

# Structure-Preference Enabled Graph Embedding Generation under Differential Privacy

Sen Zhang, Qingqing Ye, Haibo Hu  
The Hong Kong Polytechnic University, Hong Kong  
{senzhang, qqing.ye, haibo.hu}@polyu.edu.hk

**Abstract**—Graph embedding generation techniques aim to learn low-dimensional vectors for each node in a graph and have recently gained increasing research attention. Publishing low-dimensional node vectors enables various graph analysis tasks, such as structural equivalence and link prediction. Yet, improper publication opens a backdoor to malicious attackers, who can infer sensitive information of individuals from the low-dimensional node vectors. Existing methods tackle this issue by developing deep graph learning models with differential privacy (DP). However, they often suffer from large noise injections and cannot provide structural preferences consistent with mining objectives. Recently, skip-gram based graph embedding generation techniques are widely used due to their ability to extract customizable structures. Based on skip-gram, we present SE-PrivGEmb, a structure-preference enabled graph embedding generation under DP. For arbitrary structure preferences, we design a unified noise tolerance mechanism via perturbing non-zero vectors. This mechanism mitigates utility degradation caused by high sensitivity. By carefully designing negative sampling probabilities in skip-gram, we theoretically demonstrate that skip-gram can preserve arbitrary proximities, which quantify structural features in graphs. Extensive experiments show that our method outperforms existing state-of-the-art methods under structural equivalence and link prediction tasks. Our code is available at <https://github.com/DPGram/SEPrivGEmb>.

**Index Terms**—Structure preference, Differential privacy, Graph embedding generation

## I. INTRODUCTION

In recent years, graph embedding generation has gained significant research attention due to its ability to represent nodes with low-dimensional vectors while preserving the inherent properties and structure of the graph. The publication of low-dimensional node vectors facilitates a broad spectrum of graph analysis tasks, including structural equivalence and link prediction. However, improper publication creates opportunities for malicious attackers to potentially infer sensitive individual information. Therefore, it is crucial to integrate privacy-preserving techniques into graph embedding generation methods before making them publicly available.

Differential privacy (DP) is a well-studied statistical privacy model known for its rigorous mathematical framework. One common technique for achieving differentially private training is the combination of noisy Stochastic Gradient Descent (SGD) and advanced composition theorems such as Moments Accountant (MA) [1]. This combination, known as DPSGD, has been widely studied in recent years for publishing low-dimensional node vectors, as the advanced composition theorems can effectively manage the problem of excessive splitting

of the privacy budget during optimization. For instance, Yang *et al.* [2] privately publish low-dimensional node vectors by extending the MA mechanism and applying it to graph models based on Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). Another line of research [3]–[8] explores graph embedding generation by applying DPSGD to Graph Neural Networks (GNNs). Despite their success, DPSGD is inherently suitable for structured data with well-defined individual gradients, while scaling poorly for graph learning models. The main reason is that individual examples in a graph are no longer independently computed for their gradients. As a result, *existing methods typically suffer from large noise caused by high sensitivity*, making it difficult to strike a balance between privacy and utility. Additionally, *none of existing methods can offer structure-preference settings*. Setting preferences is essential for extracting specific graph structures that align with mining objectives, improve predictive accuracy, and yield meaningful insights.

To address these issues, we turn to the skip-gram model [9]–[12], an advanced graph embedding generation technique known for its capability for customizable structure extraction. Building on skip-gram, we propose SE-PrivGEmb, a novel approach for achieving differentially private graph embedding generation while supporting structure-preference settings. Achieving this target faces two main challenges:

- *High Sensitivity*. Different structure-preference settings will result in varying sensitivities. The maximum sensitivity for arbitrary preference settings can be proportional to the size of the batch sampling, which leads to large noise injection and poor data utility.
- *Theoretical Analysis*. Previous skip-gram based methods [9]–[13] only preserve the structural feature related to node degree. They lack theoretical analysis of exactly what relationships they preserve in their embedding space, which results in weak interpretability for structural preferences in graph embedding generation.

Inspired by the one-hot encoding used in skip-gram, which ensures that the gradient (typically represented as a matrix) updates only partial vectors, this provides a new perspective for addressing high sensitivity. To tackle the first challenge, we conduct an in-depth analysis of skip-gram optimization, and design a unified noise tolerance mechanism by perturbing non-zero vectors to accommodate arbitrary structure preferences. This mechanism effectively mitigates the utility degradation

caused by high sensitivity. To address the second challenge, we carefully design the negative sampling probability in skip-gram and theoretically prove that skip-gram can preserve arbitrary node proximities (which quantify structural features). Through formal privacy analysis, we demonstrate that the low-dimensional node vectors generated by SE-PrivGEmb satisfy node-level Rényi Differential Privacy (RDP).

The main contributions can be summarized as follows:

- We present a novel method for achieving differentially private graph embedding generation, named SE-PrivGEmb. To our best knowledge, it is the *first* method that supports structure-preference settings while satisfying node-level RDP and preserving high data utility.
- We deeply analyze the optimization of skip-gram and design a unified noise tolerance mechanism via perturbing non-zero vectors to accommodate arbitrary structure preferences. This mechanism can effectively mitigate the utility degradation caused by high sensitivity.
- We theoretically prove that the skip-gram can preserve arbitrary node proximities by carefully designing its negative sampling probabilities. This ensures that one can achieve structural preferences that align with the desired mining objectives.
- Extensive experimental results on several large-scale networks (e.g., social networks, citation networks) demonstrate that our proposed method outperforms various state-of-the-art methods across structural equivalence and link prediction tasks.

The outline of the paper is as follows. Section II introduces the preliminaries. In Section III-A, we define the problem and give a first-cut solution. Section IV explains our proposed SE-PrivGEmb, and its privacy and time complexity are discussed in Section V. The experimental results are presented in Section VI, while related work is reviewed in Section VII. Finally, our research findings are summarized in Section VIII.

TABLE I  
FREQUENTLY USED SYMBOLS

Symbol	Description
$\epsilon, \delta$	Privacy parameters
$G, \mathbf{A}$	Original graph and its adjacency matrix
$\mathcal{A}$	Randomized algorithm
$D_\alpha$	Rényi divergence of order $\alpha$
$ V ,  E $	Number of nodes and edges in $G$
$\mathbf{x}, \mathbf{y}$	Lowercase letters denoting vectors
$\mathbf{x} \cdot \mathbf{y}$	Inner product between two vectors
$\mathbf{X}, \mathbf{Y}$	Bold capital letters denoting matrices
$L$	Loss function
$\mathbf{W}_{in}, \mathbf{W}_{out}$	Embedding matrices of skip-gram
$k$	Negative sampling number
$r$	Dimension of low-dimensional vectors
$\mathbf{I}$	Identity matrix

## II. PRELIMINARIES

In this section, we provide an overview of graph embedding generation, DP, DPSGD, and node proximity, highlighting their key concepts and important properties. We also summarize the notations commonly used in this paper in Table I.

### A. Notations

We consider an undirected and unweighted graph  $G = (V, E)$ , where  $V$  is the set of nodes and  $E$  is the set of edges. For any two nodes  $v_i$  and  $v_j$  belonging to  $V$ ,  $(i, j) \in E$  represents an edge in  $G$ . The adjacency matrix of  $G$  is  $\mathbf{A}$ , where  $\mathbf{A}_{ij} = 1$  if  $(v_i, v_j) \in E$  and  $\mathbf{A}_{ij} = 0$  otherwise. The goal of graph embedding generation techniques is to learn a function  $f : V \rightarrow \mathbf{W}$ , where  $\mathbf{W} \in \mathbb{R}^{|V| \times r}$  with embedding dimension  $r \ll |V|$ , while preserving the inherent properties and structures of the original  $G$ . We denote  $\mathbf{v}_i = f(v_i)$  as the vector embedding of the node  $v_i$ .

### B. Graph Embedding Generation

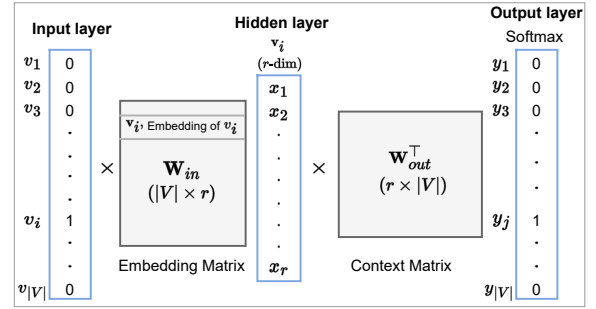


Fig. 1. Architecture of a Skip-gram Model. The embedding matrices  $\mathbf{W}_{in}$ , with a size of  $|V| \times r$ , and  $\mathbf{W}_{out}^T$ , with a size of  $r \times |V|$ , are two model parameters that require optimization. For any node pair  $(v_i, v_j)$ ,  $\mathbf{v}_i$  represents the vector for  $v_i$  in the input weight matrix  $\mathbf{W}_{in}$ , while  $\mathbf{v}_j$  represents the vector for  $v_j$  in the output weight matrix  $\mathbf{W}_{out}$ .

The skip-gram model is a neural network architecture that learns word embeddings by predicting the surrounding words given a target word. In the context of graph embedding, each node in the network can be thought of as a “word”, and the surrounding words are defined as the nodes that co-occur with the target node in the network. Inspired by this setting, DeepWalk [9] achieves graph embedding generation by treating the paths traversed by random walks over networks as sentences and using skip-gram (see Fig. 1) to learn latent representations of nodes. With the advent of DeepWalk, several skip-gram based graph embedding generation models have emerged, including LINE [10], PTE [11], and node2vec [12]. These methods aim at learning informative node representations to predict neighboring nodes. As a result, these methods share a common objective of maximizing the log-likelihood function, i.e.  $\max p(v_j | v_i)$ , and in machine learning, the convention is to minimize the cost function:

$$\min -p(v_j | v_i) = -\frac{\exp(\mathbf{v}_i \cdot \mathbf{v}_j)}{\sum_{v^n \in V} \exp(\mathbf{v}_i \cdot \mathbf{v}^n)}. \quad (1)$$

However, the objective of Eq. (1) is hard to optimize, due to the costly summation over all inner product with every vertex of the graph. To improve the training efficiency, we adopt the

skip-gram with negative sampling, which turns to optimize the following objective function for each observed  $(v_i, v_j)$  pair:

$$\begin{aligned} & \min L(v_i, v_j) \\ & = -\log \sigma(\mathbf{v}_j \cdot \mathbf{v}_i) - \sum_{n=1}^k \mathbb{E}_{v^n \sim \mathbb{P}_n(v)} [\log \sigma(-\mathbf{v}^n \cdot \mathbf{v}_i)], \end{aligned} \quad (2)$$

where  $k$  is the number of negative samples,  $\mathbf{v}_i \in \mathbf{W}_{in}$  is the central vector of  $v_i$ ,  $\mathbf{v}_j \in \mathbf{W}_{out}$  is the context vector of  $v_j$ ,  $\sigma(\mathbf{v}_i \cdot \mathbf{v}_j)$  is the normalized similarity of  $v_i$  and  $v_j$  under their representations  $\mathbf{v}_i$  and  $\mathbf{v}_j$  learned from the model,  $\sigma(\cdot) = \frac{1}{1+\exp(-x)}$  is the classic Sigmoid activation function.  $\mathbb{P}_n(v)$  is the noise distribution for negative sampling, which is carefully designed in Section IV-B of this paper to achieve provable structure preference.

