Hu et al. 2022). Although some straightforward solutions can combine these two branches of research, such as adopting differential privacy on graph data before pretraining GNNs (Wu et al. 2022, Sajadmanesh et al. 2023), these approaches often compromise important transferable structures, resulting in an unfavorable privacy–generalizability trade-off (Abadi et al. 2016). Therefore, this study aims to propose a model that *empowers the graph pretraining models with both generalizability and privacy-preservation capabilities*.

Our study yields multifold contributions. First, we are among the first to introduce a model-sharing strategy for graph data collaboration, which is an area of great need yet largely overlooked. In contrast to conventional strategies of direct graph data sharing or releasing embeddings, we are pioneers in releasing a privacy-preserving pretrained GNN model. This innovative strategy ensures both privacy preservation and model generalizability when utilized with model users' own data to acquire additional knowledge. Second, we design a novel privacy-preserving operator that integrates seamlessly with graph data augmentation, resulting in a privacy-preserving graph data augmentation. This approach addresses the privacy concern by removing private links with a high possibility of privacy leakage while guaranteeing model generalizability. In contrast, existing graph data augmentation methods often fall short in effectively addressing privacy concerns in data collaboration. Third, we introduce a novel method of generalizability learning designed to enhance model generalizability. This method simulates a variety of environments and learns representations invariant across these different environments. Such an approach ensures that the pretrained GNN model, developed by the data owner, can be easily adapted to downstream tasks on unseen graphs. In contrast, most existing GNNs are designed with the assumption that the training and test graphs are drawn from the same distribution, which often leads to poor performance on unseen graphs (Cao et al. 2023, Huang et al. 2024). Finally, to the best of our knowledge, we are

among the first to empirically explore the graph properties and graph substructures for which privacy or generalizability matters through the analysis of observed network data. We consider both graph-level and node-level properties, suggesting that graphs with specific properties, such as low average degree and high network constraint, enhance privacy protection in data sharing. However, complex substructures, like 4-clique, compromise privacy preservation while enhancing generalizability. These findings inspire scholars and practitioners (e.g., data owners) to figure out more tailored information sharing strategies based on the graph structures.

2. Literature Review

This research is highly related to three streams of literature: pretraining GNNs, privacy attacks and privacy-preserving approaches, and graph data augmentation.

2.1. Pretraining GNNs

Pretraining GNNs has shown great effectiveness for learning graph representations without costly labels, and can enhance the generalizability of GNNs. In the pretraining stage, these models learn universal knowledge from large-scale graph data using self-supervised or unsupervised techniques. This knowledge is then transferred to specific downstream tasks during the fine-tuning stage. Initially, GNNs are pretrained using self-supervised or unsupervised methods that rely on neighborhood similarity assumptions, as proposed by Grover and Leskovec (2016) and Hamilton et al. (2017). These methods, however, struggle with generalizability to unseen graphs because of their restrictive assumptions. To overcome these limitations, researchers have developed GNN pretraining strategies using graph generative models (Kipf and Welling 2016, Hu et al. 2020a) and contrastive learning methods (Hu et al. 2020b), which typically involve pretraining on graphs from one domain and fine-tuning on another data set from a similar or the same domain. Furthermore, to advance the generalizability across different domains, attempts have been made at cross-domain graph pretraining (Qiu et al. 2020, Cao et al. 2023, Xu et al. 2024). They focus on learning transferable subgraph structural patterns through contrastive learning. Such models can seamlessly transfer between different domains, and thus are employed as the backbone model in our study. Despite advancements in pretraining GNNs for enhanced generalizability, prior efforts have largely overlooked the privacy risks associated with pretrained models.

2.2. Privacy Attacks and Privacy-Preserving Approaches

Model-based privacy attacks, which target the provided machine learning (ML) model and focus on extracting information about either the training data or the model itself, are categorized into four primary types: membership inference, reconstruction, property inference, and model extraction (Rigaki and Garcia 2023). Membership inference attacks determine whether an input training sample was used in the training set. Reconstruction attacks (also known as attribute inference or model inversion) aim to recover sensitive features in the training data or the full training data sample (Rigaki and Garcia 2023, Zhou et al. 2023). When the training data are graph data (e.g., Zhou et al. 2023), graph reconstruction attacks are introduced to reconstruct the adjacency of nodes from the provided graph-based ML model. Property inference attacks are designed to extract undisclosed properties of the training data set that are not explicitly encoded as features or directly related to the learning task (Rigaki and Garcia 2023). Last, model extraction attacks involve inferring the structure and parameters of the provided ML model (Orekondy et al. 2019). The privacy attack discussed in our paper is a kind of reconstruction attack that attempts to reconstruct private links, from the pretrained GNN model.

The growing concern for privacy issues has led to a focus on developing privacy-preserving techniques within the field. Considering the types of graph data collaboration, the main privacy-preserving methods include collaboration through network embeddings, data publishing, and federated learning. Specifically, the first type of privacy-preserving network embedding releases node embeddings generated by privacy-preserving network embeddings (e.g., Wang et al. 2021, Hu et al. 2022, Han et al. 2024). The data publishing approaches directly publish faithfully generated data rather than the actual private data (e.g., Arora and Upadhyay 2019, Wu et al. 2022). The third type is federated learning, wherein a graph model is constructed across decentralized clients without data exchange (e.g., Xie et al. 2021). This presents a fundamentally different perspective for achieving free data sharing, which necessitates a trusted server keeping users' private information. Table A.1 (see Online Appendix A in the supplemental materials) outlines the comparison of these approaches with ours. Unlike privacy-preserving network embedding methods that release node embeddings, our novel approach releases a privacy-preserving pretrained GNN model. This allows model users to query with their own data to obtain knowledge. Additionally, the released model demonstrates the ability to generalize to model users' unseen graphs. Unlike data publishing that requires direct data exchange, often compromising the balance between privacy and utility

(Li and Li 2009), our approach avoids data sharing and enhances this balance. Additionally, unlike federated learning, our model does not require a trusted third party.

2.3. Graph Data Augmentation

The primary objective of graph data augmentation is to find a transformation function that converts a given graph into augmented views (or augmented graphs). These augmented views enrich the original graph by introducing variations that have the potential to help the model generalize better across different graphs (You et al. 2020). Graph data augmentations fall into two main categories: structure oriented and feature oriented. *Structure-oriented augmentations* alter the graph's adjacency matrix, such as typical methods like edge perturbation, subgraph sampling, node dropping, etc. Edge perturbation involves randomly adding or removing edges to perturb the graph structure (You et al. 2020), whereas subgraph sampling typically generates a connected subgraph induced from sampled nodes (Qiu et al. 2020, Xu et al. 2024). Node dropping, on the other hand, aims to drop or add a set of nodes and a set of edges from the input graph (You et al. 2020). Another line of structure-oriented augmentation approaches modifies graph structure while incorporating domain knowledge (Rong et al. 2020); however, this often confines their applicability to specific domains. *Feature-oriented augmentations* modify node features through techniques such as adding noise (Feng et al. 2019) or masking features (You et al. 2020). Although multiple graph data augmentations have been introduced, few effectively address privacy concerns. This study introduces a privacy-preserving graph data augmentation method that removes private links with a high possibility of privacy leakage while ensuring generalizability in the augmented view.

3. Problem Formulation

The model sharing for graph data is a kind of data collaboration that facilitates the use of closed graph data without sharing them. This strategy involves two parties: the data owner and model user. Each party has its own graph data $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}})$ (with the node and edge sets V_{train} and E_{train}) and $G_{\text{down}} = (V_{\text{down}}, E_{\text{down}})$ (with the node and edge sets V_{down} and E_{down}), respectively. Notably, the data owner, typically with a more extensive or exclusive data set, gives the model user an advantage by utilizing the data owner's pretrained GNN model. This allows the model user to gain insights that would otherwise be unattainable because of their own data limitations. For instance, small and medium-sized enterprises often struggle with inadequate and low-quality data for developing ML models (Bauer et al. 2020), leading them to adopt pretrained models from larger enterprises. Additionally, these two parties often share a subset of nodes, such as an overlap in user bases. For example, 90% of Twitter users also engage with Facebook, and 50% of Instagram users are active on Twitter (Smith and Anderson 2018). This commonality facilitates the extrapolation of insights from one company's data to enhance downstream performance on another, providing a solid foundation for collaboration. Specifically, the data owner pretrains a GNN model $e_\theta : G \to Z$ (e for short) with parameters θ on G_{train}, mapping a graph G to a D-dimensional representation $Z \in \mathbb{R}^D$, and then shares query access to the model with the model user. This allows the model user to query the model with their own data, gaining insights without directly accessing the private data. Now we formally define the model-sharing strategy for graph data collaboration (MSS4Graph) problem.

