Privacy-Preserving Graph Embedding based on Local Differential Privacy

Zening Li
Beijing Institute of Technology
Beijing, China
zening-li@outlook.com

Rong-Hua Li
Beijing Institute of Technology
Beijing, China
lironghuabit@126.com

Meihao Liao
Beijing Institute of Technology
Beijing, China
mhliao@bit.edu.cn

Fusheng Jin
Beijing Institute of Technology
Beijing, China
jfs21cn@bit.edu.cn

Guoren Wang
Beijing Institute of Technology
Beijing, China
wanggrbit@gmail.com

Abstract

Graph embedding has become a powerful tool for learning latent representations of nodes in a graph. Despite its superior performance in various graph-based machine learning tasks, serious privacy concerns arise when the graph data contains personal or sensitive information. To address this issue, we investigate and develop graph embedding algorithms that satisfy local differential privacy (LDP). We introduce a novel privacy-preserving graph embedding framework, named PrivGE, to protect node data privacy. Specifically, we propose an LDP mechanism to obfuscate node data and utilize personalized PageRank as the proximity measure to learn node representations. Furthermore, we provide a theoretical analysis of the privacy guarantees and utility offered by the PrivGE framework. Extensive experiments on several real-world graph datasets demonstrate that PrivGE achieves an optimal balance between privacy and utility, and significantly outperforms existing methods in node classification and link prediction tasks.

CCS Concepts

• **Security and privacy** → **Privacy protections**; • **Information systems** → **Data mining**.

Keywords

Differential Privacy; Graph Embedding; Graph Neural Networks; Personalized PageRank

ACM Reference Format:
Zening Li, Rong-Hua Li, Meihao Liao, Fusheng Jin, and Guoren Wang. 2024. Privacy-Preserving Graph Embedding based on Local Differential Privacy. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679759

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679759

1 Introduction

Many types of real-world data can be naturally represented as graphs, such as social networks, financial networks, and transportation networks. Over the past few years, graph embedding has attracted much attention due to its superior performance in various machine learning tasks, such as node classification and link prediction [30, 36, 41, 45]. Graph embedding is a representation learning problem that uses vectors to capture node features and structural information in a graph. However, most real-world graphs involve sensitive information about individuals and their activities, such as users' profile information and comments in social networks. The direct release of embedding vectors of users provides a potential way for malicious attackers to infer the attributes and social interactions, which could potentially be private to the users. The growing awareness of privacy and the establishment of regulations and laws indicate that it is important to develop privacy-preserving graph embedding algorithms.

Differential privacy (DP) [9] has recently become the dominant paradigm for safeguarding individual privacy in data analysis. A vast majority of differentially private graph learning algorithms are designed under the centralized model [2, 10, 22, 33, 39, 43], which assumes that a trusted data curator holds the personal data of all users and releases sanitized versions of statistics or machine learning models. However, this assumption is impractical in some applications due to security or logistical reasons. The centralized model carries the risk of users' data being breached by the trusted data curator through illegal access or internal fraud [13, 14]. In addition, some social networks are inherently decentralized and distributed, where no centralized party holds the entire social graph. As a result, the centralized DP algorithms cannot be applied to these decentralized social networks.

In contrast to centralized DP, local differential privacy (LDP) [18] has emerged as a promising approach that ensures stronger privacy guarantees for users in scenarios involving data aggregation. LDP operates under a local model, where each user perturbs their data locally and transmits only the perturbed data to an untrusted data curator. This means that the original personal data remains confined to the users' local devices, effectively eliminating the risk of data leakage that might occur in the centralized DP settings. As a result, LDP provides an enhanced level of privacy protection for individuals. The practicality and effectiveness of LDP have been recognized by major technology companies, such as Google [11],

and Microsoft [5], which have deployed LDP-based solutions to handle sensitive user data while preserving privacy. Moreover, the applicability of LDP extends beyond centralized settings, making it an appealing choice for decentralized applications.

In this paper, our focus is on exploring the design of graph embedding algorithms that satisfy LDP, where the node features are private and sensitive, and the global graph structure information is maintained by the data curator. This scenario arises in various domains, particularly in social network analysis and mobile computing. For instance, consider a social networking platform or a dating application that utilizes node representations to capture relationships between users. In this context, individuals' attributes and preferences are treated as private node features. Integrating LDP in graph embedding for such applications can safeguard users' sensitive data while empowering the platform to offer valuable services.