### C. Differential Privacy

**$(\epsilon, \delta)$ -DP.** DP [14] has emerged as the de-facto standard notion in private data analysis. In general, it requires an algorithm to be insensitive to the changes in any individual's record. In the context of graph data, two graph datasets  $G$  and  $G'$  are considered as neighboring databases if they differ by one record (i.e., edge or node). Formally, DP for graph data is defined as follows.

**Definition 1** (Edge (Node)-Level DP [15]). *A graph analysis mechanism  $\mathcal{A}$  achieves edge (node)-level  $(\epsilon, \delta)$ -DP, if for any pair of input graphs  $G$  and  $G'$  that are neighbors (differ by at most one edge or node), and for all possible  $O \subseteq \text{Range}(\mathcal{A})$ , we have:*

$$\mathbb{P}[\mathcal{A}(G) \in O] \leq \exp(\epsilon) \cdot \mathbb{P}[\mathcal{A}(G') \in O] + \delta.$$

The concept of the neighboring dataset  $G, G'$  is categorized into two types. Specifically, if  $G'$  can be derived by replacing a single data instance in  $G$ , it is termed **bounded DP** [14]. If  $G'$  can be obtained by adding or removing a data sample from  $G$ , it is termed **unbounded DP** [16]. The parameter  $\epsilon$  represents the privacy budget, which determines the trade-off between privacy and utility in the algorithm. A smaller value of  $\epsilon$  indicates a higher level of privacy protection. The parameter  $\delta$  is typically chosen to be very small and is informally referred to as the failure probability.

**$(\alpha, \epsilon)$ -RDP.** In this paper, we use an alternative definition of DP, called RDP [17], which allows obtaining tighter sequential composition results:

**Definition 2** (RDP [17]). *Given  $\alpha > 1$  and  $\epsilon > 0$ , a randomized algorithm  $\mathcal{A}$  satisfies  $(\alpha, \epsilon)$ -RDP if for every neighboring datasets  $G$  and  $G'$ , we have  $D_\alpha(\mathcal{A}(G) \parallel \mathcal{A}(G')) \leq \epsilon$ , where  $D_\alpha(P \parallel Q)$  is the Rényi divergence of order  $\alpha$  between probability distributions  $P$  and  $Q$  defined as  $D_\alpha(P \parallel Q) = \frac{1}{\alpha-1} \log \mathbb{E}_{x \sim Q} \left[ \frac{P(x)}{Q(x)} \right]^\alpha$ .*

A crucial feature of RDP is that it can be transformed into standard  $(\epsilon, \delta)$ -DP using Proposition 3 from [17]. This conversion can be done in the following way:

**Theorem 1** (RDP conversion to  $(\epsilon, \delta)$ -DP [17]). *If  $\mathcal{A}$  is an  $(\alpha, \epsilon)$ -RDP algorithm, then it also satisfies  $(\epsilon + \frac{\log(1/\delta)}{\alpha-1}, \delta)$ -DP for any  $\delta \in (0, 1)$ .*

**Gaussian mechanism.** Suppose a function  $f$  maps a graph  $G$  to an  $r$ -dimensional output in  $\mathbb{R}^r$ . To create a differentially private mechanism of  $f$ , it is common practice to inject random noise into the output of  $f$ . The size of this noise is determined by the sensitivity of  $f$ , which is defined as follows.

**Definition 3** (Sensitivity [14]). *Given a function  $f : G \rightarrow \mathbb{R}^r$ , for any neighboring datasets  $G$  and  $G'$ , the  $\ell_2$ -sensitivity of  $f$  is defined as:*

$$S_f = \max_{G, G'} \|f(G) - f(G')\|_2.$$

The Gaussian mechanism is a commonly used approach to achieve RDP, whereby Gaussian noise is added to the output of an algorithm to protect user privacy. By adding Gaussian noise with variance  $\sigma^2$  to function  $f$ , such that  $\mathcal{A}(G) = f(G) + \mathcal{N}(\sigma^2 \mathbf{I})$ , we can obtain an  $(\alpha, \epsilon)$ -RDP algorithm for all  $\alpha > 1$ , where  $\epsilon = \frac{\alpha S_f^2}{2\sigma^2}$  [17].

**Remark 1.** *It is worth noting that the concept of sensitivity implies that satisfying node-level DP is challenging as varying one node could result in the removal of  $|V| - 1$  edges in the worst case. As a result, a significant amount of noise must be added to ensure privacy protection.*

**Important properties.** RDP has two crucial properties that are essential when designing intricate algorithms from simpler ones:

- **Sequential Composition:** When an  $(\alpha, \epsilon)$ -RDP algorithm is applied multiple times ( $m$  instances) on the same dataset, the resulting output will be at most  $(\alpha, m\epsilon)$ -RDP.
- **Robustness to Post-processing:** The  $(\alpha, \epsilon)$ -RDP property remains intact even after any post-processing steps are applied to the output of an  $(\alpha, \epsilon)$ -RDP algorithm.

### D. DPSGD

DPSGD [1] is an advanced training algorithm for deep learning models. **Its main purpose is to manage the issue of excessive splitting of the privacy budget during optimization.** In DPSGD, the gradient  $\mathbf{g}(x_i)$  is computed for each example  $x_i$  in a randomly selected batch of size  $B$ . The  $\ell_2$  norm of each gradient is then clipped using a threshold  $C$  to control the sensitivity of  $\mathbf{g}(x_i)$ . After clipping, the gradients are summed, and Gaussian noise  $\mathcal{N}(C^2 \sigma^2 \mathbf{I})$  is added to ensure privacy. The resulting noisy cumulative gradient  $\tilde{\mathbf{g}}$  is used to update the model parameters by taking its average. The expression for  $\tilde{\mathbf{g}}$  is as follows:

$$\tilde{\mathbf{g}} \leftarrow \frac{1}{B} \left( \sum_{i=1}^B \text{Clip}(\mathbf{g}(x_i)) + \mathcal{N}(C^2 \sigma^2 \mathbf{I}) \right), \quad (3)$$

in which  $\text{Clip}(\cdot)$  represents the clipping function defined as  $\text{Clip}(\mathbf{g}(x_i)) = \mathbf{g}(x_i) / \max(1, \frac{\|\mathbf{g}(x_i)\|_2}{C})$ .

### E. Node Proximity

Node proximity measures are commonly used to quantify the closeness or relationships between two or more nodes in a graph. A general definition of node proximity can be expressed as follows:

**Definition 4** (Node Proximity Matrix). *The node proximity matrix  $\mathbf{P}$  of a graph  $G$  is a  $|V| \times |V|$  matrix that quantifies the closeness between pairs of nodes. Each element  $p_{ij} \in \mathbf{P}$  reflects the proximity of nodes  $v_i$  and  $v_j$  based on their structural relationships in  $G$ . This proximity is defined as:*

$$p_{ij} = g(N(v_i), N(v_j), G), \quad (4)$$

where  $N(v_i)$  and  $N(v_j)$  are the neighbors of nodes  $v_i$  and  $v_j$ , and  $g(\cdot)$  is a function that applies various measures such as common neighbors to determine their proximity.

The function  $g(\cdot)$  can consist of first-order, second-order, or high-order features. First-order features, such as common neighbors and preferential attachment [18], consider only the one-hop neighbors of the target nodes. Second-order features, like Adamic-Adar and resource allocation [19], are based on two-hop neighborhoods. High-order heuristics, which require knowledge of the entire network, include Katz [20], PageRank [21], and random walk-based proximity [22].

## III. PROBLEM DESCRIPTION AND A FIRST-CUT SOLUTION

### A. Problem Description

**Threat Model.** In this paper, we consider a white-box attack [23], where the adversary possesses full information about the skip-gram model used for graph embedding, including its architecture and parameters. This means that the attacker has access to the published model but not the training data and training process. With the knowledge of some (or even all except the target) samples in the training dataset, the attacker aims to infer the existence of a target training data sample.

**Privacy Model.** In reality, data owners who have the ability to analyze graph structural features usually hope to be able to customize the extracted graph structural information. They aim to input the extracted features into a unified differentially private graph embedding generation model, with the goal of obtaining private low-dimensional node vectors that effectively support specific mining tasks. As stated in Section II-E, the node proximity can quantify structural features. **Given a graph  $G = (V, E)$  and an arbitrary node proximity  $p_{ij}$ , we aim to achieve differentially private graph embedding generation while preserving arbitrary structure preferences.** Formally, the definition of differentially private graph embedding generation is as follows:

**Definition 5** (Private Graph Embedding Generation under Bounded DP). *Let  $\Theta$  be a collection of the input and output weight matrix  $\{\mathbf{W}_{in}, \mathbf{W}_{out}\}$ , that is  $\Theta = \{\mathbf{W}_{in}, \mathbf{W}_{out}\}$ . A graph embedding generation model  $\mathcal{L} = \mathbb{E}_{(i,j) \in E} L(v_i, v_j)$  satisfies  $(\epsilon, \delta)$ -node-level DP under bounded constraints if two*

*neighboring graphs  $G$  and  $G'$ , which differ in only a node, and for all possible  $\Theta_S \subseteq \text{Range}(\mathcal{L})$ :*

$$\mathbb{P}(\mathcal{L}(G) \in \Theta_S) \leq \exp(\epsilon) \times \mathbb{P}(\mathcal{L}(G') \in \Theta_S) + \delta.$$

where  $\Theta_S$  denotes the set comprising all possible values of  $\Theta$ .

The definition of node-level private graph embedding generation is only applicable under bounded DP. Consider a pair of neighboring graphs  $G^* \subseteq G$  and  $G' \subseteq G$ , where  $G' = G^* \cup \{\text{node } z\}$ . If we use unbounded DP as the privacy definition, we face a challenge when attempting to express the following inequality:

$$\mathbb{P}(\mathcal{L}_z(G') \in \Theta_S) \leq \exp(\epsilon) \times \mathbb{P}(\mathcal{L}_z(G^*) \in \Theta_S) + \delta.$$

The issue arises because the node  $z$  is not included in  $G^*$ . Therefore, the node-level differentially private graph embedding generation is incompatible with the definition of unbounded DP.

Based on the robustness to post-processing, we can immediately conclude that the  $(\epsilon, \delta)$ -private graph embedding generation is robust to graph downstream tasks, as stated in the following theorem:

**Theorem 2.** *Let  $\mathcal{L}$  be an  $(\epsilon, \delta)$ -node-level private graph embedding generation model, and  $f$  be any graph downstream task whose input is the private graph embedding matrix (i.e.,  $\mathbf{W}_{in}$ ). Then,  $f \circ \mathcal{L}$  satisfies  $(\epsilon, \delta)$ -node-level DP.*

### B. A First-cut Solution

To establish structural preference settings in graph embedding generation, we design a novel objective function  $L_{nov}$  for each edge by redefining the objective function  $L$  in Eq. (2), as follows:

$$\begin{aligned} \min L_{nov}(v_i, v_j, p_{ij}) &= p_{ij} L(v_i, v_j), \\ &= -p_{ij} \log \sigma(\mathbf{v}_j \cdot \mathbf{v}_i) - p_{ij} \sum_{n=1}^k \mathbb{E}_{\mathbf{v}^n \sim \mathbb{P}_n(v)} [\log \sigma(-\mathbf{v}^n \cdot \mathbf{v}_i)], \end{aligned} \quad (5)$$

where  $p_{ij}$  denotes an arbitrary node proximity that quantifies structural features.