Definition 1 (MSS4Graph Problem). Consider a data owner, company $\mathcal{A}$, which privately owns a training graph G_{train}. This company collaborates with a company $\mathcal{B}$ (i.e., model user), which owns its downstream graph G_{down}. It is assumed that G_{train} and G_{down} share a subset of nodes or have the same node set. The goal of company $\mathcal{A}$ is to pretrain a GNN model e_θ on G_{train}. Then, company $\mathcal{A}$ shares the query access to e with company $\mathcal{B}$, such that company $\mathcal{B}$ can query the model with their own data G_{down} to obtain representations of G_{down} for downstream task. The model e is designed to concurrently achieve the following two principles:

i. *Privacy-preserving principle*: The adversary cannot precisely infer the private information from querying e_θ with its own data G_{down}. This ensures the privacy of data owner's graph data.

ii. *Generalizability principle*: The pretrained model is supposed to be transferable to company $\mathcal{B}$'s downstream graph G_{down}, though G_{down} is unseen to company $\mathcal{A}$, thereby enabling company $\mathcal{B}$ to effectively perform downstream tasks on G_{down}. That is, the pretrained model can perform well on new, previously unseen data G_{down}, not just on its training data G_{train}.

In the MSS4Graph problem, we follow prior studies, for example, Han et al. (2024) and Wang et al. (2021), to concentrate our efforts on protecting private links. We define the link existence between all node pairs in the data owner's training graph as the private information (or private links). The adversary performs a model-based private link reconstruction attack as follows.

Definition 2 (Model-Based Private Link Reconstruction Attack). Given the black-box query access to a pretrained GNN model e, trained on a private training graph $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}})$, the adversary aims to reconstruct the edge set of the training graph (i.e., E_{train}) by querying the pretrained GNN model e using the downstream graph $G_{\text{down}} = (V_{\text{down}}, E_{\text{down}})$.

4. Proposed Method

In this section, we propose a novel method for the data owner to train a privacy-preserving pretrained GNN model, aimed at addressing the MSS4Graph problem outlined in Section 3. Our method is designed to mitigate privacy leakage risks while ensuring model generalizability. Before delving into our method details, we outline the framework of our method in Section 4.1. Then, Section 4.3 introduces the proposed privacy-preserving operator to fulfill the privacy-preserving principle, and Section 4.4 presents the generalizability learning to fulfill the generalizability principle. The overall algorithm and complexity analysis can be found in Online Appendix B.

4.1. Framework Overview

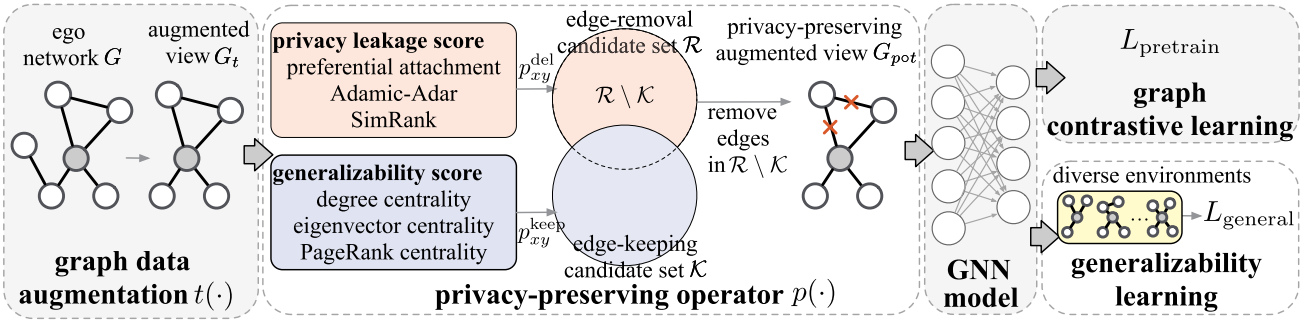
The primary objective of our model is to achieve the privacy-preserving and generalizability principles simultaneously. Figure 1 presents the overall framework of our model, consisting of five major modules: graph data augmentation, privacy-preserving operator, GNN model, graph contrastive learning, and generalizability learning. The graph data augmentation, GNN model, and contrastive learning modules constitute the basic graph pretraining framework. The privacy-preserving operator and generalizability learning are dedicated to fulfilling the privacy-preserving and generalizability principles, respectively. Below, we detail the specific responsibilities of these modules.

4.1.1. Graph Data Augmentation, GNN Model, and Graph Contrastive Learning Modules. These three modules form the core of the basic graph pretraining framework (Qiu et al. 2020, Xu et al. 2024). Specifically, the graph data augmentation takes the training graph as input and generates augmented views. Given these augmented views, we adopt graph contrastive learning as the graph pretraining task, aiming to learn transferable structural representations. Its goal is to distinguish nodes based on their local structures. Drawing on the principles of contrastive learning, graph contrastive learning maximizes the consistency between two augmented views of the same ego network of the same node compared with those of different ego networks (Qiu et al. 2020, Xu et al. 2024). Finally, we use a GNN model, specifically, the graph isomorphism network (GIN; Xu et al. 2019), to obtain node representations by mapping its local structure to a latent representation.

4.1.2. Privacy-Preserving Operator Module. The privacy-preserving operator is designed to satisfy the privacy-preserving principle. Current graph data augmentation functions, as referred to in Equation (1), predominantly perform random perturbations on the graph (Qiu et al. 2020, You et al. 2020), or based on domain knowledge (Rong et al. 2020). However, these graph data augmentations often do not adequately protect the privacy of training graphs. To address this, we introduce a privacy-preserving operator, which is designed to achieve the privacy-preserving principle by removing private links with a high possibility of leakage from the augmented view, while ensuring that the retained links can guarantee model generalizability. This results in a privacy-preserving augmented view G_{pot}, which is then fed into the GNN model for representation learning.

4.1.3. Generalizability Learning Module. This module is introduced to fulfill the generalizability principle. As a model-sharing strategy, the pretrained GNN model should have generalizability. Specifically, the pretrained

Figure 1. (Color online) The Overall Framework of Our Proposed Model for the MSS4Graph Problem



GNN model can be easily adapted to downstream tasks on model users' unseen graphs, that is, the downstream graph. However, most GNNs are designed under the assumption that the training and test graphs come from the same distribution, and thus struggle with out-of-distribution generalization (Cao et al. 2023, Huang et al. 2024). These models cannot effectively share knowledge to model users with unseen data. To this end, we are inspired by the invariant learning (Li et al. 2022) and posit that the learned representation is invariant across different environments, thereby ensuring generalizability. Therefore, we introduce a novel generalizability learning method to produce a GNN model whose output representations can survive a large class of distribution shifts implied by different environments.

In summary, these five modules collaboratively address the MSS4Graph problem as follows. First, a training graph provided by the model user is processed through graph data augmentation to generate augmented views. To mitigate the privacy leakage risk in these views, they are then passed through the privacy-preserving operator, resulting in the privacy-preserving augmented views. Subsequently, these privacy-preserving augmented views are fed into the GNN model and obtain graph representations. The GNN model is trained with graph contrastive learning to learn structural representations across graphs and generalizability learning to ensure the model generalizability.

4.2. Basic Graph Pretraining Framework

A basic graph pretraining framework, without specifically addressing the privacy-preserving and generalizability principles, is commonly a combination of graph data augmentation, a GNN model, and contrastive learning (Qiu et al. 2020, Xu et al. 2024).

4.2.1. Graph Data Augmentation. Given an ego network[4] of a node $G \in \mathcal{G}$, the graph data augmentation function $t(\cdot)$ would generate new views of G (i.e., G's augmented views) as

$$G_t = t(G), t \sim T, \tag{1}$$

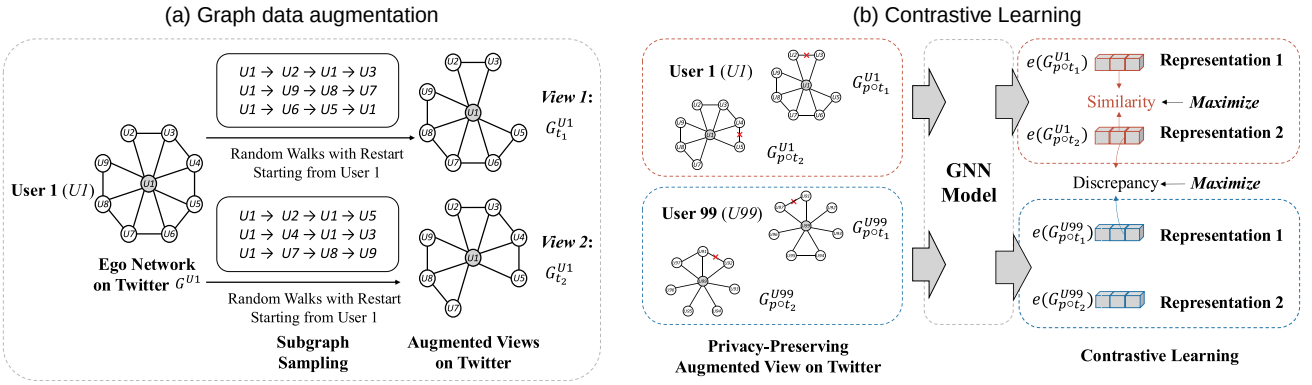
where T is the set of graph data augmentation functions, and each function t sampled from T maps G to the new augmented view G_t. Graph data augmentation can generate different views of the same ego network, facilitating the graph contrastive learning process.