In the local setting for DP, each user sends only perturbed node features to the data collector. However, a key challenge arises when dealing with high-dimensional node features used for graph embedding, as the perturbation in such scenarios can result in significant information loss. To overcome this challenge, some researchers have proposed various approaches, including sampling techniques and tailored perturbation mechanisms, aimed at preserving utility in the high-dimensional space [8, 15, 24, 32, 37]. Nevertheless, these mechanisms also introduce excessive noise to the data, potentially compromising overall performance.

In this paper, we propose PrivGE, a novel privacy-preserving graph embedding framework based on private node data. Our framework offers provable privacy guarantees, building on the principles of local differential privacy. Specifically, to protect the privacy of node features, we propose the HDS (an acronym for High-Dimensional Square wave) mechanism, an LDP perturbation technique tailored for high-dimensional data. Each user can adopt this perturbation mechanism to obfuscate their features before sending them to the data curator. The server leverages graph structure information and perturbed node features to learn graph representations. To avoid neighborhood explosion and over-smoothing issues, we decouple the feature transformation from the graph propagation. Furthermore, we adopt personalized PageRank as the proximity measure to learn node representations. Importantly, we conduct a comprehensive theoretical analysis of the utility of the PrivGE framework. Our findings indicate that the proposed approach yields smaller error bounds than existing mechanisms, specifically, from $O(\frac{d\log(d/\delta)}{\epsilon})$ down to $O(\log(d/\delta))$, making it a more efficient solution[1]. Finally, to assess the effectiveness of the PrivGE framework, we conduct extensive experiments on various real-world datasets. The results demonstrate that our proposed method establishes state-of-the-art performance and achieves decent privacy-utility trade-offs in node classification and link prediction tasks. In summary, we highlight the main contributions as follows:

- We propose PrivGE, an innovative framework that aims to preserve privacy in graph embedding. Our method provides provable privacy guarantees and simultaneously ensures effective graph representation learning.
- To address the challenge dealing with high-dimensional node features, we propose the HDS mechanism to protect node feature privacy. This perturbation technique empowers users to obfuscate their features locally before reporting them to the data curator, thus enhancing privacy protection.
- We conduct a comprehensive theoretical analysis of the utility of PrivGE and alternative mechanisms. The results demonstrate that our mechanism offers smaller error bounds than the others, reducing them from $O(\frac{d\log(d/\delta)}{\epsilon})$ to $O(\log(d/\delta))$.
- We conduct extensive experiments on various real datasets. The experimental results show that our proposed method achieves better privacy-utility trade-offs than existing solutions. For instance, our proposed method achieves about 8% higher accuracy than the best competitor on the Pubmed dataset in node classification.

2 Preliminaries

Problem Statement. We consider an undirected and unweighted graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (i.e., users) and $\mathcal{E}$ represents the set of edges. Let $|\mathcal{V}|$ be the number of nodes. Each user $v \in \mathcal{V}$ is characterized by a d-dimensional feature vector $\mathbf{x}_v$, and we use $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ to denote the feature matrix. Without loss of generality, we assume the node features are normalized into $[-1, 1]$[2]. Let $\mathbf{A}$ and $\mathbf{D}$ represent the adjacency matrix and the diagonal degree matrix, respectively. For each node $v \in \mathcal{V}$, $\mathcal{N}(v)$ is the set of neighbors of v, and the degree of v is $|\mathcal{N}(v)|$.

We assume that the data curator is an untrusted party with access to the node set $\mathcal{V}$ and edge set $\mathcal{E}$. However, the data curator cannot observe the feature matrix $\mathbf{X}$, which is private to users. Our ultimate objective is to learn a node embedding matrix and simultaneously protect the privacy of node data.

Local Differential Privacy. Differential privacy [9] has become the dominant model for the protection of individual privacy from powerful and realistic adversaries. DP can be bifurcated into centralized DP and local DP. Centralized DP assumes a scenario where a trusted curator holds all users' personal data and releases sanitized versions of the statistics. In contrast to centralized DP, local DP operates under the assumption of a local model, where the data curator is considered untrusted. Specifically, each user perturbs their data via a randomized perturbation mechanism and sends the obfuscated data to the untrusted data curator.