As stated in Section II-D, DPSGD with the advanced composition mechanism can manage the issue of excessive splitting of the privacy budget during optimization. One straightforward approach is to perturb the gradient on  $\mathbf{v}$ , where  $\mathbf{v}$  is a general notation representing either  $\mathbf{v}_i$  or  $\mathbf{v}_j$ . Formally, the noisy gradient  $\tilde{\nabla}_{\mathbf{v}} L_{nov}^B$  is expressed as follows:

$$\begin{aligned} &\tilde{\nabla}_{\mathbf{v}} L_{nov}^B(v_{ib}, v_{jb}, p_{ibjb}) \\ &= \frac{1}{B} \left( \sum_{b=1}^B \text{Clip} \left( \frac{\partial L_{nov}(v_{ib}, v_{jb}, p_{ibjb})}{\partial \mathbf{v}} \right) + \mathcal{N}(S_{\nabla_{\mathbf{v}}}^2 \sigma^2 \mathbf{I}) \right), \end{aligned} \quad (6)$$

where  $B$  denotes the size of batch sampling, and the upper bound of  $S_{\nabla_{\mathbf{v}}}$  is  $BC$  under node-level DP, with  $C$  being the clipping threshold. The main reason for significant noise is that in graph learning, individual examples no longer compute gradients independently.



**Limitation.** The approach discussed above, however, produces poor results. The root cause of the poor performance is the large sensitivity.

#### IV. OUR PROPOSAL: SE-PRIVGEMB

To overcome the above limitation, we present SE-PrivGEmb, a novel method for achieving differentially private graph embedding generation, which incorporates structural preferences while satisfying node-level RDP and high data utility. In particular, we deeply analyze the optimization of skip-gram and design a noise tolerance mechanism via perturbing non-zero vectors (see Section IV-A), which can effectively mitigate the utility degradation caused by high sensitivity. We theoretically prove that the skip-gram can preserve arbitrary node proximities by carefully designing its negative sampling probabilities (see Section IV-B). This ensures that one can achieve structural preferences that align with the desired mining objectives. The training algorithm is shown in Section IV-C.

##### A. Noise Tolerance via Perturbing Non-zero Vectors

For the gradient with respect to input weight matrix, that is  $\frac{\partial L_{nov}}{\partial \mathbf{W}_{in}}$ , it is equivalent to taking the derivative for the hidden layer (see Fig. 1). Specifically,

$$\frac{\partial L_{nov}}{\partial \mathbf{W}_{in}} = \frac{\partial L_{nov}}{\partial \mathbf{v}_i} = p_{ij} \sum_{n=0}^k (\sigma(\mathbf{v}^n \cdot \mathbf{v}_i) - \mathbb{I}_{v_j}[v^n]) \cdot \mathbf{v}^n, \quad (7)$$

since the input layer is a one-hot encoded vector. Here,  $\mathbb{I}_{v_j}[v^n]$  is an indicator function to indicate whether  $v^n$  is a positive node and when  $n = 0$ ,  $v^n = v_j$ .

For the gradient with respect to output weight matrix, that is

$$\frac{\partial L_{nov}}{\partial \mathbf{W}_{out}} = \frac{\partial L_{nov}}{\partial \mathbf{v}^n} = p_{ij} (\sigma(\mathbf{v}^n \cdot \mathbf{v}_i) - \mathbb{I}_{v_j}[v^n]) \cdot \mathbf{v}_i, \quad (8)$$

with negative sampling, only a fraction of the node vectors in  $\mathbf{W}_{out}$  are updated. Gradients for  $k + 1$  node vectors, representing both positive and negative nodes in  $\mathbf{W}_{out}$ , are computed using Eq. (8).

This encourages us to inject noise exclusively into non-zero vectors within the gradients of  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$  to improve utility. As outlined in Section III-B, utilizing  $\mathbf{v}$  as a general notation representing either  $\mathbf{v}_i$  or  $\mathbf{v}_j$ , we have:

$$\begin{aligned} & \tilde{\nabla}_{\mathbf{v}} L_{nov}^B(v_{i_b}, v_{j_b}, p_{i_b j_b}) \\ &= \frac{1}{B} \left( \sum_{b=1}^B \text{Clip} \left( \frac{\partial L_{nov}(v_{i_b}, v_{j_b}, p_{i_b j_b})}{\partial \mathbf{v}} \right) + \tilde{\mathcal{N}}(S_{\nabla_{\mathbf{v}}}^2 \sigma^2 \mathbf{I}) \right), \end{aligned} \quad (9)$$

where  $\tilde{\mathcal{N}}(\cdot)$  denotes the noise matrix that selectively adds noise to non-zero vectors in the gradient. Taking  $\mathbf{W}_{in}$  as an example, the resulting  $\mathbf{W}_{in}$  using Eq. (9) is illustrated in Fig. 2(d) after the gradient update.

**Remark 2.** As stated in Definition 5, we only publish the final graph embedding matrices that satisfy DP after iterative optimization. Therefore, in this scenario, the gradient updates of each iteration in Fig. 2(b) are not published, so the attackers do not reveal information about which edges are sampled.

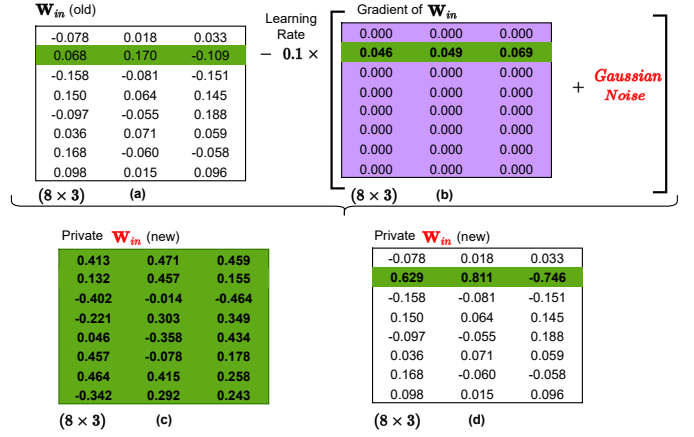


Fig. 2. An Illustration of Private Update for  $\mathbf{W}_{in}$ . (a) represents the original input weight, (b) represents the gradient of the input weight, while (c) and (d) depict the perturbed input weight using Eq. (6) and Eq. (9), respectively.

##### B. Theoretical Guarantee on Structure Preference

In this section, by carefully designing the negative sampling probability  $\mathbb{P}_n(v)$ , we theoretically prove that the skip-gram objective can preserve arbitrary node proximities that quantify structural features.

**Theorem 3.** Consider a node proximity matrix  $\mathbf{P}$  for a given graph  $G$ , with its elements denoted as  $p_{ij} \in \mathbf{P}$ . Let the negative sampling probability  $\mathbb{P}_n(v)$  in objective (5) be proportional to  $\frac{\min(\mathbf{P})}{\sum_{v_j \in V} p_{ij}}$ , where  $\sum_{v_j \in V} p_{ij}$  represents the sum of proximity values associated with node  $v_i$  across all the nodes in the graph, and

$$\min(\mathbf{P}) = \min\{p_{ij} | p_{ij} > 0, \forall i, j\}.$$

Let  $x_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j$  in Eq. (5). We can derive the theoretical optimal solution in Eq. (5) with respect to  $x_{ij}$  as follows:

$$x_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j = \log \left( \frac{p_{ij}}{k \min(\mathbf{P})} \right), \quad (10)$$

where  $k$  and  $\min(\mathbf{P})^1$  are both constants. According to Eq. (10), we observe that  $x_{ij}$  preserves the logarithmic value of  $p_{ij}$  with a constant shift.

*Proof.* For ease of understanding, we first show the expected objective function of skip-gram as follows:

$$\min_{\mathbf{v}_i, \mathbf{v}_j} \mathcal{L} = \mathbb{E}_{(i,j) \in E} L_{nov}(v_i, v_j, p_{ij}). \quad (11)$$

Consider an arbitrary proximity matrix  $\mathbf{P}$  of a graph, where  $p_{ij} \in \mathbf{P}$ . We rewrite the objective (11) as follows:

<sup>1</sup> $\min(\mathbf{P})$  is a constant for a given graph dataset, which is derived from precomputing its proximity matrix (see Line 1 in Algorithm 2).  $\min(\mathbf{P})$  can be substituted with another constant, such as  $\mathcal{O}(\min(\mathbf{P}))$ , provided that the probability  $\frac{\mathcal{O}(\min(\mathbf{P}))}{\sum_{v_j \in V} p_{ij}} \in (0, 1)$ .

$$\begin{aligned}
\min_{\mathbf{v}_i, \mathbf{v}_j} \mathcal{L} &= \sum_{v_i \in V} \sum_{v_j \in V} p_{ij} \cdot \left( -\log \sigma(\mathbf{v}_j \cdot \mathbf{v}_i) \right. \\
&\quad \left. - k \cdot \mathbb{E}_{v^n \sim \mathbb{P}_n(v)} \log \sigma(-\mathbf{v}_n \cdot \mathbf{v}_i) \right) \\
&= - \sum_{v_i \in V} \sum_{v_j \in V} p_{ij} \cdot \log \sigma(\mathbf{v}_j \cdot \mathbf{v}_i) \\
&\quad - \sum_{v_i \in V} \sum_{v_j \in V} p_{ij} \cdot (k \cdot \mathbb{E}_{v^n \sim \mathbb{P}_n(v)} \log \sigma(-\mathbf{v}_n \cdot \mathbf{v}_i)) \\
&= - \sum_{v_i \in V} \sum_{v_j \in V} p_{ij} \cdot \log \sigma(\mathbf{v}_j \cdot \mathbf{v}_i) - \\
&\quad \sum_{v_i \in V} \left( \sum_{v_j \in V} p_{ij} \right) \cdot k \cdot \mathbb{E}_{v^n \sim \mathbb{P}_n(v)} \log \sigma(-\mathbf{v}_n \cdot \mathbf{v}_i).
\end{aligned}$$

In this context, we define  $\mathbb{P}_n(v)$  to be proportional to  $\frac{\min(\mathbf{P})}{\sum_{v_j \in V} p_{ij}}$ , and can explicitly express the expectation term as:

$$\begin{aligned}
&\mathbb{E}_{v^n \sim \mathbb{P}_n(v)} \log \sigma(-\mathbf{v}_n \cdot \mathbf{v}_i) \\
&= \sum_{v^n \in V} \frac{\min(\mathbf{P})}{\sum_{v_j \in V} p_{ij}} \log \sigma(-\mathbf{v}_n \cdot \mathbf{v}_i).
\end{aligned} \tag{12}$$

With Eq. (12), we have:

$$\begin{aligned}
\min_{\mathbf{v}_i, \mathbf{v}_j} \mathcal{L} &= - \sum_{v_i \in V} \sum_{v_j \in V} p_{ij} \cdot \log \sigma(\mathbf{v}_j \cdot \mathbf{v}_i) \\
&\quad - \sum_{v_i \in V} \sum_{v_j \in V} k \cdot \min(\mathbf{P}) \cdot \log \sigma(-\mathbf{v}_j \cdot \mathbf{v}_i).
\end{aligned} \tag{13}$$

To minimize Eq. (13), we set the partial derivative of each independent variable  $x_{ij} = \mathbf{v}_j \cdot \mathbf{v}_i$  to zero:

$$\frac{\partial \mathcal{L}}{\partial x_{ij}} = \sum_{v_i \in V} \sum_{v_j \in V} ((p_{ij} + k \cdot \min(\mathbf{P})) \cdot \sigma(x_{ij}) - p_{ij}) = 0.$$

This leads to the following expression:

$$x_{ij} = \log \left( \frac{p_{ij}}{k \min(\mathbf{P})} \right),$$

which completes the proof.  $\square$

In Theorem 3,  $x_{ij}$  represents the inner product of  $v_i$  and  $v_j$  to estimate the proximity of node pairs  $(v_i, v_j)$ . From the optimal solution in Eq. (10), we observe that  $x_{ij}$  preserves the logarithmic value of  $p_{ij}$  with a constant shift, indicating that our method maintains the proximity between any two nodes.