To ensure the GNN model trained on augmented views from the training graph is transferable to the model user's downstream graph, it is crucial that the graph data augmentation $t(\cdot)$ preserves transferable patterns, that is, structural patterns commonly present in ego networks. This definition underlines that nodes with similar structural patterns in their ego networks are likely to share similar semantic meanings across different graphs, a finding largely supported by previous studies. For example, User 1's ego networks on Twitter and Foursquare, depicted as  and  , respectively.

Given that users usually exhibit stable social personalities and preferences across social platforms (Benevenuto et al. 2012), it is notable that the structural patterns—specifically, that many neighbors of this user are interconnected (demonstrating a high clustering coefficient)—are maintained across different social media networks. Such consistent structural patterns across platforms exemplify what we define as the transferable pattern.

Given the defined transferable pattern, we are now ready to define the graph data augmentation function that can preserve the transferable pattern in augmented views. Specifically, we define the graph data augmentation function as the subgraph sampling (Qiu et al. 2020), a technique to sample representative local structures of nodes from the original graph. This sampling is achieved by performing random walks with restart in a specified node's ego network (Tong et al. 2006). The walker starts from the specified node, goes randomly to one of its neighborhood with a probability, and returns to the ego node with another probability. This process is repeated multiple times to traverse various paths within the ego network. The subset of nodes visited during these walks induces the subgraph, known as the augmented view. Figure 2(a) illustrates the graph data augmentation on User 1's ($U1$) ego network. Through three walks of four steps each, paths like $U1 \rightarrow U2 \rightarrow U1 \rightarrow U3$ yield a subset of nodes visited: $\{U1, U2, U3, U5, U6, U7, U8, U9\}$. These nodes induce View 1, a subgraph that retains all the edges among the visited nodes present in the original ego network of User 1. A similar process generates View 2, differing only in the specific paths taken by the walker. These augmented views can be taken as a simulation of User 1's varying interaction patterns across different platforms. In this example, it is observed that despite fewer nodes, many neighbors of User 1 are interconnected in the augmented views (i.e., transferable pattern), similar to the ego network of User 1 on Twitter. Such augmented views thus effectively mimic User 1's ego network on Foursquare or other platforms.

Figure 2. (Color online) Illustrative Example of Graph Data Augmentation and Contrastive Learning



4.2.2. Graph Contrastive Learning. Given the augmented views generated by graph data augmentation, graph contrastive learning is then performed to empower the GNN model to learn transferable structural representations (Qiu et al. 2020, Xu et al. 2024). This can be done by maximizing the consistency between two augmented views of the same ego network compared with those of different ones. The optimization can be formulated as

$$\min_{\theta} \mathcal{L}_{\text{pretrain}} = \mathbb{E}_G \mathbb{E}_{t_1, t_2 \sim T}\, l_{\text{cl}}(e_\theta(t_1(G)), e_\theta(t_2(G))), \tag{2}$$

where l_{cl} is the contrastive loss InfoNCE, that is, maximizing the mutual information between the two representations (van den Oord et al. 2018), t_1 and t_2 are two augmented views sampled from graph data augmentation function set T, and e_θ is a GNN model with learnable parameter θ (serving as our graph pretraining model).

Figure 2(b) shows an illustrating example of how contrastive learning learns the transferable pattern. First, the privacy-preserving augmented views (which will be introduced in Section 4.3) of User 1, $G_{p\circ t_1}^{U1}$ and $G_{p\circ t_2}^{U1}$, are input into a GNN model, which then produces their corresponding representations, $e(G_{p\circ t_1}^{U1})$ and $e(G_{p\circ t_2}^{U1})$. Similarly, for User 99, we can also obtain the representations $e(G_{p\circ t_1}^{U99})$ and $e(G_{p\circ t_2}^{U99})$. Then, the graph contrastive learning trains the GNN model with the goal of maximizing the similarity between representations of two privacy-preserving augmented views of User 1, for example, $e(G_{p\circ t_1}^{U1})$ and $e(G_{p\circ t_2}^{U1})$ in this example. Simultaneously, it aims to maximize the discrepancy between representations of two different users' views, for example, $e(G_{p\circ t_1}^{U1})$ and $e(G_{p\circ t_1}^{U99})$.

4.2.3. GNN Model. For the GNN model, we utilize the GIN (Xu et al. 2019), one of the most expressive and state-of-the-art GNN architectures because of its capability of distinguishing graph structures. This capability is particularly critical in our graph data collaboration setup, where graph structure is the most valuable information. GIN effectively maps a graph G to distinct representations. At the l-th layer, GIN updates node v's representation as $h_v^{(l)} = \text{MLP}^{(l)}\left(h_v^{(l-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(l-1)}\right)$, where $\mathcal{N}(v)$ is the set of neighboring nodes of node v, and $\text{MLP}^{(l)}$ is a multilayer perceptron. Given the node representations, the graph G's representation h_G can be obtained by averaging the node representations at the final layer L, that is, $h_G = \text{MEAN}(\{h_v^{(L)} | v \in G\})$.

4.3. Privacy-Preserving Operator

Existing graph data augmentation discussed in Section 4.2 cannot preserve the privacy of the training data. To address this significant privacy concern, we propose a *privacy-preserving operator* designed to achieve the privacy-preserving principle. This operator integrates seamlessly with existing graph data augmentation techniques to form a novel *privacy-preserving graph data augmentation*. Through this privacy-preserving graph data augmentation, we can generate privacy-preserving augmented views that effectively eliminate private links with a high potential for leakage but also maintain model generalizability without compromise.

4.3.1. Privacy-Preserving Graph Data Augmentation. We present the formulation of the privacy-preserving graph data augmentation, which consists of two fundamental components: the graph data augmentation function $t(\cdot)$ and the privacy-preserving operator $p(\cdot)$.

Definition 3 (Privacy-Preserving Graph Data Augmentation). Given an ego network of a node G and a set of graph data augmentation functions T, the privacy-preserving graph data augmentation aims to eliminate edges that

pose a high risk of privacy leakage and have a low contribution to model generalizability in $t(G)$. This is achieved by applying a privacy-preserving operator $p(\cdot)$ on $t(G)$. The obtained privacy-preserving augmented view is

$$G_{p\circ t} = (p \circ t)(G), t \sim T, \tag{3}$$

where $(p \circ t)(G) = p(t(G))$, and $t(\cdot)$ is a function sampled from T.

In the privacy-preserving graph data augmentation, the graph data augmentation function $t(\cdot)$ and the privacy-preserving operator $p(\cdot)$ work collaboratively to generate the privacy-preserving augmented views that are both secure and effective. We will next detail the design of the privacy-preserving operator $p(\cdot)$, which is tailored to eliminate edges that pose significant privacy risks while having low contribution to the model generalizability.

4.3.2. Design of Privacy-Preserving Operator $p(\cdot)$. Our privacy-preserving operator is designed with the dual objective of minimizing privacy risks while maintaining model generalizability. Directly utilizing the augmented graph G_t may not suffice, as redundant graph features, such as private links with a high possibility of leakage, could still be retained in the randomly augmented views. In the effort to alleviate privacy leakage, we introduce a privacy-preserving operator $p(\cdot)$ and redefine the composition of $t(\cdot)$ and $p(\cdot)$ as our proposed privacy-preserving data augmentation. The resulting $G_{p\circ t}$ can be used as the privacy-preserving augmented view for graph pretraining, which mitigates possible privacy leakage and also maintains generalizability.