Definition 2.1 (ϵ-Local Differential Privacy [18])**.** Given $\epsilon > 0$, a randomized algorithm $\mathcal{A}$ satisfies ϵ-local differential privacy if and only if for any two users' private data x and x', and for any possible output $y \in Range(\mathcal{A})$, we have

$$\Pr[\mathcal{A}(x) = y] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(x') = y]. \tag{1}$$

Here, the parameter ϵ is called the privacy budget, which controls the strength of privacy protection: a lower privacy budget indicates stronger privacy preservation but leads to lower utility. In addition, LDP satisfies some important properties that can help us develop more sophisticated algorithms.

[1]Note that d denotes the dimension of the node features, and ϵ represents the privacy budget. Additionally, δ is a constant between $(0, 1]$.

[2]Note that it is a common assumption in [32] that the feature fields are known to the users, so this normalization step does not compromise privacy.

Proposition 2.2 (Sequential Composition [3]). *Given the sequence of computations $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$, if each $\mathcal{A}_i$ satisfies ϵ_i-LDP, then their sequential execution on the same dataset satisfies $\sum_i \epsilon_i$-LDP.*

Proposition 2.3 (Post-processing [3]). *Given $\mathcal{A}(\cdot)$ that satisfies ϵ-LDP, then for any algorithm $\mathcal{B}$, the composed algorithm $\mathcal{B}(\mathcal{A}(\cdot))$ also satisfies ϵ-LDP.*

Graph Embedding. The task of graph embedding is to learn the latent representation of each node. Numerous studies have shown that the latent representations can capture the structural and inherent properties of the graph, which can facilitate downstream inference tasks, such as node classification and link prediction [30, 41, 45].

As an important class of graph embedding methods, the message passing framework is of interest due to its flexibility and favorable performance. This framework comprises two phases: (i) message propagation among neighbors, and (ii) message aggregation to update representation. Most GNN models, such as GCN [20] and GAT [35], employ a message passing process to spread information. At each layer, feature transformation is coupled with aggregation and propagation. Increasing the number of layers allows the model to incorporate information from more distant neighbors, which promotes a more comprehensive node representation. However, this approach may lead to over-smoothing and neighborhood explosion [1].

To address these inadequacies, some studies [1, 21] decouple the feature transformation from the graph propagation and exploit node proximity queries to incorporate multi-hop neighborhood information. Personalized PageRank [1], a widely-used proximity measure, can characterize node distances and similarities. Consequently, we apply the personalized PageRank matrix to the feature matrix $\mathbf{X}$ to derive the representation matrix $\mathbf{Z}$. The graph propagation equation is defined as follows:

$$\mathbf{Z} = \mathbf{\Pi} \cdot \mathbf{X} = \sum_{\ell=0}^{\infty} \alpha(1-\alpha)^{\ell} \cdot (\mathbf{D}^{r-1}\mathbf{A}\mathbf{D}^{-r})^{\ell} \cdot \mathbf{X}, \tag{2}$$

where $\mathbf{\Pi}$ is the personalized PageRank matrix, $r \in [0, 1]$ is the convolution coefficient and $\alpha \in (0, 1)$ is the decay factor. The parameter α controls the amount of information we capture from the neighborhood. To be specific, for the values of α closer to 1, we place more emphasis on the immediate neighborhood of the node, which can avoid over-smoothing. As the value of α decreases to 0, we instead give more attention to the multi-hop neighborhood of the node.

3 The Proposed Method

In this section, we describe our proposed differentially private framework for graph embedding. It consists of two components: a perturbation module and a propagation module. The perturbation module is utilized to locally obfuscate node features before they are sent to the data curator. This component helps relieve users' concerns about sharing their private information. However, the node features to be collected are likely high-dimensional, whereas most LDP perturbation functions focus on one-dimensional data, such as the Laplace mechanism. Since each user is authorized a limited privacy budget, the allocated privacy budget in each dimension is diluted as the number of dimensions increases, which results in more noise injection. While some LDP mechanisms, such as One-bit mechanism [5] and Piecewise mechanism [37], have been extended to handle the high-dimensional data [32, 37], these mechanisms introduce much noise into the data, which compromises performance. The Square wave mechanism [23] is another LDP mechanism that aims to reconstruct the distribution of one-dimensional numerical attributes. This mechanism provides a more concentrated perturbation than the two mechanisms mentioned above (i.e., One-bit mechanism and Piecewise mechanism). Thus, to address this problem, we extend the Square wave mechanism to handle high-dimensional data, and develop an LDP mechanism to perturb the node features.