**Comparison with Prior Works.** In what follows, we also compare our findings with the optimal embedding presented in prior research to better reflect our contribution.

According to previous research [13], [24], the expectation term within the objective (11) can be expressed as:

$$\mathbb{E}_{v^n \sim \mathbb{P}_n(v)} [\log \sigma(-\mathbf{v}^n \cdot \mathbf{v}_i)] = \sum_{v^n \in V} \frac{d_{v^n}}{\mathcal{D}} \log \sigma(-\mathbf{v}^n \cdot \mathbf{v}_i), \tag{14}$$

where  $\mathbb{P}_n(v) \propto \frac{d_v^n}{|E|}$  is used for negative sampling,  $d_v$  represents the degree of node  $v$ , and  $\mathcal{D} = \sum_{ij} p_{ij}$ . By fusing Eq. (14), the theoretical optimal solution of the objective (11) is:

$$\mathbf{v}_j \cdot \mathbf{v}_i = \log \left( \frac{p_{ij} \mathcal{D}}{d_i d_j} \right) - \log(k). \tag{15}$$

The optimal embedding in Eq. (15) fails to preserve the exact relationships for specified structural information within the embedding space.

### C. Training Algorithm

The overall training algorithm of SE-PrivGEmb is shown in Algorithm 2. We compute the node proximity matrix for a graph  $G$  (line 1), and divide this graph into sets of disjoint subgraphs  $G_S = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_{|E|}\}$  (line 2), where each  $\mathcal{S}_i$  obtained through Algorithm 1 consists of an edge and its corresponding negative samples<sup>2</sup>. We initialize the weight matrices  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$  (line 3). In each epoch, we sample  $B$  subgraphs uniformly at random according to Algorithm 1 (line 5). Next, it updates  $\mathbf{W}_{in}$  according to  $\nabla_{\mathbf{v}_i} L_{nov}^B$  (line 6), and updates  $\mathbf{W}_{out}$  according to  $\nabla_{\mathbf{v}_j} L_{nov}^B$  (line 7). Throughout the training process, the algorithm also updates the overall privacy cost  $\epsilon$  to ensure it does not exceed the target privacy budget (lines 8-10). Finally, after completing all epochs, the algorithm returns differentially private  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$ .

---

#### Algorithm 1: Generating Disjoint Subgraphs

---

**Input:** graph  $G$ .  
**Output:** Set of subgraphs,  $G_S$ .

```

1 Set  $G_S = \{\}$  and  $\mathcal{S} = \{\}$ ;
2 for  $(i, j) \in E$  do
3   Assign  $(v_i, v_j)$  to  $\mathcal{S}$ ;
4   for  $n = 1, 2, \dots, k$  do
5     while True do
6        $v_n \leftarrow$  randomly sample one node from  $V$ ;
7       if  $(v_i, v_n)$  is not in  $E$  then
8         Break;
9     end
10    end
11    Assign  $(v_i, v_n)$  to  $\mathcal{S}$ ;
12  end
13  Assign  $\mathcal{S}$  to  $G_S$ ;
14 end
15 return  $G_S$ ;

```

---

## V. PRIVACY AND COMPLEXITY ANALYSIS

### A. Privacy Analysis

Subsampling introduces a non-zero probability of an added or modified sample not to be processed by the randomized

<sup>2</sup>To facilitate privacy analysis in Section V-A, we collect positive and negative samples prior to training. An alternative solution is to implement Algorithm 1 during the training of Algorithm 2. However, the sampling probability needed for privacy amplification involves more complex analysis, which we will address in future work.

---

**Algorithm 2:** SE-PrivGEmb Algorithm

---

**Input:** privacy parameters  $\epsilon$  and  $\delta$ , standard deviation  $\sigma$ , learning rate  $\eta$ , embedding dimension  $r$ , negative sampling number  $k$ , batch size  $B$ , gradient clipping threshold  $C$ , number of maximum training epochs  $n^{epoch}$ .

**Output:** Differentially Private  $\mathbf{W}_{in}$ ,  $\mathbf{W}_{out}$ .

```

1 Compute the node proximity matrix for a graph  $G$ ;
2 Divide the graph  $G$  into sets of disjoint subgraphs
    $G_S = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_{|E|}\}$  based on Algorithm 1;
3 Initialize the weight matrices  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$ ;
4 for  $epoch = 0$ ;  $epoch < n^{epoch}$  do
    // Generate samples
5   Sample  $B$  subgraphs uniformly at random from  $G_S$ ;
    // Optimize weights
6   Update  $\mathbf{W}_{in}$  according to  $\tilde{\nabla}_{\mathbf{v}_i} L_{nov}^B$  in Eq. (9);
7   Update  $\mathbf{W}_{out}$  according to  $\tilde{\nabla}_{\mathbf{v}_j} L_{nov}^B$  in Eq. (9);
    // Update privacy accountant of RDP
8   Calculate RDP with sampling probability  $\frac{B}{|E|}$ ;
9    $\hat{\delta} \leftarrow$  get privacy spent given the target  $\epsilon$ ;
10  Stop optimization if  $\hat{\delta} \geq \delta$ ;
11 end
12 return  $\mathbf{W}_{in}$ ,  $\mathbf{W}_{out}$ ;
```

---

algorithm. Random sampling will enhance privacy protection and reduce privacy loss [25]–[27]. In this paper, we focus on the “subsampling without replacement” setup, which adheres to the following privacy amplification theorem for  $(\epsilon, \delta)$ -DP.

**Definition 6** (Subsample [25]). *Given a dataset  $X$  containing  $n$  points, the subsample procedure selects a random sample from the uniform distribution over all subsets of  $X$  with size  $m$ . The ratio  $\gamma = m/n$  is termed as the sampling parameter of the subsample procedure.*

**Theorem 4** (RDP for Subsampled Mechanisms [25]). *Given a dataset of  $n$  points drawn from a domain  $\mathcal{X}$  and a mechanism  $\mathcal{A}$  that accepts inputs from  $\mathcal{X}^m$  for  $m \leq n$ , we consider the randomized algorithm  $\mathcal{A}$  for subsampling, which is defined as follows: (1) sample  $m$  data points without replacement from the dataset, where the sampling parameter is  $\gamma = m/n$ , and (2) apply  $\mathcal{A}$  to the subsampled dataset. For all integers  $\alpha \geq 2$ , if  $\mathcal{A}$  satisfies  $(\alpha, \epsilon(\alpha))$  RDP, then the subsampled mechanism  $\mathcal{A} \circ \text{subsample}$  satisfies  $(\alpha, \epsilon'(\alpha))$ -RDP in which*

$$\begin{aligned} \epsilon'(\alpha) \leq & \frac{1}{\alpha - 1} \log \left( 1 + \gamma^2 \binom{\alpha}{2} \min \left\{ 4(e^{\epsilon(2)} - 1), \right. \right. \\ & \left. \left. e^{\epsilon(2)} \min \left\{ 2, (e^{\epsilon(\infty)} - 1)^2 \right\} \right\} \right) \\ & + \sum_{j=3}^{\alpha} \gamma^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min \left\{ 2, (e^{\epsilon(\infty)} - 1)^j \right\}. \end{aligned}$$

Following Theorem 4, we conduct a privacy analysis of the generated  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$  in Algorithm 2. We adopt the functional perspective of RDP, where  $\epsilon$  is a function of

$\alpha$ , with  $1 < \alpha < \infty$ , and this function is determined by the private algorithm. For ease of presentation, we replace  $(\alpha, \epsilon'(\alpha))$  with  $(\alpha, \epsilon^\gamma(\alpha))$  in the following proof, where  $\gamma$  denotes the sampling probability.

**Theorem 5.** *Given the number of batch size  $B$ , the generated  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$  in Algorithm 2 satisfy node-level  $(\alpha, n^{epoch} \epsilon^{\frac{B}{|E|}}(\alpha))$ -RDP after  $n^{epoch}$  iterations.*

*Proof.* Without considering privacy amplification, every iteration  $epoch$  of Algorithm 2 is  $(\alpha, \epsilon(\alpha))$ -RDP where  $\epsilon(\alpha) = \frac{\alpha \cdot S_{\nabla}^2}{2\sigma^2}$ , which can be directly derived from Corollary 3 in [17]. Since we take  $B$  subgraphs uniformly at random from  $G_S$ , by setting  $\gamma$  to  $\frac{B}{|E|}$ , the privacy guarantee of Algorithm 2 can be directly analyzed by Theorem 4, yielding a RDP of  $(\alpha, \epsilon^\gamma(\alpha))$  with  $\epsilon^\gamma(\alpha)$  defined in Eq. (4). Finally, we apply the conversion rule in Theorem 1 to convert the RDP back to DP.  $\square$

### B. Complexity Analysis

We analyze the computational complexity of each step of SE-PrivGEmb. DeepWalk proximity [22] and node degree proximity are used in experiments. Computing DeepWalk proximity [22] takes  $\mathcal{O}(|V|^2)$  time, and computing node degree proximity takes  $\mathcal{O}(|V|)$  time. The time complexity of dividing the graph  $G$  into sets of subgraphs is  $|E|k$ . Initializing the weight matrices  $\mathbf{W}_{in}$  and  $\mathbf{W}_{out}$  is constant time  $\mathcal{O}(1)$ . The time complexity of updating  $\mathbf{W}_{in}$  depends on the embedding dimension  $r$  and the number of batch samples, which can be approximated as  $\mathcal{O}(rB)$ . Similarly, the time complexity of updating  $\mathbf{W}_{out}$  can be approximated as  $\mathcal{O}(rB)$ . The time complexity of updating the DP cost using RDP depends on the specific implementation of RDP. Different versions of RDP may result in slight differences in time complexity, but according to [28], these implementations all have asymptotic complexity of  $\mathcal{O}(rB\gamma)$ , where  $\gamma$  denotes the sampling probability. After  $n^{epoch}$  epochs, the total time complexity is  $\mathcal{O}(|V|^2 + |E|k + n^{epoch}rB + n^{epoch}rB\gamma)$  when DeepWalk proximity is used. Alternatively, when node degree proximity is used, the total time complexity becomes  $\mathcal{O}(|V| + |E|k + n^{epoch}rB + n^{epoch}rB\gamma)$ .