Admittedly, deleting all links to create an augmented graph view would ensure no privacy exposure but would create a dilemma: retaining some links could lead to privacy leakage, yet a model trained on a graph without any edges (or with randomly sampled links) could be ineffective. Therefore, we achieve privacy preservation by measuring the possibility of privacy leakage for each link and selectively removing links based on their possibility of privacy leakage. We refer to several heuristic metrics for computing the node similarity scores as the likelihood of link existence (Lü and Zhou 2011). By increasing the probability of removing edges with high heuristic metrics when generating augmentation views, the risk of leaking privacy could be lowered. Specifically, we choose three heuristic metrics to measure the possibility of privacy leakage associated with each link:

- *Preferential attachment*: This is a first-order heuristic based on one-hop neighbors of two target nodes (Barabási and Albert 1999). Preferential attachment (PA) between nodes x and y is $\mathrm{PA}(x,y) = |N(x)| \cdot |N(y)|$, where $N(x)$ denotes the neighbor set of node x.
- *Adamic–Adar*: This is a second-order heuristic that measures closeness based on shared neighbors within two-hop neighborhoods of two target nodes (Zhou et al. 2009). The Adamic–Adar (AA) score between nodes x and y is calculated as $\mathrm{AA}(x,y) = \sum_{z \in N(x) \cap N(y)} 1/\log|N(z)|$.
- *SimRank*: This measure reflects node similarity based on the entire graph, positing that similar nodes have similar neighbors (Jeh and Widom 2002). The SimRank (SR) score is calculated recursively: if $x = y$, then SimRank is defined as one. Otherwise, the SimRank score between nodes x and y is given by $\mathrm{SR}(x,y) = \gamma \sum_{a \in N(x)} \sum_{b \in N(y)} \mathrm{SR}(a,b)/(|N(x)| \cdot |N(y)|)$, where $\gamma \in [0,1]$ is a damping factor.

To measure privacy risks, edges with high scores in preferential attachment, Adamic–Adar, and SimRank are considered more vulnerable to privacy breaches. We calculate the privacy leakage score of an edge (x,y) as

$$p_{xy}^{\mathrm{del}} = \frac{\mathrm{MEAN}\left(\hat{\mathrm{PA}}(x,y), \hat{\mathrm{AA}}(x,y), \hat{\mathrm{SR}}(x,y)\right)}{\sum_{(a,b) \in E_t} \mathrm{MEAN}\left(\hat{\mathrm{PA}}(a,b), \hat{\mathrm{AA}}(a,b), \hat{\mathrm{SR}}(a,b)\right)}, \tag{4}$$

where E_t is the edge set of G_t, and $\hat{\mathrm{PA}}(x,y) = \mathrm{PA}(x,y)/\sum_{(a,b) \in E_t} \mathrm{PA}(a,b)$, and similarly for $\hat{\mathrm{AA}}$ and $\hat{\mathrm{SR}}$.

Given the risk posed by high privacy leakage scores p_{xy}^{del}, it is wise to consider removing such edges from the augmented graph G_t. To this end, we sample a designated percentage ($\tau^{\mathrm{del}}\%$) of the edges in G_t without replacement based on p_{xy}^{del} for potential removal. The sampled edges constitute an edge-removal candidate set $\mathcal{R}$.

Although removing all high-risk edges could safeguard privacy, this might overly distance the augmented view from the original graph, undermining the model's effectiveness for downstream tasks due to lost generalizability. Thus, we aim for our privacy-preserving data augmentation to avoid excessively aggressive edge removal by only *selectively* removing edges that pose high privacy risks but low contribution to generalizability. To achieve this balance, on one hand, we introduce a perturbation bound to constrain the number of edge removals. On the other hand, the operator should ensure essential edges remain to preserve the graph's semantics and maintain model generalizability. We leverage node centrality measures—degree centrality, eigenvector centrality, and PageRank centrality—as key metrics to identify crucial edges and nodes, drawing on their widespread use in network science (Newman 2010):

- *Degree centrality*: This measure reflects a node's influence based on the number of connections it has. Nodes with higher degrees are often more influential and can represent key figures in a network, like influencers in social networks.
- *Eigenvector centrality*: Unlike degree centrality, eigenvector centrality accounts for the influence of a node's neighbors, assigning more importance to nodes connected to other highly connected nodes. It is defined as the eigenvector corresponding to the greatest eigenvalue of the adjacency matrix of a graph. The eigenvector centrality of node x is the xth element of the eigenvector u, which is defined by $Au = \lambda u$, where A is the adjacency matrix of the training graph with the greatest eigenvalue λ.
- *PageRank centrality*: The PageRank algorithm can uncover the most influential nodes based on the mechanism of influence propagation along edges. The PageRank centrality of node x is $\pi_x = \alpha P \pi_x + \mathbf{1}$, where P is the transition matrix with $P_{i,j} = 1/|N(j)|$ if node i and j are connected, and otherwise $P_{i,j} = 0$. The vector $\mathbf{1}$ denotes an all-ones vector; α is a damping factor set as 0.85.

Using the centrality measures of nodes, edge centrality for an edge (x,y) is defined as the average node centrality of its connecting nodes, specifically, $\mathrm{Deg}(x,y)$, $\mathrm{Eig}(x,y)$, and $\mathrm{PR}(x,y)$ for the edge centrality of degree, eigenvector, and PageRank, respectively. The generalizability score of an edge (x,y) is calculated as a normalized weighted sum of these centralities, enhancing model generalization by emphasizing structurally important edges:

$$p_{xy}^{\mathrm{keep}} = \frac{\mathrm{MEAN}\left(\hat{\mathrm{Deg}}(x,y), \hat{\mathrm{Eig}}(x,y), \hat{\mathrm{PR}}(x,y)\right)}{\sum_{(a,b)\in E_t} \mathrm{MEAN}\left(\hat{\mathrm{Deg}}(a,b), \hat{\mathrm{Eig}}(a,b), \hat{\mathrm{PR}}(a,b)\right)}, \tag{5}$$

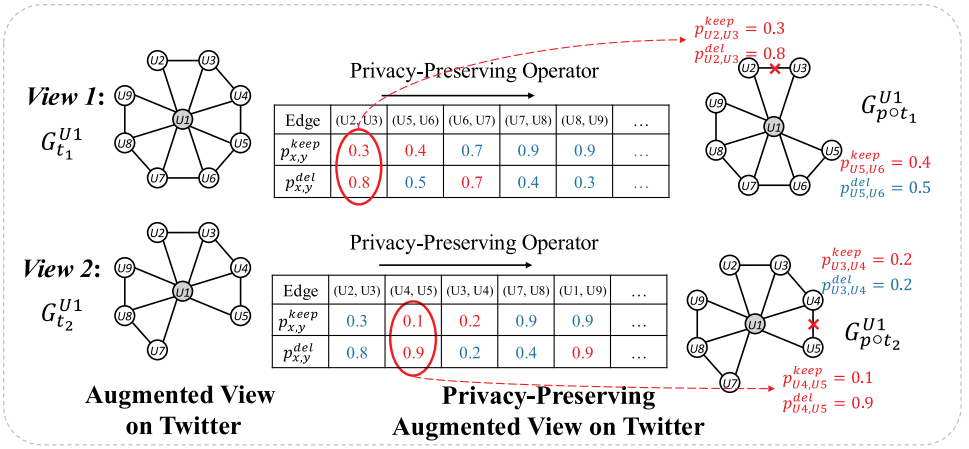
where $\hat{\mathrm{Deg}}(x,y) = \mathrm{Deg}(x,y)/\sum_{(a,b)\in E_t} \mathrm{Deg}(a,b)$, and similarly for $\hat{\mathrm{Eig}}$ and $\hat{\mathrm{PR}}$.

Given that edges with a higher generalizability score p_{xy}^{keep} contribute more to model generalizability, it is better to retain them in the augmented graph G_t. Therefore, we randomly sample the τ^{keep} percentage of the edges in G_t without replacement based on the score p_{xy}^{keep} for retention. The sampled edges constitute an edge-keeping candidate set $\mathcal{K}$.

To obtain the privacy-preserving augmented view $G_{p\circ t}$, we ensure it preserves privacy without sacrificing model generalizability. This involves removing edges from the augmented graph G_t that are identified in the edge-removal candidate set $\mathcal{R}$ but not protected by the edge-keeping candidate set $\mathcal{K}$. The resulting removal edges form the set $\mathcal{R} \setminus \mathcal{K}$. The final privacy-preserving augmented view $G_{p\circ t}$ is then obtained by removing edges in $\mathcal{R} \setminus \mathcal{K}$ from the augmented graph G_t. The privacy-preserving augmented view is subsequently used for model training.

Figure 3 details the workflow of the privacy-preserving operator applied to the augmented views of User 1's ego network (shown in Figure 2(a)). This operator first calculates two key scores for each edge (x,y) in the augmented view: the privacy leakage score p_{xy}^{del} and the generalizability score p_{xy}^{keep}. Edges with high privacy leakage scores are marked as removal candidates, forming the edge-removal candidate set $\mathcal{R}$. Edges with low generalizability scores are also marked as removal candidates, whereas those with high generalizability scores are identified as candidates for retention, comprising the edge-keeping candidate set $\mathcal{K}$. Consequently, edges featuring in both high privacy risk and low generalizability (i.e., edges in $\mathcal{R} \setminus \mathcal{K}$)—such as (*U2*, *U3*) in View 1 and (*U4*, *U5*) in

Figure 3. (Color online) Illustrative Example of Privacy-Preserving Operator



View 2—are removed from the augmented views. This selective removal process yields the final, privacy-preserving augmented view.