The propagation module is used to spread node features via information exchange between adjacent nodes, where the representation of each node is updated based on the aggregation of its neighbors' features. However, most methods suffer from neighborhood explosion and over-smoothing issues, as explained in Section 2. To address these problems, we decouple feature transformation and propagation, and adopt personalized PageRank as the graph propagation formula to obtain the representation matrix. More importantly, we provide a comprehensive theoretical analysis of the utility of our proposed method and alternative mechanisms.

In the rest of this section, we first introduce the technical details of the perturbation module used for privacy assurances and some theoretical properties of our proposed mechanism. Next, we present the propagation process designed for graph embedding. Finally, we conduct a utility analysis of the proposed framework.

3.1 Perturbation Module

The target of the perturbation module is to gather node features from individuals under LDP. In specific, each user $v \in \mathcal{V}$ perturbs his/her private feature vector $\mathbf{x}_v$ using the perturbation mechanism and sends the perturbed data $\tilde{\mathbf{x}}_v$ to the data curator. The crucial aspect is to devise a randomization mechanism that provides plausible deniability. In this section, we first review three existing perturbation mechanisms and discuss their deficiencies. Then, we propose our mechanism for feature perturbation.

Existing Solutions. Laplace mechanism [9] is a well-established approach enforcing differential privacy. It can be applied to the LDP setting in the following manner. Assuming each user possesses a one-dimensional value x in the range of $[-1, 1]$, we define a randomized function that generates a perturbed value $\tilde{x} = x + Lap(2/\epsilon)$. Here, $Lap(\lambda)$ represents a random variable that follows a Laplace distribution with a scale parameter λ. The Laplace distribution is characterized by the probability density function $f(x) = \frac{1}{2\lambda} \exp(-\frac{|x|}{\lambda})$.

To extend this mechanism to high-dimensional values, a straightforward method is to collect the perturbed value separately using the Laplace mechanism in each dimension. In this approach, each dimension is assigned a privacy budget of ϵ/d. Applying the composition property of LDP as described in Proposition 2.2, we can conclude that the entire collection of values satisfies ϵ-LDP. In the high-dimensional space, since the injected Laplace noise in each dimension follows $Lap(2d/\epsilon)$, it is evident that the perturbed value $\tilde{x}$ is unbiased, and its variance is $\frac{8d^2}{\epsilon^2}$. Furthermore, the Laplace mechanism embodies a category of LDP mechanisms known as

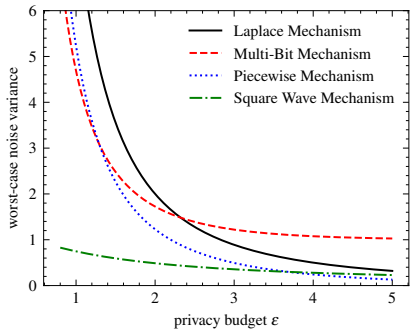


Figure 1: The worst-case noise variance vs. privacy budget for one-dimensional data.

unbounded mechanisms, where the noise injected into the original value ranges from negative to positive infinity.

In the one-dimensional Piecewise mechanism [37], the input domain is $[-1, 1]$, and the range of perturbed data is $[-s, s]$, where $s = \frac{e^{\epsilon/2}+1}{e^{\epsilon/2}-1}$. Given an original value $x \in [-1, 1]$, the perturbed value $\tilde{x}$ is sampled from the following distribution:

$$\Pr[\tilde{x} = c|x] = \begin{cases} p, & \text{if } c \in [\ell(x), r(x)], \\ \frac{p}{e^{\epsilon}}, & \text{if } c \in [-s, \ell(x)) \cup (r(x), s], \end{cases} \tag{3}$$