## VI. EXPERIMENTS

In this section, we evaluate the performance of SE-PrivGEmb in three downstream tasks: structural equivalence, link prediction, and node clustering. Structural equivalence refers to a concept wherein two nodes within a graph are deemed structurally equivalent if they possess identical connections with the same nodes, which is widely acknowledged as the simplest and most rigorous metric [29]. Link prediction is a commonly used benchmark task in graph learning models, which is used to further display the performance of SE-PrivGEmb. Node clustering evaluates an algorithm’s ability to identify and group nodes into meaningful clusters based on their similarities, assessing how well it discovers inherent group structures. Our specific goal is to answer the following five questions:

TABLE II  
STATISTICS OF DATASETS

Datasets	Nodes	Edges	Labels	Type
Chameleon	2,277	31,421	-	Web page
PPI	3,890	76,584	50	Protein
Power	4,941	6,594	-	Electrical grid
Arxiv	5,242	14,496	-	Collaboration
BlogCatalog	10,312	333,983	39	Social
DBLP	2,244,021	4,354,534	-	Citation

- How much do the parameters impact SE-PrivGEmb’s performance? (See Section VI-B)
- How much do the perturbation strategies impact SE-PrivGEmb’s performance? (See Section VI-C)
- How much does the privacy budget affect SE-PrivGEmb’s performance in structural equivalence? (See Section VI-D)
- How much does the privacy budget affect SE-PrivGEmb’s performance in link prediction? (See Section VI-E)
- How much does the privacy budget affect SE-PrivGEmb’s performance in node clustering? (See Section VI-F)

#### A. Experimental Setup

**Datasets.** We run experiments on six real-world datasets, Chameleon, PPI, Power, Arxiv, BlogCatalog, and DBLP. Table II shows the basic information of the datasets. Since we focus on simple graphs in this work, all datasets are pre-processed to remove self-loops. The details of the datasets are as follows.

- **Chameleon**<sup>3</sup>. The dataset is collected from the English Wikipedia on the chameleon topic, where nodes represent articles and edges indicate mutual links between them.
- **PPI** [30]. This is a human Protein-Protein Interaction network where nodes represent proteins and edges indicate interactions between these proteins.
- **Power**<sup>4</sup>. This dataset represents an electrical grid that stretches across the western United States. The nodes represent the buses within the grid, and the edges signify the transmission lines that connect these buses.
- **Arxiv**<sup>5</sup>. This network is derived from the e-print arXiv and specifically examines scientific collaborations among authors who submit papers to the Astrophysics category.
- **BlogCatalog**<sup>6</sup>. This network visualizes the intricate web of social interactions among online bloggers, where nodes represent individual bloggers and edges denote the connections or relationships between them.
- **DBLP**<sup>7</sup>. This dataset illustrates a scholarly network in which nodes represent papers, authors, and venues, while edges indicate authorship relationships and the publication venues of the papers.

**Evaluation Metrics.** we measure the performance of SE-PrivGEmb by examining its effectiveness in two downstream tasks: structural equivalence and link prediction.

- For structural equivalence, two nodes are considered to be structurally equivalent if they have many of the same neighbors. To evaluate the ability of an embedding method to recover structural equivalence, we introduce a distance metric. The distance  $dist(\mathbf{A}_i, \mathbf{A}_j)$  is defined as the difference between the lines of the adjacency matrix for each pair of nodes,  $(v_i, v_j)$ , whereas  $dist(\mathbf{Y}_i, \mathbf{Y}_j)$  represents the distance between their corresponding node vectors in the embedding space. To quantify structural equivalence, we calculate the correlation coefficient (i.e., Pearson) between these values for all node pairs. In this paper, we define  $StrucEqu = pearson(dist(\mathbf{A}_i, \mathbf{A}_j), dist(\mathbf{Y}_i, \mathbf{Y}_j))$ , where the Euclidean distance is used.
- For the link prediction task, as with [31], the existing links in each dataset are randomly divided into a training set (90%) and a test set (10%). To evaluate the performance of link prediction, we randomly select an equal number of node pairs without connected edges as negative test links for the test set. Additionally, for the training set, we sample the same number of node pairs without edges to construct negative training data. The performance metric is the area under the ROC curve (AUC).
- For the node clustering task, we feed the embedding vectors generated by each algorithm into a node clustering algorithm. Following [32], we adopt the Affinity Propagation algorithm [33] as the clustering method and evaluate the clustering results in terms of mutual-information (MI).

We measure each result over ten experiments and report the average value. The larger the values of **StrucEqu**, **AUC**, and **MI**, the better the utility.

**Competitive Methods.** In this paper, we design two PrivEmb methods fusing DeepWalk proximity [22] and node degree proximity: **SE-PrivGEmb<sub>DW</sub>** and **SE-PrivGEmb<sub>Deg</sub>**.

To establish a baseline for comparison, we utilize four state-of-the-art private graph learning methods, namely **DPG-GAN** [2], **DPGVAE** [2], **GAP** [6], and **ProGAP** [7]. Moreover, we also use **SE-GEmb<sub>DW</sub>** and **SE-GEmb<sub>Deg</sub>** as non-private counterparts to **SE-PrivGEmb<sub>DW</sub>** and **SE-PrivGEmb<sub>Deg</sub>**, respectively, to better show the performance of SE-PrivGEmb. In this study, we simulate a scenario where the graphs only contain structural information, while **GAP** and **ProGAP** rely on node features. To ensure a fair evaluation, similar to prior research [34], we use randomly generated features as inputs for both methods. Note that skip-gram incorporates two vectors for each node, namely a context (output) vector and a node (input) vector. However, during our testing phase, we do not observe any performance improvement by utilizing both vectors together. We only employ the node vectors of skip-gram for our experiments, similar to previous methods [10],

<sup>3</sup><https://snap.stanford.edu/data/wikipedia-article-networks.html>

<sup>4</sup><http://konect.cc/networks/opsahl-powergrid/>

<sup>5</sup><https://snap.stanford.edu/data/ca-GrQc.html>

<sup>6</sup><http://datasets.syr.edu/datasets/BlogCatalog3.html>

<sup>7</sup><https://www.aminer.cn/citation>



[35].

**Parameter Settings.** To demonstrate the performance of our proposed method across different downstream tasks, we set the same parameters for structural equivalence and link prediction except for the training epochs. Specifically, we set the training epochs to  $n^{epoch} = 200$  for structural equivalence and  $n^{epoch} = 2000$  for link prediction. The embedding dimension is fixed at  $r = 128$ . It is worth noting that we do not specifically highlight the impact of  $r$  as it is commonly used in various network embedding methods [9], [10], [36], [37]. For the privacy parameters, as with [2], we fix  $\delta = 10^{-5}$  and  $\sigma = 5$ , and vary the privacy budget  $\epsilon$  among the values  $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$  for the private methods. Additionally, we vary the batch size  $B$ , learning rate  $\eta$ , and the parameters  $C$  and  $k$  to verify their impact on the utility of SE-PrivGEmb. To ensure consistency with the original papers, we utilize the official GitHub implementations for DPGGAN, DPGVAE, **GAP** and **ProGAP**, and replicate the experimental setups described in those papers.

### B. Impact of Parameters

In this section, we vary the batch size  $B$ , learning rate  $\eta$ , and the parameters  $C$  and  $k$  to determine suitable values for the structural equivalence task. Given the privacy budget  $\epsilon = 3.5$ , our experiments are conducted on three datasets: Chameleon, Power, and Arxiv.

1) **Parameter  $B$ :** In this experiment, we investigate the impact of the parameter  $B$  in Table III. For SE-PrivGEmb<sub>DW</sub>, the results indicate that for the Chameleon dataset,  $B = 64$  is optimal, achieving the highest StrucEqu value of 0.4599 with a relatively small standard deviation (SD) value of 0.0530. In the case of the Power dataset,  $B = 128$  is recommended, as it produces a higher StrucEqu value of 0.2522, with a comparable SD value of 0.0543. Similarly, on the Arxiv dataset,  $B = 128$  performs well, yielding a StrucEqu value of 0.3457 and an SD value of 0.0303. For SE-PrivGEmb<sub>Deg</sub>, we can observe that on the Chameleon dataset,  $B = 1024$  emerges as the best choice, achieving a StrucEqu value of 0.3967 and an SD value of 0.0535. On the Power dataset,  $B = 128$  yields a StrucEqu value of 0.2322 and an SD value of 0.0476. On the Arxiv dataset,  $B = 128$  remains suitable, achieving a StrucEqu value of 0.2341 and an SD value of 0.0193. In summary, for both SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub>,  $B = 128$  consistently proves to be a suitable choice. Therefore, we set the batch size  $B$  to 128 in the subsequent experiments.

2) **Parameter  $\eta$ :** In this experiment, we investigate the impact of the learning rate  $\eta$ , as summarized in Table IV. For SE-PrivGEmb<sub>DW</sub>, on the Chameleon dataset,  $\eta = 0.15$  achieves the highest StrucEqu value of 0.4573, with a small SD value of 0.0072. The optimal learning rate for the Power dataset is  $\eta = 0.1$ , which yields a StrucEqu value of 0.2522 and an SD value of 0.0543.  $\eta = 0.25$  performs well on the Arxiv dataset, with a StrucEqu value of 0.4353 and an SD value of 0.0198. For SE-PrivGEmb<sub>Deg</sub>, on the Chameleon dataset,  $\eta = 0.1$  is identified as the best choice, achieving a StrucEqu value of 0.3661 and an SD value of 0.0092. On the Power

TABLE III  
SUMMARY OF STRUCEQU VALUES WITH DIFFERENT  $B$ , GIVEN  $\epsilon = 3.5$   
(RESULT: AVERAGE STRUCEQU  $\pm$  SD; **BOLD**: BEST)

$B$	SE-PrivGEmb <sub>DW</sub>		
	Chameleon	Power	Arxiv
32	0.4281 $\pm$ 0.0529	0.1549 $\pm$ 0.0196	0.3375 $\pm$ 0.0640
64	<b>0.4599<math>\pm</math>0.0530</b>	0.1815 $\pm$ 0.0327	0.3376 $\pm$ 0.0871
128	0.4507 $\pm$ 0.0341	<b>0.2522<math>\pm</math>0.0543</b>	<b>0.3457<math>\pm</math>0.0303</b>
256	0.3701 $\pm$ 0.0181	0.2053 $\pm$ 0.0670	0.3326 $\pm$ 0.0564
512	0.3293 $\pm$ 0.0324	0.2462 $\pm$ 0.0508	0.3411 $\pm$ 0.0031
1024	0.2780 $\pm$ 0.0406	0.2520 $\pm$ 0.0157	0.3416 $\pm$ 0.0161

$B$	SE-PrivGEmb <sub>Deg</sub>		
	Chameleon	Power	Arxiv
32	0.2611 $\pm$ 0.0714	0.1127 $\pm$ 0.0124	0.1394 $\pm$ 0.0815
64	0.3084 $\pm$ 0.0172	0.1216 $\pm$ 0.0581	0.1625 $\pm$ 0.0722
128	0.3661 $\pm$ 0.0092	<b>0.2322<math>\pm</math>0.0476</b>	<b>0.2341<math>\pm</math>0.0193</b>
256	0.3956 $\pm$ 0.0090	0.1595 $\pm$ 0.0418	0.2260 $\pm$ 0.0292
512	0.3710 $\pm$ 0.0834	0.1833 $\pm$ 0.0303	0.2204 $\pm$ 0.0656
1024	<b>0.3967<math>\pm</math>0.0535</b>	0.1773 $\pm$ 0.0317	0.2251 $\pm$ 0.0509

TABLE IV  
SUMMARY OF STRUCEQU VALUES WITH DIFFERENT  $\eta$ , GIVEN  $\epsilon = 3.5$   
(RESULT: AVERAGE STRUCEQU  $\pm$  SD; **BOLD**: BEST)