4.4. Generalizability Learning

This section introduces a generalizability learning strategy for pretraining a GNN model to uphold the generalizability principle. Specifically, this approach ensures that the pretrained GNN model, developed by the data owner, can be seamlessly adapted to downstream tasks on unseen graphs.

Drawing on invariant learning (Li et al. 2022), which suggests that the learned representation invariant across different environments can enhance generalizability, our objective is to produce a GNN model whose output representations can survive a large class of distribution shifts implied by different environments. To achieve such environment-invariant representations, existing invariant learning requires that their training data are collected from multiple environments (Chen et al. 2022, Li et al. 2022). For example, in a multigraph scenario, each graph typically comes from a different environment. However, the change of environments is unavailable in our cases, because the GNN model is only trained under a single environment, that is, the data owner's single graph G_{train}. To address this limitation, we propose simulating various environments, aimed at creating a set of environments diverse enough to facilitate the learning of environment-invariant representations, thereby mimicking the effect of having multiple real-world environments.

To simulate these environments effectively, we introduce a diverse graph data augmentation set Q, where each function $q_i \sim Q$ represents one diverse graph data augmentation function that can introduce variability. More specifically, given the input graph G, the set Q enables us to generate a series of augmented graphs $\{q_1(G), \ldots, q_K(G)\}$, ensuring that each resulting augmented graph is diverse from the others.

We further detail the workflow of each function $q_i(\cdot)$. Given the input training graph $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}})$, we introduce variability to simulate diverse environments by assigning a random bias to each edge. For each edge $(x, y) \in E_{\text{train}}$, a random bias $\pi_i^{(x,y)}$ is assigned from a uniform distribution:

$$\pi_i^{(x,y)} \sim \text{Uniform}(0,1), \quad \forall (x,y) \in E_{\text{train}}.$$

This transforms G_{train} into a graph with bias $G_{\text{train}}^{\pi_i}$ (which can also be viewed as G_{train} with edge weights π_i). This step injects distinct randomness in different $q_i(\cdot)$, thereby enabling different $q_i(\cdot)$ to mimic different environments.

Subsequently, to generate the augmented graphs from $G_{\text{train}}^{\pi_i}$, a biased random walk with restart is employed to sample subgraphs. The transition probability from node x to node y is

$$P(y|x) = \frac{\pi_i^{(x,y)}}{\sum_{j \in N(x)} \pi_i^{(x,y)}},$$

where $N(x)$ denotes the set of neighbors of x in G_{train}. This biased random walk ensures that the probability of moving to a neighboring node y is proportional to the weight $\pi_i^{(x,y)}$, simulating different "environmental" effects on the graph structure.

By doing so, $q_i(\cdot)$ can help generate different environments. Recall that the invariant learning aims to learn the representations that are invariant across different environments. Therefore, the learned representation $e(G)$ of the GNN model should satisfy

$$\mathbb{P}^{\text{do}(p \circ q_i)}(e(G)|G) = \mathbb{P}^{\text{do}(p \circ q_j)}(e(G)|G), \ \forall q_i, q_j \sim Q, \tag{6}$$

where q_i and q_j are the functions sampled from Q that simulate different environments, and $\mathbb{P}^{\text{do}(p \circ q_i)}$ denotes the distribution by doing $p \circ q_i$ on the graph.

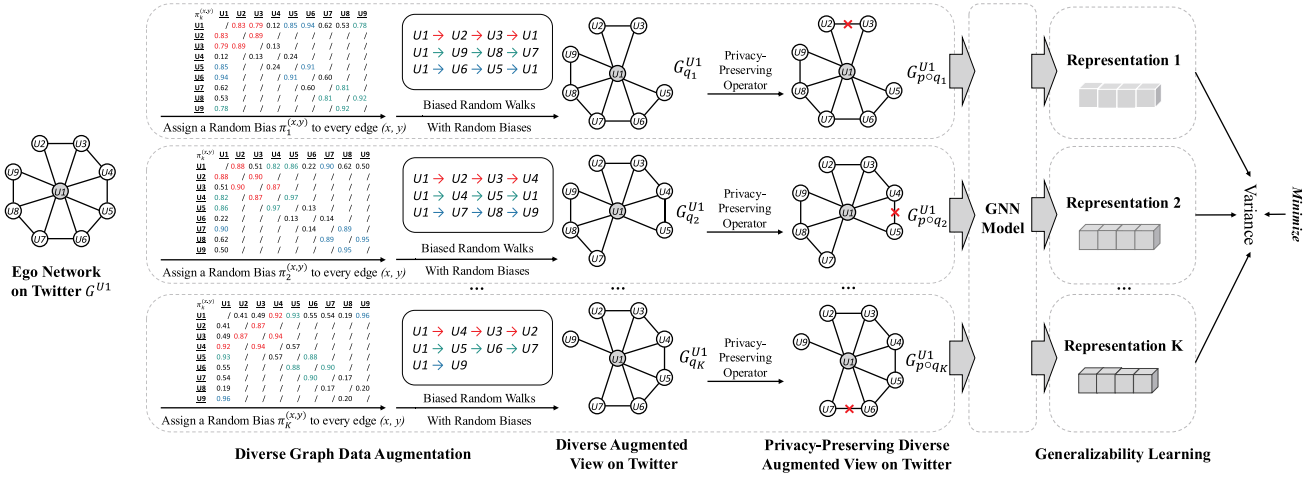
To satisfy Equation (6), the GNN model e must produce similar output representations under varying environments $G_{p \circ q_i}$. This can be done by minimizing the representation variance:

$$\min_e \mathcal{L}_{\text{general}} = \mathbb{E}_G \text{Var}(\{\|e(G_{p \circ q_i}) - \mu\| : q_i \sim Q\}), \tag{7}$$

where μ is the expectation of $\{e(G_{p \circ q_i}) : q_i \sim Q\}$.

We use Figure 4 to illustrate the workflow of generalizability learning. To simulate diverse environments, the diverse graph data augmentation technique is first applied. We take the first view $G_{q_1}^{U1}$ as an illustrative example. To obtain $G_{q_1}^{U1}$, a random bias $\pi_1^{(x,y)}$ is assigned to every edge (x, y) from a uniform distribution, as shown by the numbers in the table in Figure 4. This bias influences the transition probabilities for biased random walks, dictating three walks of length four, resulting in paths like $U1 \to U2 \to U3 \to U1$. This process induces a subgraph $G_{q_1}^{U1}$ from the nodes traversed. Notably, edges with low biases such as $(U1, U4)$ are excluded, which are less likely to

Figure 4. (Color online) Illustrative Example of Generalizability Learning



be traversed. Ultimately, by implementing K diverse graph data augmentations, each characterized by unique biases during the biased random walk, we obtain K augmented views $G^{U1}_{q_1}, G^{U1}_{q_2}, \ldots, G^{U1}_{q_K}$. Each view is diverse from the others, ensuring a wide variety of simulated environments. Next, these views are processed by the privacy-preserving operator to yield corresponding privacy-preserving diverse augmented views $G^{U1}_{p\circ q_1}, G^{U1}_{p\circ q_2}, \ldots, G^{U1}_{p\circ q_K}$ (a process we have already introduced by the example in Figure 3). Subsequently, the GNN model processes them to generate their representations. The objective of generalizability learning is to learn representations that are invariant to environmental changes by minimizing the variance among these representations.

5. Experiments

This section evaluates our method's privacy-preservation and generalizability capabilities. Additional experiments and results on cross-relation and cross-time scenarios, ablation studies, efficiency analysis, and case studies can be found in Online Appendix D.