where $p = \frac{e^{\epsilon}-e^{\epsilon/2}}{2e^{\epsilon/2}+2}$, $\ell(x) = \frac{s+1}{2} \cdot x - \frac{s-1}{2}$, and $r(x) = \ell(x) + s - 1$. Wang et al. [37] also propose an extension of the Piecewise mechanism to process high-dimensional data. The extended mechanism adopts a sampling technique so that each user reports only k out of d dimensions of his/her perturbed data to the data curator. In that case, each reporting dimension is allocated ϵ/k privacy budget, and the reporting data $\hat{x}$ is calibrated to ensure that the final outcome $\tilde{x}$ is unbiased. Formally, $\tilde{x}$ is obtained by $\tilde{x} = \frac{d}{k} \cdot \hat{x}$. In the high-dimensional setting, the variance of $\tilde{x}$ induced by Piecewise mechanism is $\mathrm{Var}[\tilde{x}] = \frac{d(e^{\epsilon/(2k)}+3)}{3k(e^{\epsilon/(2k)}-1)^2} + [\frac{d \cdot e^{\epsilon/(2k)}}{k(e^{\epsilon/(2k)}-1)} - 1] \cdot x^2$.

Multi-bit mechanism [32] is another perturbation function used to handle high-dimensional data under LDP. In its one-dimensional form, the original data $x \in [-1, 1]$ is perturbed into -1 or 1, with the following probabilities:

$$\Pr[\tilde{x} = c|x] = \begin{cases} \frac{1}{e^{\epsilon}+1} + \frac{x+1}{2} \cdot \frac{e^{\epsilon}-1}{e^{\epsilon}+1}, & \text{if } c = 1, \\ \frac{e^{\epsilon}}{e^{\epsilon}+1} - \frac{x+1}{2} \cdot \frac{e^{\epsilon}-1}{e^{\epsilon}+1}, & \text{if } c = -1. \end{cases} \tag{4}$$

In the high-dimensional space, similar to the Piecewise mechanism, the algorithm first uniformly samples k out of d dimensions without replacement and then performs $\frac{\epsilon}{k}$-LDP perturbation for each sampled dimension. In the end, the data curator transforms the reporting data $\hat{x}$ to its unbiased estimate $\tilde{x} = \frac{d(e^{\epsilon/k}+1)}{k(e^{\epsilon/k}-1)} \cdot \hat{x}$. The variance of $\tilde{x}$ induced by Multi-bit mechanism is $\mathrm{Var}[\tilde{x}] = \frac{d}{k}(\frac{e^{\epsilon/k}+1}{e^{\epsilon/k}-1})^2 - x^2$. In contrast to the Laplace mechanism, the Piecewise and Multi-bit mechanisms perturb the original value into a bounded domain. Consequently, these mechanisms are referred to as bounded mechanisms.

Deficiencies of Existing Solutions. Even though these mechanisms can handle high-dimensional data, they introduce a significant amount of noise to the private data, leading to a decline in performance. To be specific, in the one-dimensional scenario, we visualize the worst-case noise variance of different mechanisms under varying privacy budgets, as illustrated in Figure 1. Note that in

Algorithm 1: Extended Square Wave Mechanism

Input: single feature $x \in [-1, 1]$, privacy budget $\epsilon > 0$
Output: perturbed feature $\tilde{x} \in [-b-1, 1+b]$

Let $b = \frac{\epsilon e^{\epsilon} - e^{\epsilon} + 1}{e^{\epsilon}(e^{\epsilon} - \epsilon - 1)}$;
Let η be sampled uniformly from $[0, 1]$;
if $\eta < \frac{be^{\epsilon}}{be^{\epsilon}+1}$ **then**
  Sample a random value $\tilde{x}$ uniformly from $[x-b, x+b]$;
else
  Sample a random value $\tilde{x}$ uniformly from $[-b-1, x-b) \cup (x+b, 1+b]$;
return $\tilde{x}$

Figure 1, we also add the Square wave mechanism [23] for comparison. The Square wave mechanism was originally proposed in [23], which can achieve LDP when handling one-dimensional data. We can observe that the Square wave mechanism [23] provides a considerably smaller noise variance than the Piecewise mechanism for $\epsilon \leq 3.5$, and only slightly larger than the latter for $\epsilon > 3.5$. Moreover, the worst-case noise variance provided by the Square wave mechanism is consistently smaller than that of the Laplace mechanism and the Multi-bit mechanism when $\epsilon \leq 5.0$. In consequence, the Square wave mechanism provides more concentrated perturbation. In the high-dimensional setting, unbiased calibration is not conducive to graph embedding and can result in injecting excessive noise. The Square wave mechanism is designed to estimate the distribution of one-dimensional numerical data. Inspired by the ideas of [37] and [32], we generalize the Square wave mechanism to feature collection in high-dimensional space.