$\eta$	SE-PrivGEmb <sub>DW</sub>		
	Chameleon	Power	Arxiv
0.01	0.0703 $\pm$ 0.0378	0.0515 $\pm$ 0.0180	0.0436 $\pm$ 0.0121
0.05	0.4290 $\pm$ 0.0045	0.1756 $\pm$ 0.0241	0.3989 $\pm$ 0.0311
0.1	0.4507 $\pm$ 0.0341	<b>0.2522<math>\pm</math>0.0543</b>	0.3457 $\pm$ 0.0303
0.15	<b>0.4573<math>\pm</math>0.0072</b>	0.2431 $\pm$ 0.0198	0.3938 $\pm$ 0.0204
0.2	0.4152 $\pm$ 0.0470	0.1912 $\pm$ 0.0070	0.4282 $\pm$ 0.0285
0.25	0.4564 $\pm$ 0.0112	0.2478 $\pm$ 0.0546	<b>0.4353<math>\pm</math>0.0198</b>
0.3	0.4524 $\pm$ 0.0262	0.2213 $\pm$ 0.0342	0.4011 $\pm$ 0.0154

$\eta$	SE-PrivGEmb <sub>Deg</sub>		
	Chameleon	Power	Arxiv
0.01	0.0409 $\pm$ 0.0148	0.0266 $\pm$ 0.0331	0.0122 $\pm$ 0.0292
0.05	0.3221 $\pm$ 0.0419	0.1524 $\pm$ 0.0531	0.1876 $\pm$ 0.0445
0.1	<b>0.3661<math>\pm</math>0.0092</b>	<b>0.2322<math>\pm</math>0.0476</b>	<b>0.2341<math>\pm</math>0.0193</b>
0.15	0.3503 $\pm$ 0.0632	0.1742 $\pm$ 0.0598	0.1725 $\pm$ 0.0320
0.2	0.3169 $\pm$ 0.0529	0.1372 $\pm$ 0.0723	0.2189 $\pm$ 0.0445
0.25	0.3656 $\pm$ 0.0552	0.1435 $\pm$ 0.0595	0.2335 $\pm$ 0.0464
0.3	0.2628 $\pm$ 0.0471	0.1229 $\pm$ 0.0085	0.2086 $\pm$ 0.0765

dataset,  $\eta = 0.1$  is recommended, resulting in a StrucEqu value of 0.2322 and an SD value of 0.0476. On the Arxiv dataset,  $\eta = 0.1$  remains suitable, achieving a StrucEqu value of 0.2341, with an SD value of 0.0193. Overall, across both SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub>,  $\eta = 0.1$  emerges as a consistently suitable choice. It consistently yields higher StrucEqu values across datasets while maintaining relatively small SD values.

3) **Parameter  $C$ :** In this experiment, we investigate the impact of the gradient clipping threshold  $C$ , as shown in Table V. For SE-PrivGEmb<sub>DW</sub>, on the Chameleon dataset, the highest StrucEqu value of 0.4507 is achieved at  $C = 2$ , with a relatively low SD value of 0.0341, indicating stable results. Other strong contenders are  $C = 3$  and  $C = 4$ , though their StrucEqu values are slightly lower than those at  $C = 2$ . On the Power dataset, the highest StrucEqu value of 0.2522 appears at  $C = 2$ , with an SD value of 0.0543, showing superior performance at this setting. While  $C = 1$  and  $C = 4$  also exhibit relatively high StrucEqu values,  $C = 2$  remains notably preferable. On the Arxiv dataset,  $C = 1$  and  $C = 5$

TABLE V

SUMMARY OF STRUCEQU VALUES WITH DIFFERENT  $C$ , GIVEN  $\epsilon = 3.5$   
(RESULT: AVERAGE STRUCEQU  $\pm$  SD; **BOLD**: BEST)

$C$	SE-PrivGEmb <sub>DW</sub>		
	Chameleon	Power	Arxiv
1	0.3248 $\pm$ 0.0595	0.2216 $\pm$ 0.0281	<b>0.4266<math>\pm</math>0.0588</b>
2	<b>0.4507<math>\pm</math>0.0341</b>	<b>0.2522<math>\pm</math>0.0543</b>	0.3457 $\pm$ 0.0303
3	0.4319 $\pm$ 0.0475	0.1774 $\pm$ 0.0407	0.3514 $\pm$ 0.0219
4	0.4368 $\pm$ 0.0214	0.1972 $\pm$ 0.0414	0.3555 $\pm$ 0.0142
5	0.4079 $\pm$ 0.0172	0.1652 $\pm$ 0.0345	0.3671 $\pm$ 0.0352
6	0.3719 $\pm$ 0.0328	0.1710 $\pm$ 0.0557	0.3476 $\pm$ 0.0305
$C$	SE-PrivGEmb <sub>Deg</sub>		
	Chameleon	Power	Arxiv
1	0.2485 $\pm$ 0.0757	0.2030 $\pm$ 0.0846	0.2206 $\pm$ 0.0858
2	<b>0.3661<math>\pm</math>0.0092</b>	<b>0.2322<math>\pm</math>0.0476</b>	<b>0.2341<math>\pm</math>0.0193</b>
3	0.2907 $\pm$ 0.0921	0.1140 $\pm$ 0.0308	0.1819 $\pm$ 0.0767
4	0.3116 $\pm$ 0.0750	0.1820 $\pm$ 0.0541	0.2107 $\pm$ 0.0450
5	0.3384 $\pm$ 0.0892	0.1731 $\pm$ 0.0260	0.1464 $\pm$ 0.0531
6	0.3136 $\pm$ 0.0699	0.1763 $\pm$ 0.0482	0.2143 $\pm$ 0.0269

show slightly higher StrucEqu values than other  $C$  values while maintaining stability. In addition,  $C = 2$  and  $C = 6$  are considered balanced options in terms of both StrucEqu and SD. For SE-PrivGEmb<sub>Deg</sub>, on the Chameleon dataset,  $C = 2$  yields the highest StrucEqu value of 0.3661, with an extremely low SD value of 0.0092, indicating highly stable results. Other  $C$  values show lower StrucEqu values yet higher SD values, making  $C = 2$  the optimal choice. On the Power dataset,  $C = 2$  performs best with the highest StrucEqu value of 0.2322 and a moderate SD value of 0.0476. Although  $C = 1$  and  $C = 4$  exhibit relatively high StrucEqu values,  $C = 2$  remains the preferred option. On the Arxiv dataset,  $C = 1$  and  $C = 2$  show favorable combinations of StrucEqu and SD, particularly  $C = 2$  with a StrucEqu value of 0.2341 and an SD value of 0.0193. In summary,  $C = 2$  emerges as a suitable choice for both SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub>, consistently delivering high structure preservation and stability across different datasets.

4) **Parameter  $k$ :** We also explore the impact of the negative sampling number  $k$  within  $\{1, 2, 3, 4, 5, 6, 7\}$ , as shown in Table VI. For SE-PrivGEmb<sub>DW</sub>, all SD values demonstrate comparable performance across various datasets. For the StrucEqu value, on the Chameleon dataset,  $k = 5$  achieves the highest StrucEqu value of 0.4507. On the Power dataset, StrucEqu values fluctuate across different  $k$  values but also peak at  $k = 5$ . On the Arxiv dataset, the highest StrucEqu value is observed at  $k = 7$ , with  $k = 5$  being close to this peak. For SE-PrivGEmb<sub>Deg</sub>, on the Chameleon dataset,  $k = 6$  achieves a high StrucEqu value of 0.3824, yet it also derives the highest SD value of 0.0574, indicating relatively poor stability. Notably,  $k = 5$  achieves a StrucEqu value that is competitive with  $k = 6$ , while maintaining a significantly lower SD value of 0.0092. On the Power dataset, SE-PrivGEmb<sub>Deg</sub> achieves a significant peak at  $k = 5$  in terms of StrucEqu value. On the Arxiv dataset, the highest StrucEqu value is observed at  $k = 7$ , but  $k = 5$  also demonstrates competitive performance with a lower SD value compared to  $k = 7$ . Thus, taking into account the performance across various datasets, selecting  $k = 5$  is a

TABLE VI

SUMMARY OF STRUCEQU VALUES WITH DIFFERENT  $k$ , GIVEN  $\epsilon = 3.5$   
(RESULT: AVERAGE STRUCEQU  $\pm$  SD; **BOLD**: BEST)

$k$	SE-PrivGEmb <sub>DW</sub>		
	Chameleon	Power	Arxiv
1	0.4024 $\pm$ 0.0249	0.2036 $\pm$ 0.0766	0.3423 $\pm$ 0.0228
2	0.4121 $\pm$ 0.0339	0.2033 $\pm$ 0.0249	0.3500 $\pm$ 0.0450
3	0.4325 $\pm$ 0.0304	0.1913 $\pm$ 0.0132	0.3126 $\pm$ 0.0447
4	0.4190 $\pm$ 0.0670	0.2441 $\pm$ 0.0177	0.3801 $\pm$ 0.0421
5	<b>0.4507<math>\pm</math>0.0341</b>	<b>0.2522<math>\pm</math>0.0543</b>	0.3457 $\pm$ 0.0303
6	0.4053 $\pm$ 0.0501	0.2108 $\pm$ 0.0619	0.3599 $\pm$ 0.0456
7	0.3916 $\pm$ 0.0692	0.2010 $\pm$ 0.0565	<b>0.4247<math>\pm</math>0.0391</b>
$k$	SE-PrivGEmb <sub>Deg</sub>		
	Chameleon	Power	Arxiv
1	0.2286 $\pm$ 0.0121	0.1425 $\pm$ 0.0279	0.2203 $\pm$ 0.0587
2	0.2338 $\pm$ 0.0450	0.1542 $\pm$ 0.0202	0.2229 $\pm$ 0.0673
3	0.2406 $\pm$ 0.0531	0.0839 $\pm$ 0.0752	0.2068 $\pm$ 0.0654
4	0.2371 $\pm$ 0.0717	0.1987 $\pm$ 0.0714	0.1993 $\pm$ 0.0462
5	0.3661 $\pm$ 0.0092	<b>0.2322<math>\pm</math>0.0476</b>	0.2341 $\pm$ 0.0193
6	<b>0.3824<math>\pm</math>0.0574</b>	0.1762 $\pm$ 0.0183	0.2232 $\pm$ 0.0076
7	0.3710 $\pm$ 0.0375	0.1354 $\pm$ 0.0441	<b>0.2492<math>\pm</math>0.0432</b>

TABLE VII

IMPACT OF PERTURBATION STRATEGIES IN STRUCTURAL EQUIVALENCE  
(RESULT: AVERAGE STRUCEQU  $\pm$  SD; **BOLD**: BEST)