5.1. Experiment Setup

5.1.1. Experimental Settings. To evaluate the privacy-preserving performance, we examine the effectiveness of an adversary in conducting a model-based private link reconstruction attack. Specifically, we assume that there exists an adversary in the model user that aims to reconstruct all the links in G_{train}, that is, infer the link existence between all node pairs. The adversary typically could query the released pretrained GNN model with its own graph data (i.e., the downstream graph G_{down}) and obtain the node representations of G_{down}. To mimic the adversary's strategy, we first train an adversarial link predictor $h_\phi : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ by taking 90% existent links in G_{down} as positive samples and the same number of nonexistent links in G_{down} as negative samples, and the remaining 10% of existent links and the same number of nonexistent links as the validation set of the adversarial link predictor. The goal of adversarial link predictor is optimized as

$$\max_{\phi} \sum_{(u,v)\in P_{\text{pos}}} \log h_\phi(e(G_{\text{down}})[u], e(G_{\text{down}})[v]) - \sum_{(u,v)\in P_{\text{neg}}} \log h_\phi(e(G_{\text{down}})[u], e(G_{\text{down}})[v]),$$

where P_{pos} and P_{neg} are the sets of existing and nonexistent links in G_{train}, respectively, and $e(G_{\text{down}})[u]$ and $e(G_{\text{down}})[v]$ are the queried representations of nodes u and v from the pretrained model e. Then, we use the trained link predictor to infer the private information (i.e., the link existence of all node pairs) in G_{train}. We take all existent links in G_{train} as positive samples and four times as many nonexisting links in G_{train} as negative samples to evaluate the pretrained GNN model's privacy-preserving performance. We report the area under the curve (AUC), where a lower value is preferable as it indicates reduced ability of the adversary to infer private links, thereby enhancing privacy protection.

Next, we measure generalizability by evaluating the model user's downstream task performance on the downstream graph G_{down}. These tasks include *node classification* and *link prediction*. For node classification, the model user can obtain node representations of G_{down} by querying the pretrained GNN model provided by the data owner. Then, the obtained node representations are fed into a logistic classifier for node classification, evaluated

using 10-fold cross-validation. For link prediction, we construct the training data using all existing links as positive samples and randomly select nonexistent links, at five times the number of positive samples, as negative samples. Then we train a downstream link predictor on the training data that takes a pair of node representations as input and outputs the link existence probability. We evaluate the performance on the test data that comprise unobserved existing links and nonexistent links at five times the number of these existing links. A higher performance in downstream tasks (i.e., downstream performance), measured by micro F1 scores, indicates better model generalizability. This is because we evaluate the downstream performance on a graph that differs from the training graph, effectively accounting for the model's generalizability to unseen data.

5.1.2. Data Sets and Scenarios. As mentioned earlier, data collaborations typically begin when a data owner, who possesses high-quality and abundant data, partners with a model user having much fewer data. We thus examine scenarios where the data owner has about twice the data of the model user. In the first, the data owner's and model user's graphs share the same types of relations. In the second, their graphs derive from different data sets.

In the first scenario, we evaluate our model on three social network data sets: Deezer (Rozemberczki et al. 2019), Facebook (Rozemberczki et al. 2021), and LastFM (Rozemberczki and Sarkar 2020). Each data set involves a node classification task: predicting user gender for Deezer, classifying Facebook pages into categories such as politicians or companies, and determining user location for LastFM. To simulate a model-sharing scenario, we construct the data owner's training graph using Kruskal's (1956) algorithm to form a shortest spanning tree. We then sample and add half of the remaining edges from the graph G to the spanning tree, forming the training graph $G_{\text{train}} = (V_{\text{train}}, E_{\text{train}})$. This approach ensures the connectivity of the training graph. Additionally, we construct the model user's downstream graph in line with the conditions of data advantage from the data owner and data commonality, ensuring the model user benefits from the model-sharing strategy. More specifically, we sample half of the nodes from the training graph's nodes V_{train} as downstream graph nodes V_{down}. The edges among V_{down} are sampled by performing random walks started at each node in V_{down}.

In the more realistic scenario where data owner and model user graphs stem from two different data sets, we evaluate on the Twitter–Foursquare (Zhang and Philip 2015) and phone–email (Zhang et al. 2020) data sets. As these data sets lack node labels, we focus on link prediction tasks. The Twitter–Foursquare data set contains two social networks from Twitter and Foursquare. To simulate the data advantage, Twitter is used as the training graph, whereas a subgraph induced from random walks on half of the Foursquare nodes serves as the downstream graph for the model user. Similarly, for the phone–email data set, we use the phone network as the training graph and create the downstream graph by performing random walks on a sampled node subset of the email network. The subgraph induced from these walks serves as the downstream graph.

In all these scenarios, for the link prediction downstream task, the test data (i.e., comprising links unobserved from the downstream graph) are constructed by sampling edges from the original data set that are among V_{down} but not part of E_{down}, selecting approximately $\text{int}(3/7|E_{\text{down}}|)$ as positive samples. Nonexistent links are randomly sampled at a rate five times the number of these positive samples to form the negative samples.

5.1.3. Baselines. We compare our method with the following privacy-preserving baselines that can be divided into three types: (1) adversarial learning–based methods, GAL-W and GAL-TV (Liao et al. 2021); (2) differential privacy–based methods, EdgeRand (Wu et al. 2022) and LapGraph (Wu et al. 2022); and (3) a graph pretraining model without a privacy-preserving mechanism, GCC (Qiu et al. 2020). Details can be found in Online Appendix C.

5.1.4. Implementation Details. We follow Qiu et al. (2020) and Liao et al. (2021) to tune the hyperparameters of our method. During pretraining, we set the iteration number to 132, batch size to 128, and learning rate to 5e-3, and use a five-layer graph encoder with a hidden size of 64. The τ^{del}, τ^{keep}, and K are set to 0.8, 0.8, and 10. When evaluating model generalizability on the link prediction task, the queried representations from the pretrained model are used as node attributes in the downstream graph. This graph is then fed into a one-layer GIN (Xu et al. 2019) to obtain learned node representations. The probability of link existence is determined by taking the inner product of their learned node representations, followed by a sigmoid function. The adversarial link predictor is trained using a two-layer MLP with a hidden size of 32. For other implementation details, refer to Online Appendix C.

5.1.5. Availability. The data sets, source code, and detailed results are available online (Xu et al. 2025).

5.2. Experimental Results

5.2.1. Privacy-Preserving Performance. Table 1 shows the privacy-preserving capability of different methods. Among all the baselines, EdgeRand, LadGraph, and GCC are three models empowered with generalizability.

Table 1. The Privacy-Preserving Performance of Private Link Reconstruction Attack, Reported as AUC

	Private link reconstruction attack (AUC%)				
Methods	Deezer	Facebook page	LastFM	Twitter–Foursquare	Phone–email
GCC	86.19 (0.38)	92.02 (0.21)	88.70 (0.09)	72.28 (0.51)	82.32 (0.23)
GAL-W	69.29 (0.29)	72.12 (0.08)	74.49 (0.24)	64.24 (0.27)	70.06 (0.64)
GAL-TV	71.52 (0.25)	75.83 (0.26)	78.28 (0.23)	65.02 (0.31)	71.67 (0.24)
EdgeRand	83.05 (0.62)	85.29 (0.58)	87.45 (0.40)	69.15 (0.21)	78.90 (0.22)
LapGraph	83.75 (0.08)	85.90 (0.19)	86.83 (0.30)	69.33 (0.15)	79.47 (0.39)
Ours	82.35 (0.17)	83.80 (0.15)	83.55 (0.11)	68.82 (0.24)	74.87 (0.37)

Note. Standard deviations are in parentheses.

Compared with them, we can see that our model achieves the best privacy-preserving capability, which indicates our superiority in protecting private information in the training graph. The privacy-preserving performance of our model is attributable to the carefully designed privacy-preserving graph data augmentation, which utilizes various heuristic metrics to estimate the private links with a high possibility of privacy leakage. Note that though GAL-W and GAL-TV show privacy-preserving capability, they cannot serve the pretraining model in the model-sharing strategy because of their extremely limited generalizability performance (see Table 2).

5.2.2. Model Generalizability Performance. Table 2 shows the generalizability of each method on downstream tasks of node classification and link prediction. The results indicate that the privacy-preserving capability of our model is obtained without sacrificing generalizability and, in some cases, even slightly enhances it. Compared with the graph pretraining model without privacy protection (i.e., GCC), our model achieves almost the same generalizability results but well preserves the privacy. We also find that LapGraph and EdgeRand achieve strong generalizability but show poor privacy-protection capability. The potential reason might be that both methods add noise to the adjacency matrix of the training graph, thus improving generalizability. However, the random noise is not sufficient to preserve privacy because it fails to effectively remove the private links that have a high possibility of leakage.