HDS Mechanism. Our HDS mechanism is built upon the Square wave mechanism [23], which can handle only one-dimensional data. The Square wave mechanism is based on the following intuition. Given a single feature x, the perturbation module should report a value close to x with a higher probability than a value far away from x. To some extent, the value close to x also carries useful information about the input. The Square wave mechanism is initially designed for an input domain of $[0, 1]$. However, in our scenario, the node features are normalized to the range of $[-1, 1]$. Consequently, we extend this mechanism to enhance its capability in handling a broader range of node features, specifically $[-1, 1]$. Algorithm 1 outlines the perturbation process for one-dimensional data. The algorithm takes a single feature $x \in [-1, 1]$ as input and produces a perturbed feature $\tilde{x} \in [-b-1, 1+b]$, where $b = \frac{\epsilon e^{\epsilon} - e^{\epsilon} + 1}{e^{\epsilon}(e^{\epsilon} - \epsilon - 1)}$. Let $p = \frac{e^{\epsilon}}{2be^{\epsilon}+2}$ and $q = \frac{1}{2be^{\epsilon}+2}$. The noisy output $\tilde{x}$ follows the distribution as below:

$$\Pr[\tilde{x} = c|x] = \begin{cases} p, & \text{if } c \in [x-b, x+b], \\ q, & \text{if } c \in [-b-1, x-b) \cup (x+b, 1+b]. \end{cases} \tag{5}$$

Algorithm 2 presents the pseudo-code of our HDS mechanism for high-dimensional feature collection. The algorithm requires that each user perturbs only k dimensions of their features vector rather than d. This is because, according to the composition property of LDP described in Proposition 2.2, it increases the privacy budget for each dimension from ϵ/d to ϵ/k, reducing the noise variance. Given a feature vector $\mathbf{x}$, the algorithm first uniformly samples k values from $\{1, 2, \ldots, d\}$ without replacement, where k is a parameter that controls the number of dimensions to be perturbed. Then

Algorithm 2: HDS Mechanism

Input: feature vector $\mathbf{x} \in [-1,1]^d$, privacy budget $\epsilon > 0$, sampling parameter $k \in \{1,2,...,d\}$
Output: perturbed feature vector $\tilde{\mathbf{x}} \in [-b-1, 1+b]^d$

Let $\mathcal{S}$ be a set of k values sampled uniformly without replacement from $\{1,2,\ldots,d\}$;
for $j \in \{1,2,\ldots,d\}$ **do**
  if $j \in \mathcal{S}$ **then**
    $\tilde{x}_j \leftarrow$ Feed x_j and $\frac{\epsilon}{k}$ as input to Algorithm 1;
  else
    $\tilde{x}_j \leftarrow 0$;
return $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_d]^\top$

for each sampled value j, the perturbed feature $\tilde{x}_j$ is generated by Algorithm 1, taking x_j and $\frac{\epsilon}{k}$ as input. Correspondingly, the rest of the $d-k$ features are encoded into 0 to prevent privacy leakage. Given that the output domain of the HDS mechanism is bounded, our proposed mechanism can be categorized as a bounded mechanism.

Theorem 3.1. *The HDS mechanism presented in Algorithm 2 satisfies ϵ-local differential privacy for each node.*

Proof. First, we prove that the Algorithm 1 provides ϵ-LDP. Let $\mathcal{A}(x)$ be the Algorithm 1 that takes the single feature x as input, and $\tilde{x} = \mathcal{A}(x)$ is the obfuscated feature corresponding to x. Suppose x_1 and x_2 are private features of any two users. According to Equation (5), for any output $\tilde{x} \in [-b-1, 1+b]$, we have $\frac{\Pr[\mathcal{A}(x_1)=\tilde{x}]}{\Pr[\mathcal{A}(x_2)=\tilde{x}]} \le \frac{e^\epsilon}{2be^\epsilon+2} \cdot (2be^\epsilon+2) = e^\epsilon$. Thus, the Algorithm 1 satisfies ϵ-LDP. Since Algorithm 2 executes $\frac{\epsilon}{k}$-LDP operations (Algorithm 1) k times on the same input data, then according to the composition property, Algorithm 2 satisfies ϵ-local differential privacy. □

In the following analysis, we examine the bias and variance of the HDS mechanism.