Datasets	SE-PrivGEmb <sub>DW</sub>	
	Naive	Non-zero
Chameleon( $\epsilon = 0.5$ )	0.1169 $\pm$ 0.0010	<b>0.3927<math>\pm</math>0.0341</b>
Chameleon( $\epsilon = 2$ )	0.1192 $\pm$ 0.0281	<b>0.4177<math>\pm</math>0.0359</b>
Chameleon( $\epsilon = 3.5$ )	0.1224 $\pm$ 0.0159	<b>0.4507<math>\pm</math>0.0341</b>
Power( $\epsilon = 0.5$ )	0.0857 $\pm$ 0.0164	<b>0.1945<math>\pm</math>0.0567</b>
Power( $\epsilon = 2$ )	0.0788 $\pm$ 0.0227	<b>0.2003<math>\pm</math>0.0330</b>
Power( $\epsilon = 3.5$ )	0.0796 $\pm$ 0.0160	<b>0.2522<math>\pm</math>0.0543</b>
Arxiv( $\epsilon = 0.5$ )	0.0888 $\pm$ 0.0033	<b>0.1770<math>\pm</math>0.0756</b>
Arxiv( $\epsilon = 2$ )	0.0844 $\pm$ 0.0263	<b>0.2199<math>\pm</math>0.0171</b>
Arxiv( $\epsilon = 3.5$ )	0.0848 $\pm$ 0.0086	<b>0.3457<math>\pm</math>0.0303</b>
Datasets	SE-PrivGEmb <sub>Deg</sub>	
	Naive	Non-zero
Chameleon( $\epsilon = 0.5$ )	0.1119 $\pm$ 0.0315	<b>0.2914<math>\pm</math>0.0188</b>
Chameleon( $\epsilon = 2$ )	0.1144 $\pm$ 0.0094	<b>0.3392<math>\pm</math>0.0367</b>
Chameleon( $\epsilon = 3.5$ )	0.1258 $\pm$ 0.0330	<b>0.3661<math>\pm</math>0.0092</b>
Power( $\epsilon = 0.5$ )	0.0638 $\pm$ 0.0013	<b>0.0877<math>\pm</math>0.0261</b>
Power( $\epsilon = 2$ )	0.0852 $\pm$ 0.0094	<b>0.1542<math>\pm</math>0.0231</b>
Power( $\epsilon = 3.5$ )	0.0833 $\pm$ 0.0151	<b>0.2322<math>\pm</math>0.0476</b>
Arxiv( $\epsilon = 0.5$ )	0.0779 $\pm$ 0.0267	<b>0.1629<math>\pm</math>0.0381</b>
Arxiv( $\epsilon = 2$ )	0.0611 $\pm$ 0.0130	<b>0.2112<math>\pm</math>0.0768</b>
Arxiv( $\epsilon = 3.5$ )	0.0784 $\pm$ 0.0168	<b>0.2341<math>\pm</math>0.0193</b>

well-balanced choice, as it reliably achieves strong structure preservation while ensuring stability.

### C. Impact of Perturbation Strategies

Table VII presents the results of the structural equivalence task using different noise addition strategies, namely naive perturbation using Eq. (6) and non-zero perturbation using Eq. (9), under different privacy budgets  $\epsilon \in \{0.5, 2, 3.5\}$ . Across all three datasets, the non-zero perturbation strategy consistently outperforms the naive perturbation strategy, indicating that our proposed methods (SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub>) are more effective in preserving structural equivalence. As  $\epsilon$  increases, the utility of the results generally improves under both strategies. However, the non-zero perturbation strategy maintains a significant advantage over the naive perturbation strategy, even at very large privacy budgets, demonstrating its

robustness and effectiveness in maintaining data utility while ensuring privacy.

#### D. Impact of Privacy Budget on Structural Equivalence

In the context of DP, the privacy budget  $\epsilon$  is a crucial parameter for determining privacy levels. We compare the StrucEqu results of various methods across seven privacy budgets from 0.5 to 3.5. Fig. 3 illustrates the StrucEqu outcomes for structural equivalence derived from eight algorithms. From this figure, we observe a consistent trend: utility generally improves with higher privacy budgets. In particular, SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> consistently achieve the highest accuracy in terms of StrucEqu among all privacy-preserving algorithms and maintain competitive performance compared to the non-private SE-GEmb<sub>DW</sub> and SE-GEmb<sub>Deg</sub>. DPGGAN and DPGVAE frequently yield poor results, primarily because they tend to converge prematurely when using MA, especially when the privacy budget is small. GAP perturbs the aggregate information, and the aggregation perturbation encounters compatibility issues with GNNs. Consequently, all aggregate outputs need to be re-perturbed at each training iteration, resulting in poor performance. ProGAP offers slightly better utility than GAP, and yet its performance remains significantly lower than that of SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub>, particularly at small privacy budgets.

#### E. Impact of Privacy Budget on Link Prediction

Fig. 4 displays the link prediction results for six private algorithms and two non-private algorithms across three datasets: Chameleon, Power, and Arxiv. It is evident that the non-private SE-GEmb<sub>DW</sub> and SE-GEmb<sub>Deg</sub> consistently outperform all private algorithms across all values of  $\epsilon$  on three datasets. On the Chameleon dataset, SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> outperform the remaining private algorithms across all values of  $\epsilon$ . DPGGAN and DPGVAE outperform GAP at  $\epsilon \in \{0.5, 1, 1.5\}$ , while DPGVAE and GAP excel against DPGGAN under  $\epsilon \in \{2.5, 3, 3.5\}$ . On the Power dataset, SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> generally outperform other private algorithms across most values of  $\epsilon$ . All private algorithms yield similar AUC values at  $\epsilon = 0.5$ , while SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> outperform the remaining private algorithms under  $\epsilon \in \{1, 1.5, 2, 2.5, 3, 3.5\}$ . For the Arxiv dataset, SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> outperform the other private algorithms for all values of  $\epsilon$ . It is worth noting that DPGGAN, GAP, and ProGAP show comparable performance, whereas DPGVAE consistently underperforms compared to the other private algorithms. In summary, our proposed SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> exhibit strong performance in maintaining privacy while preserving predictive accuracy, making them promising approaches for link prediction under differential privacy.

#### F. Impact of Privacy Budget on Node Clustering

The MI results for node clustering are shown in Fig. 5. Note that we only evaluate MI on the PPI and BlogCatalog datasets,

due to the absence of labeled data in the other datasets. From this figure, a key observation is that our SE-PrivGEmb<sub>DW</sub> and SE-PrivGEmb<sub>Deg</sub> consistently achieve the highest MI accuracy among all private methods. These results further confirm the effectiveness of our proposed noise tolerance mechanism through perturbing non-zero vectors. Additionally, it is observed that SE-PrivGEmb<sub>Deg</sub> is more suitable for node clustering than SE-PrivGEmb<sub>DW</sub>.

## VII. RELATED WORK

The related work of this paper covers differentially private deep learning and differentially private graph embedding generation.

### A. Differentially Private Deep Learning

DP is a widely used technique in privacy-preserving machine learning that provides strong privacy guarantees [38]–[42]. It addresses the challenge of quantifying the difference in outputs between two datasets that differ by only one element. Applying DP to deep learning is complex due to the intricate workings of deep neural networks (DNNs). Previous research [43], [44] has proposed incorporating norm-based gradient clipping into the SGD algorithm to mitigate this challenge. This technique limits the influence of individual examples on the gradients and introduces DP mechanisms to perturb the gradients accordingly. By ensuring that each gradient descent step has the differentially private property, the final output model achieves a level of DP through composability. However, training DNNs using SGD involves many iterative processes, so differentially private training algorithms need to monitor the accumulated privacy loss during the process and terminate when necessary to avoid exceeding the privacy budget. Research by Abadi *et al.* [1] has shown that existing strong composition theorems [45] lack precise analysis for DP. To overcome this limitation, MA was introduced. This method tracks the logarithmic moments of privacy loss variables and provides more accurate estimates of privacy loss when combining Gaussian mechanisms under random sampling. Additionally, Mironov *et al.* [27] propose a novel analysis method based on RDP to compute privacy accuracy. This method outperforms the MA method and significantly improves the performance of DPSGD. Further research [46]–[49] explores methods that involve slight modifications to the model structure or learning algorithm. For example, advanced approaches replace traditional activation functions (such as ReLU) in convolutional neural networks (CNNs) with more gentle Sigmoid functions. On the other hand, research [50]–[52] focuses on optimizations such as adaptive adjustment of gradient clipping bounds or dynamically partitioning the privacy budget. However, due to the excessive noise introduced by improper privacy budget partitioning in each epoch, DPSGD still struggles to achieve a satisfactory balance between privacy and utility.

### B. Differentially Private Graph Embedding Generation

There are two main types of techniques for generating graph embeddings: neural network-based methods and ma-