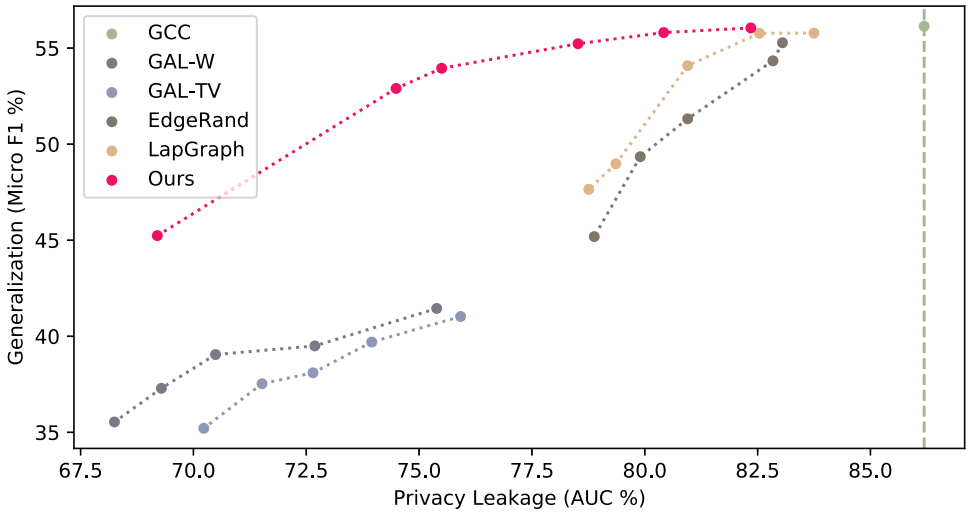
5.2.3. Privacy–Generalizability Trade-Off. Our analysis further delves into the privacy–generalizability trade-off across different models. Specifically, we systematically assessed this trade-off by varying the trade-off parameters of our model, as well as those of baselines, which include differential privacy–based methods (i.e., EdgeRand and LapGraph), adversarial training–based methods (i.e., GAL-W and GAL-TV), and a pretraining model without privacy preservation (i.e., GCC) on the Deezer data set. To elaborate, the trade-off parameter τ^{del} in our model varies among 0.7, 0.8, 0.9, 1.0, 1.1, and 1.3; ϵ in EdgeRand and LapGraph varies among 2, 3, 4, 5, and 6; and the equivalent parameter λ in GAL-W and GAL-TV spans 0.4, 0.5, 0.6, 0.7, and 0.8. Figure 5 illustrates that higher x-axis values indicate lower privacy preservation, whereas higher y-axis values reflect better generalizability. Although GAL-W and GAL-TV are good at preserving privacy, their limited generalizability affects their use as pretraining models in a model-sharing strategy. Compared with EdgeRand, LapGraph, and GCC, our model, ideally positioned in the upper-left corner, demonstrates the best balance between generalizability and privacy protection.

Table 2. The Generalizability Performance of Downstream Tasks on Node Classification and Link Prediction, Reported as Micro F1

	Node classification (micro F1%)			Link prediction (micro F1%)				
Methods	Deezer	Facebook page	LastFM	Deezer	Facebook page	LastFM	Twitter–Foursquare	Phone–email
GCC	56.14 (0.35)	49.26 (0.14)	23.62 (0.35)	78.24 (0.69)	78.85 (0.59)	88.30 (0.45)	82.32 (0.23)	87.80 (0.29)
GAL-W	37.29 (0.16)	40.85 (0.10)	11.83 (0.06)	67.41 (0.53)	70.87 (0.64)	75.73 (0.51)	68.75 (0.20)	76.36 (0.20)
GAL-TV	37.53 (0.60)	42.96 (0.06)	13.15 (0.37)	69.59 (0.38)	70.92 (0.33)	76.91 (0.36)	69.50 (0.42)	77.40 (0.22)
EdgeRand	55.28 (0.31)	48.53 (0.16)	23.22 (0.26)	77.42 (0.29)	79.67 (0.61)	88.88 (0.69)	82.53 (0.38)	88.05 (0.74)
LapGraph	55.78 (0.38)	49.06 (0.13)	22.91 (0.53)	78.96 (0.51)	79.56 (0.75)	88.09 (0.25)	83.10 (0.35)	87.34 (0.59)
Ours	56.05 (0.18)	49.62 (0.19)	23.57 (0.30)	78.77 (0.54)	78.05 (0.51)	89.30 (0.24)	83.39 (0.24)	88.41 (0.22)

Note. Standard deviations are in parentheses.

Figure 5. (Color online) Experimental Results of Privacy–Generalizability Trade-Off



Notes. The spatial positions of the methods in the figure help distinguish them. "Ours" appears as the topmost line in the upper-right corner, indicating the best generalization-privacy trade-off. LapGraph and EdgeRand are located just below "Ours" on the upper right, showing high generalization but slightly more privacy leakage. GCC is shown as a single point at the far right, marking the highest privacy leakage. GAL-W and GAL-TV occupy the lower-left area, indicating lower privacy leakage but also lower generalization.

6. Conclusions and Discussion

This study introduces an innovative model-sharing strategy for graph data collaboration that avoids direct data sharing. The model-sharing strategy allows the data owner to pretrain a GNN model on their private data and then provide query access to the model for the model user. This enables the model user to query the pretrained model with their own data, gaining insights without directly accessing the private data. To successfully achieve this strategy, two fundamental principles are essential for the pretrained GNN model: privacy preservation and generalizability. The privacy-preserving operator integrates seamlessly with graph data augmentation to effectively eliminate private links with a high possibility of leakage while ensuring generalizability. The generalizability learning is innovated to enhance the model's adaptability to diverse environments, which enhances model generalizability when the model is deployed on unseen data. By integrating these two modules into the basic graph pretraining framework, the proposed strategy achieves a strategic balance between generalizability and privacy preservation. Our method is applicable to a variety of real-world applications. To illustrate, consider a data collaboration scenario between Twitter (data owner) and Foursquare (model user). Foursquare can enhance its business operations by leveraging Twitter's user network without compromising privacy. Our experiments with the Twitter–Foursquare data set indicate that the Twitter's model pretrained by our proposed strategy surpasses most benchmarks in privacy preservation and generalizability.

Acknowledgments

The authors are grateful to the area editor, associate editor, and reviewers for their insightful comments and constructive feedback throughout the review process.

Endnotes

[1] See "Why Using Graph Analytics for Big Data Is On the Rise," https://www.techtarget.com/searchbusinessanalytics/feature/Why-using-graph-analytics-for-big-data-is-on-the-rise (accessed on September 16, 2023).

[2] See "Global Valuation of Graph Database Market Size & Share Will Reach USD 4,500 Million By 2026: Facts & Factors," https://www.globenewswire.com/en/news-release/2021/10/07/2310420/0/en/Global-Valuation-of-Graph-Database-Market-Size-Share-Will-Reach-USD-4-500-Million-By-2026-Facts-Factors.html (accessed on September 16, 2023).

[3] Twitter was rebranded as "X" in 2023. In this paper, the platform is referred to by its original name, "Twitter", for clarity and because of the widespread recognition of the former name.

[4] For a node, its k-ego network is the subgraph induced by its neighbors within k hops.

References

Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. Weippl E, Katzenbeisser S, Kruegel C, Myers A, Halevi S, eds. *Proc. 2016 ACM SIGSAC Conf. Comput. Comm. Security* (Association for Computing Machinery, New York), 308–318.

Arora R, Upadhyay J (2019) On differentially private graph sparsification and applications. Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, eds. *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Red Hook, NY), 13411–13422.

Barabási AL (2013) Network science. *Philos. Trans. Roy. Soc. A Math. Phys. Engrg. Sci.* 371(1987):20120375.

Barabási AL, Albert R (1999) Emergence of scaling in random networks. *Science* 286(5439):509–512.

Bauer M, van Dinther C, Kiefer D (2020) Machine learning in SME: An empirical study on enablers and success factors. Anderson B, Thatcher J, Meservy R, Chudoba K, Fadel K, Brown S, eds. *AMCIS 2020 Proc. Adv. Inform. Systems Res.* (Association for Information Systems, Atlanta), 1–10.

Benevenuto F, Rodrigues T, Cha M, Almeida V (2012) Characterizing user navigation and interactions in online social networks. *Inform. Sci.* 195:1–24.

Cao Y, Xu J, Yang C, Wang J, Zhang Y, Wang C, Chen L, Yang Y (2023) When to pre-train graph neural networks? From data generation perspective! Singh A, Sun Y, Akoglu L, Gunopulos D, Yan X, Kumar R, Ozcan F, Ye J, eds. *Proc. 29th ACM SIGKDD Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 142–153.

Chen Y, Zhang Y, Bian Y, Yang H, Kaili M, Xie B, Liu T, Han B, Cheng J (2022) Learning causally invariant representations for out-of-distribution generalization on graphs. Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, eds. *Advances in Neural Information Processing Systems*, vol. 35 (Curran Associates, Red Hook, NY), 22131–22148.

Feng F, He X, Tang J, Chua TS (2019) Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Trans. Knowledge Data Engrg.* 33(6):2493–2504.

Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. Krishnapuram B, Shah M, Smola A, Aggarwal C, Shen D, Rastogi R, eds. *Proc. 22nd ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 855–864.

Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. Guyon I, Von Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, eds. *Advances in Neural Information Processing Systems*, vol. 30 (Curran Associates, Red Hook, NY), 1025–1035.

Han X, Yang Y, Wang L, Wu J (2024) Privacy-preserving network embedding against private link inference attacks. *IEEE Trans. Dependable Secure Comput.* 21(2):847–859.

Han X, Zhang Z, Ding N, Gu Y, Liu X, Huo Y, Qiu J, et al. (2021) Pre-trained models: Past, present and future. *AI Open* 2:225–250.

Hu H, Cheng L, Vap JP, Borowczak M (2022) Learning privacy-preserving graph convolutional network with partially observed sensitive attributes. Laforest F, Troncy R, Simperl E, Agarwal D, Gionis A, Herman I, Médini L, eds. *Proc. ACM Web Conf. 2022* (Association for Computing Machinery, New York), 3552–3561.

Hu Z, Dong Y, Wang K, Chang KW, Sun Y (2020a) GPT-GNN: Generative pre-training of graph neural networks. Gupta R, Liu Y, Shah M, Rajan S, Tang J, Prakash BA, eds. *Proc. 26th ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 1857–1867.

Hu W, Liu B, Gomes J, Zitnik M, Liang P, Pande VS, Leskovec J (2020b) Strategies for pre-training graph neural networks. *8th Internat. Conf. Learn. Representations, ICLR 2020 (Addis Ababa, Ethiopia)*.

Huang R, Xu J, Jiang X, Pan C, Yang Z, Wang C, Yang Y (2024) Measuring task similarity and its implication in fine-tuning graph neural networks. Wooldridge M, Dy J, Natarajan S, eds. *Proc. 38th AAAI Conf. Artificial Intelligence 36th Conf. Innovative Applications Artificial Intelligence 14th Sympos. Ed. Adv. Artificial Intelligence* (AAAI Press, Washington, DC), 12617–12625.

Jeh G, Widom J (2002) SimRank: A measure of structural-context similarity. Zaïane OR, Goebel R, Hand D, Keim D, Ng R, eds. *Proc. Eighth ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 538–543.

Kipf TN, Welling M (2016) Variational graph auto-encoders. Preprint, submitted November 21, https://arxiv.org/abs/1611.07308.

Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.* 7(1):48–50.

Li T, Li N (2009) On the tradeoff between privacy and utility in data publishing. Elder J, Fogelman FS, Flach P, Zaki M, eds. *Proc. 15th ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 517–526.

Li H, Zhang Z, Wang X, Zhu W (2022) Learning invariant graph representations for out-of-distribution generalization. Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, eds. *Advances in Neural Information Processing Systems*, vol. 35 (Curran Associates, Red Hook, NY), 11828–11841.

Liao P, Zhao H, Xu K, Jaakkola T, Gordon GJ, Jegelka S, Salakhutdinov R (2021) Information obfuscation of graph neural networks. Meila M, Zhang T, eds. *Proc. 38th Internat. Conf. Machine Learn.* (JMLR.org), 6600–6610.

Lü L, Zhou T (2011) Link prediction in complex networks: A survey. *Physica A Statist. Mechanics Appl.* 390(6):1150–1170.

Newman M (2010) *Networks: An Introduction* (Oxford University Press, New York).

Orekondy T, Schiele B, Fritz M (2019) Knockoff nets: Stealing functionality of black-box models. Davis L, Torr P, Zhu SC, eds. *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognition* (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 4954–4963.

Qiu J, Chen Q, Dong Y, Zhang J, Yang H, Ding M, Wang K, Tang J (2020) GCC: Graph contrastive coding for graph neural network pre-training. Gupta R, Liu Y, Shah M, Rajan S, Tang J, Prakash BA, eds. *Proc. 26th ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 1150–1160.

Rigaki M, Garcia S (2023) A survey of privacy attacks in machine learning. *ACM Comput. Surv.* 56(4):1–34.

Rong Y, Bian Y, Xu T, Xie W, Wei Y, Huang W, Huang J (2020) Self-supervised graph transformer on large-scale molecular data. Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, eds. *Advances in Neural Information Processing Systems*, vol. 33 (Curran Associates, Red Hook, NY), 12559–12571.

Rozemberczki B, Sarkar R (2020) Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. d'Aquin M, Dietze S, Hauff C, Curry E, Mauroux PC, eds. *Proc. 29th ACM Internat. Conf. Inform. Knowledge Management* (Association for Computing Machinery, New York), 1325–1334.

Rozemberczki B, Allen C, Sarkar R (2021) Multi-scale attributed node embedding. *J. Complex Networks* 9(2):cnab014.
Rozemberczki B, Davies R, Sarkar R, Sutton C (2019) GEMSEC: Graph embedding with self clustering. Spezzano F, Chen W, Xiao X, eds. *Proc. 2019 IEEE/ACM Internat. Conf. Adv. Soc. Networks Anal. Mining 2019* (Association for Computing Machinery, New York), 65–72.
Sajadmanesh S, Shamsabadi AS, Bellet A, Gatica-Perez D (2023) Gap: Differentially private graph neural networks with aggregation perturbation. Calandrino J, Troncoso C, eds. *Proc. 32nd USENIX Conf. Security Sympos* (USENIX Association, Berkeley, CA), 3223–3240.
Smith A, Anderson M (2018) Social media use in 2018. Accessed September 16, 2023, http://www.pewinternet.org/2018/03/01/social-media-use-in-2018.
Tong H, Faloutsos C, Pan JY (2006) Fast random walk with restart and its applications. Clifton CW, Zhong N, Liu J, Wah BW, Wu X, eds. *Proc. Sixth Internat. Conf. Data Mining* (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 613–622.
van den Oord A, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. Preprint, submitted July 10, https://arxiv.org/abs/1807.03748.
Wang B, Guo J, Li A, Chen Y, Li H (2021) Privacy-preserving representation learning on graphs: A mutual information perspective. Zhu F, Ooi BC, Miao C, Wang H, Skrypnyk I, Hsu W, Chawla S, eds. *Proc. 27th ACM SIGKDD Conf. Knowledge Discovery Data Mining* (Association for Computing Machinery, New York), 1667–1676.
Wu F, Long Y, Zhang C, Li B (2022) LINKTELLER: Recovering private edges from graph neural networks via influence analysis. Holz T, Ristenpart T, eds. *IEEE Sympos. Security Privacy* (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 2005–2024.
Xie H, Ma J, Xiong L, Yang C (2021) Federated graph classification over non-IID graphs. Ranzato M, Beygelzimer A, Dauphin Y, Liang PS, Wortman Vaughan J, eds. *Advances in Neural Information Processing Systems*, vol. 34 (Curran Associates, Red Hook, NY), 18839–18852.
Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? *7th Internat. Conf. Learn. Representations ICLR 2019 (New Orleans, LA).*
Xu J, Wang J, Zhou Z, Lu T (2025) Toward graph data collaboration in a data-sharing-free manner: A novel privacy-preserving graph pretraining model. https://doi.org/10.1287/ijoc.2023.0115.cd, https://github.com/INFORMSJoC/2023.0115.
Xu J, Huang R, Jiang X, Cao Y, Yang C, Wang C, Yang Y (2024) Better with less: A data-active perspective on pre-training graph neural networks. Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, eds. *Proc. 37th Internat. Conf. Neural Inform. Processing Systems* (Curran Associates, Red Hook, NY), 56946–56978.
You Y, Chen T, Sui Y, Chen T, Wang Z, Shen Y (2020) Graph contrastive learning with augmentations. Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, eds. *Advances in Neural Information Processing Systems*, vol. 33 (Curran Associates, Red Hook, NY), 5812–5823.
Zhang J, Philip SY (2015) Integrated anchor and social link predictions across social networks. Yang Q, Wooldridge M, eds. *Proc. 24th Internat. Joint Conf. Artificial Intelligence* (AAAI Press, Washington, DC), 2125–2131.
Zhang S, Tong H, Tang J, Xu J, Fan W (2020) Incomplete network alignment: Problem definitions and fast solutions. *ACM Trans. Knowledge Discovery Data* 14(4):1–26.
Zhou T, Lü L, Zhang YC (2009) Predicting missing links via local information. *Eur. Phys. J. B.* 71(4):623–630.
Zhou F, Zhang K, Xie S, Luo X (2020) Learning to correlate accounts across online social networks: An embedding-based approach. *INFORMS J. Comput.* 32(3):714–729.
Zhou Z, Zhou C, Li X, Yao J, Yao Q, Han B (2023) On strengthening and defending graph reconstruction attack with Markov chain approximation. Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, eds. *Proc. 40th Internat. Conf. Machine Learn.* (JMLR.org), 42843–42877.