Lemma 3.2. *Let $\tilde{\mathbf{x}}_v$ be the output of Algorithm 2 on the input vector $\mathbf{x}_v$. For any dimension $j \in \{1,2,\ldots,d\}$, $\mathbb{E}[\tilde{x}_{v,j}] = C \cdot x_{v,j}$ and $\mathrm{Var}[\tilde{x}_{v,j}] = \frac{k(b^3e^{\epsilon/k}+3b^2+3b+1)}{3d(be^{\epsilon/k}+1)} + (C-C^2)x_{v,j}^2$, where $C = \frac{kb(e^{\epsilon/k}-1)}{dbe^{\epsilon/k}+1}$.*

Proof. For the expectation, we have

$$\mathbb{E}[\tilde{x}_{v,j}] = \mathbb{E}[\tilde{x}_{v,j}|j \in \mathcal{S}]\Pr[j \in \mathcal{S}] + \mathbb{E}[\tilde{x}_{v,j}|j \notin \mathcal{S}]\Pr[j \notin \mathcal{S}]$$
$$= \frac{k}{d} \cdot \mathbb{E}[\tilde{x}_{v,j}|j \in \mathcal{S}]. \tag{6}$$

According to Equation (5), we have

$$\mathbb{E}[\tilde{x}_{v,j}|j \in \mathcal{S}] = \frac{1}{2be^{\epsilon/k}+2}\left(\int_{-1-b}^{x_{v,j}-b} t\,dt + \int_{x_{v,j}-b}^{x_{v,j}+b} te^{\epsilon/k}dt + \int_{x_{v,j}+b}^{1+b} t\,dt\right)$$
$$= \frac{b(e^{\epsilon/k}-1)}{be^{\epsilon/k}+1} \cdot x_{v,j}. \tag{7}$$

Combining (6) and (7) we conclude

$$\mathbb{E}[\tilde{x}_{v,j}] = \frac{kb(e^{\epsilon/k}-1)}{dbe^{\epsilon/k}+1} \cdot x_{v,j} = C \cdot x_{v,j}. \tag{8}$$

For the variance, we have

$$\mathrm{Var}[\tilde{x}_{v,j}] = \mathbb{E}[\tilde{x}_{v,j}^2] - (\mathbb{E}[\tilde{x}_{v,j}])^2$$
$$= \mathbb{E}[\tilde{x}_{v,j}^2|j \in \mathcal{S}]\Pr[j \in \mathcal{S}] + \mathbb{E}[\tilde{x}_{v,j}^2|j \notin \mathcal{S}]\Pr[j \notin \mathcal{S}] - \mathbb{E}^2[\tilde{x}_{v,j}]$$
$$= \frac{k}{d} \cdot \mathbb{E}[\tilde{x}_{v,j}^2|j \in \mathcal{S}] - (\mathbb{E}[\tilde{x}_{v,j}])^2. \tag{9}$$

Algorithm 3: Backward Push Propagation

Input: graph $G = (\mathcal{V}, \mathcal{E})$, perturbed feature matrix $\tilde{\mathbf{X}}$, decay factor α, convolutional coefficient r, threshold r_{max}
Output: embedding matrix $\tilde{\mathbf{Z}}$