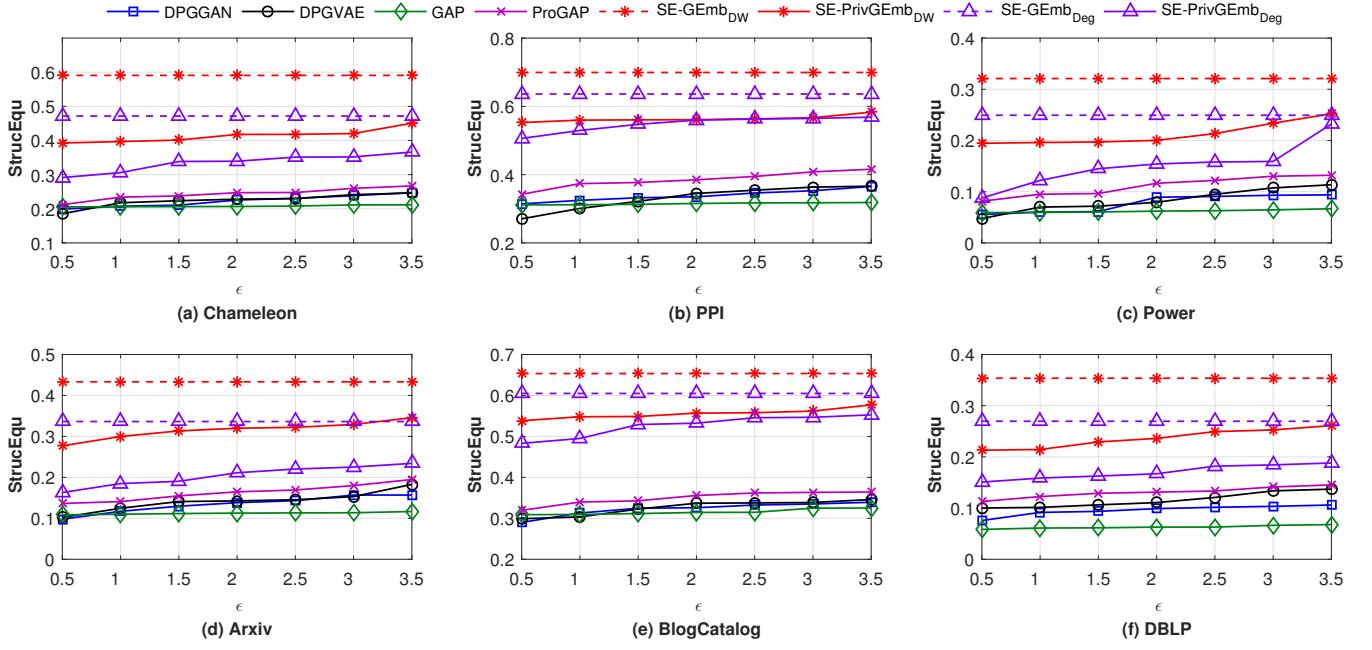


Fig. 3. Impact of Privacy Budget on Structural Equivalence

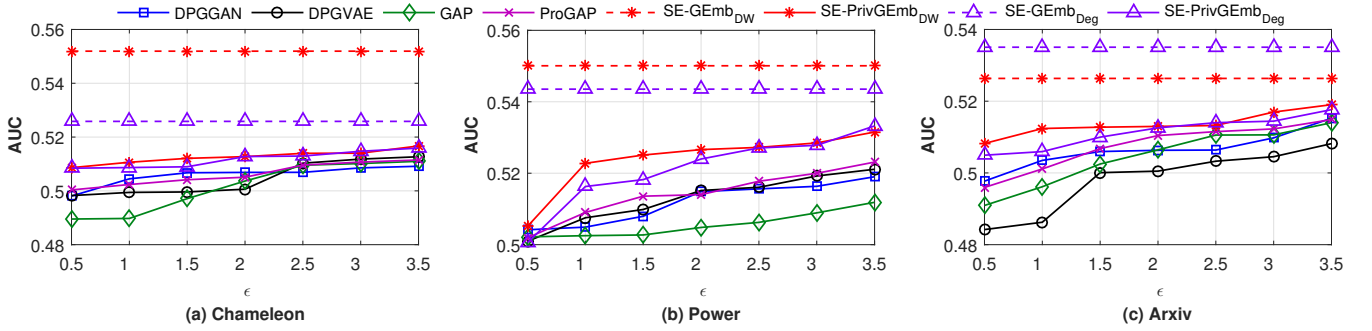


Fig. 4. Impact of Privacy Budget on Link Prediction

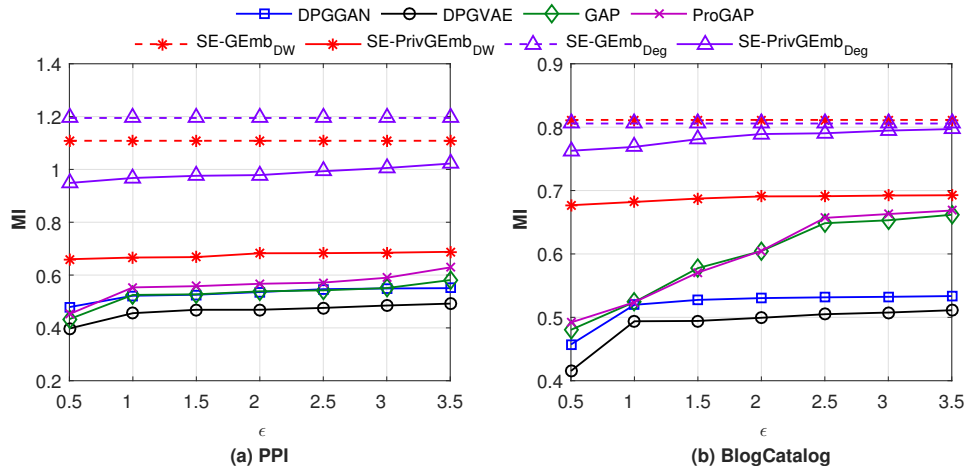


Fig. 5. Impact of Privacy Budget on Node Clustering



trix factorization-based methods. In this paper, we focus on the former category due to the scalability limitations of matrix factorization-based methods when dealing with large graphs. The introduction of word2vec [53], [54] has had a significant impact on neural network-based methods, leading to the development of various techniques such as DeepWalk [9], LINE [10], PTE [11], and node2vec [12]. Building on this foundation, Ahuja *et al.* [55] propose a private learning technique for sparse location data that combines skip-gram with differentially private stochastic gradient descent. However, their approach suffers from poor utility due to excessive partitioning of the privacy budget. Peng *et al.* [56] introduce a decentralized scalable learning framework that enables privacy-preserving learning of embeddings from multiple knowledge graphs in an asynchronous and peer-to-peer manner. Han *et al.* [57] develop a general framework for adapting knowledge graph embedding generation algorithms into differentially private versions. Pan *et al.* [58] present a federated unsupervised graph embedding generation that robustly captures graph structures, ensures DP, and operates with high communication efficiency. However, these studies, including the perturbation mechanisms, face challenges in maintaining utility similar to Ahuja *et al.* [55]. Yang *et al.* [2] propose differentially private GAN and differentially private VAE models for graph synthesis and link prediction. However, despite the low sensitivity of this approach, it tends to converge prematurely under MA, particularly with a limited privacy budget, resulting in diminished performance in both privacy and utility. Another area of research [3]–[8] within the field of differential private graph learning focuses on GNNs. The aggregation perturbation (AP) technique is commonly employed as a leading method for ensuring DP in GNNs. Unlike traditional DPSGD algorithms, most existing differentially private GNN methods perturb aggregate information obtained from the GNN neighborhood aggregation step. However, AP encounters compatibility issues with standard GNN architectures, necessitating the re-perturbation of all aggregate outputs at each training iteration, resulting in high privacy costs. Furthermore, none of all the existing methods can offer structure-preference settings. Setting preferences is essential for extracting specific graph structures that align with mining objectives, improve predictive accuracy, and yield meaningful insights.

## VIII. CONCLUSION

In this paper, we have presented SE-PrivGEmb, a structure-preference enabled graph embedding generation under DP using skip-gram. There are two technical highlights. First, we deeply analyze the optimization of skip-gram and design a noise tolerance mechanism via non-zero vector perturbation. This mechanism accommodates arbitrary structure preferences and mitigates utility degradation caused by high sensitivity. Second, by carefully designing negative sampling probabilities in skip-gram, we prove theoretically that skip-gram can preserve arbitrary node proximities, thereby aligning with mining objectives, improve predictive accuracy, and yield

meaningful insights. Through privacy analysis, we prove that the published low-dimensional node vectors satisfy node-level RDP. Extensive experiments on six real-world graph datasets show that our solution substantially outperforms state-of-the-art competitors.

For future work, we plan to extend our method to attribute graphs that include billions of nodes or edges. Notably, the attributes associated with the nodes are independent and can be easily managed due to their low sensitivity. Additionally, inspired by the dynamic graph embedding discussed in [13], we also plan to extend our method to dynamic graph embedding while obeying DP. Addressing dynamic graphs will face two significant challenges: allocating privacy budgets to each data element at each version and managing noise accumulation during continuous data publishing.

## REFERENCES

- [1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.
- [2] C. Yang, H. Wang, K. Zhang, L. Chen, and L. Sun, “Secure deep graph generation with link differential privacy,” in *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021, pp. 3271–3278.
- [3] I. E. Olatunji, T. Funke, and M. Khosla, “Releasing graph neural networks with differential privacy guarantees,” *arXiv preprint arXiv:2109.08907*, 2021.
- [4] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, “Node-level differentially private graph neural networks,” *arXiv preprint arXiv:2111.15521*, 2021.
- [5] Q. Zhang, H. k. Lee, J. Ma, J. Lou, C. Yang, and L. Xiong, “Dpar: Decoupled graph neural networks with node-level differential privacy,” in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 1170–1181.
- [6] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, “Gap: Differentially private graph neural networks with aggregation perturbation,” in *USENIX Security 2023-32nd USENIX Security Symposium*, 2023.
- [7] S. Sajadmanesh and D. Gatica-Perez, “Progap: Progressive graph neural networks with differential privacy guarantees,” in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 596–605.
- [8] Z. Xiang, T. Wang, and D. Wang, “Preserving node-level privacy in graph neural networks,” in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 4714–4732.
- [9] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.
- [10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.
- [11] J. Tang, M. Qu, and Q. Mei, “Pte: Predictive text embedding through large-scale heterogeneous text networks,” in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1165–1174.
- [12] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
- [13] L. Du, Y. Wang, G. Song, Z. Lu, and J. Wang, “Dynamic network embedding: An extended approach for skip-gram based network embedding,” in *IJCAI*, vol. 2018, 2018, pp. 2086–2092.
- [14] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

- [15] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate estimation of the degree distribution of private networks," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 169–178.
- [16] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*. Springer, 2006, pp. 1–12.
- [17] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 2017, pp. 263–275.
- [18] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.
- [19] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, pp. 623–630, 2009.
- [20] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.
- [21] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 517–526.
- [22] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Twenty-fourth international joint conference on artificial intelligence*, 2015, pp. 2111–2117.
- [23] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 148–162.
- [24] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 459–467.
- [25] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1226–1235.
- [26] Y. Zhu and Y.-X. Wang, "Poisson subsampled rényi differential privacy," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7634–7642.
- [27] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled gaussian mechanism," *arXiv preprint arXiv:1908.10530*, 2019.
- [28] Z. Bu, Y.-X. Wang, S. Zha, and G. Karypis, "Differentially private optimization on large model at small cost," in *International Conference on Machine Learning*. PMLR, 2023, pp. 3192–3218.
- [29] J. Jin, M. Heimann, D. Jin, and D. Koutra, "Toward understanding and evaluating structural node embeddings," *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 3, pp. 1–32, 2021.
- [30] C. Stark, B.-J. Breitkreutz, T. Regul, L. Boucher, A. Breitkreutz, and M. Tyers, "Biogrid: a general repository for interaction datasets," *Nucleic acids research*, vol. 34, pp. D535–D539, 2006.
- [31] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.
- [32] D. Nguyen, W. Luo, T. D. Nguyen, S. Venkatesh, and D. Phung, "Learning graph representation via frequent subgraphs," in *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018, pp. 306–314.
- [33] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.
- [34] L. Du, X. Chen, F. Gao, Q. Fu, K. Xie, S. Han, and D. Zhang, "Understanding and improvement of adversarial training for network embedding from an optimization perspective," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 230–240.
- [35] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.
- [36] Y.-A. Lai, C.-C. Hsu, W. H. Chen, M.-Y. Yeh, and S.-D. Lin, "Prune: Preserving proximity and global ranking for network embedding," *Advances in neural information processing systems*, vol. 30, 2017.
- [37] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1051–1065, 2018.
- [38] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *2014 IEEE 55th annual symposium on foundations of computer science*. IEEE, 2014, pp. 464–473.
- [39] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in neural information processing systems*, 2008, pp. 289–296.
- [40] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.
- [41] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for svm learning," *arXiv preprint arXiv:0911.5708*, 2009.
- [42] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: regression analysis under differential privacy," *Proceedings of the VLDB Endowment*, pp. 1364–1375, 2012.
- [43] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private language models without losing accuracy," *arXiv preprint arXiv:1710.06963*, 2017.
- [44] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *2013 IEEE global conference on signal and information processing*. IEEE, 2013, pp. 245–248.
- [45] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 51–60.
- [46] C. Chen and J. Lee, "Stochastic adaptive line search for differentially private optimization," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 1011–1020.
- [47] M. Nasr, R. Shokri *et al.*, "Improving deep learning with differential privacy using gradient encoding and denoising," *arXiv preprint arXiv:2007.11524*, 2020.
- [48] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson, "Tempered sigmoid activations for deep learning with differential privacy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9312–9321.
- [49] F. Tramer and D. Boneh, "Differentially private learning needs better features (or much more data)," *arXiv preprint arXiv:2011.11660*, 2020.
- [50] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar, "Adaclip: Adaptive clipping for private sgd," *arXiv preprint arXiv:1908.07643*, 2019.
- [51] L. Xiang, J. Yang, and B. Li, "Differentially-private deep learning from an optimization perspective," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 559–567.
- [52] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 332–349.
- [53] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
- [54] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.
- [55] R. Ahuja, G. Ghinita, and C. Shahabi, "Differentially-private next-location prediction with neural networks," in *Proceedings of the 23rd International Conference on Extending Database Technology*, 2020, pp. 121–132.
- [56] H. Peng, H. Li, Y. Song, V. Zheng, and J. Li, "Differentially private federated knowledge graphs embedding," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1416–1425.
- [57] X. Han, D. Dell’Aglío, T. Grubenmann, R. Cheng, and A. Bernstein, "A framework for differentially-private knowledge graph embeddings," *Journal of Web Semantics*, vol. 72, p. 100696, 2022.
- [58] Q. Pan and Y. Zhu, "Fedwalk: Communication efficient federated unsupervised node embedding with differential privacy," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1317–1326.