Initialize reserve matrix $\mathbf{Q} \leftarrow \mathbf{0}$ and residue matrix $\mathbf{R} \leftarrow \mathbf{D}^{-r}\tilde{\mathbf{X}}$;
while $\exists v$ *and* $\exists j \in \{0,\ldots,d-1\}$ *s.t.* $|\mathbf{R}(v,j)| > r_{max}$ **do**
  for $u \in \mathcal{N}(v)$ **do**
    $\mathbf{R}(u,j) \leftarrow \mathbf{R}(u,j) + (1-\alpha) \cdot \frac{\mathbf{R}(v,j)}{|\mathcal{N}(u)|}$;
  $\mathbf{Q}(v,j) \leftarrow \mathbf{Q}(v,j) + \alpha \cdot \mathbf{R}(v,j)$;
  $\mathbf{R}(v,j) \leftarrow 0$;
$\tilde{\mathbf{Z}} \leftarrow \mathbf{D}^r \cdot \mathbf{Q}$;
return $\tilde{\mathbf{Z}}$

According to Equation (5), we have

$$\mathbb{E}[\tilde{x}_{v,j}^2|j \in \mathcal{S}] = \frac{\int_{-1-b}^{x_{v,j}-b} t^2\,dt + \int_{x_{v,j}-b}^{x_{v,j}+b} t^2e^{\epsilon/k}\,dt + \int_{x_{v,j}+b}^{1+b} t^2\,dt}{2be^{\epsilon/k}+2}$$
$$= \frac{b^3e^{\epsilon/k}+3b^2+3b+1}{3(be^{\epsilon/k}+1)} + \frac{b(e^{\epsilon/k}-1)}{be^{\epsilon/k}+1} \cdot x_{v,j}^2. \tag{10}$$

Combining (8), (9) and (10) yields

$$\mathrm{Var}[\tilde{x}_{v,j}] = \frac{k(b^3e^{\epsilon/k}+3b^2+3b+1)}{3d(be^{\epsilon/k}+1)} + (C-C^2)x_{v,j}^2. \tag{11}$$

□

3.2 Propagation Module

The propagation module takes the perturbed feature matrix $\tilde{\mathbf{X}}$ as input, which comprises obfuscated feature vectors $\tilde{\mathbf{x}}_v$ for each user $v \in \mathcal{V}$, and outputs the embedding matrix $\tilde{\mathbf{Z}}$. To incorporate neighborhood information, we exploit personalized PageRank as the proximity measure for graph propagation. Formally, the process of graph propagation can be formulated as follows:

$$\tilde{\mathbf{Z}} = \mathbf{\Pi} \cdot \tilde{\mathbf{X}} = \sum_{\ell=0}^{\infty} \alpha(1-\alpha)^\ell \cdot (\mathbf{D}^{r-1}\mathbf{A}\mathbf{D}^{-r})^\ell \cdot \tilde{\mathbf{X}}. \tag{12}$$

We utilize the well-established backward push algorithm [41] as a standard technique to calculate personalized PageRank. In our work, we extend this algorithm to enable efficient graph propagation.

Algorithm 3 illustrates the pseudo-code of the backward push propagation algorithm. Intuitively, the algorithm starts by setting the residue matrix $\mathbf{R} = \mathbf{D}^{-r}\mathbf{A}$ and the reserve matrix $\mathbf{Q} = \mathbf{0}$ (Line 1). Subsequently, a push procedure (Line 2-6) is executed for each node v if the absolute value of the residue entry $\mathbf{R}(v,j)$ exceeds a threshold r_{max} until no such v exists. Specifically, if there is a node v meeting $|\mathbf{R}(v,j)| > r_{max}$, the algorithm increases the residue of each neighbor u by $(1-\alpha) \cdot \frac{\mathbf{R}(v,j)}{|\mathcal{N}(u)|}$ and increases the reserve of node v by $\alpha \times \mathbf{R}(v,j)$. After that, it reset the residue $\mathbf{R}(v,j)$ to 0. Finally, the embedding matrix is $\tilde{\mathbf{Z}} = \mathbf{D}^r \cdot \mathbf{Q}$. The propagation process can be seen as post-processing and thus does not consume additional privacy budget.

3.3 Utility Analysis

In this section, we conduct an in-depth theoretical analysis regarding the utility of the PrivGE framework. Additionally, for comparison, we also present a set of theorems that characterize the utility of alternative mechanisms, including the Laplace, Piecewise and Multi-bit mechanisms.

For each node $v \in \mathcal{V}$, we use $\mathbf{z}_v$ and $\tilde{\mathbf{z}}_v$ to represent the true node embedding vector and the perturbed node embedding vector, respectively. In addition, we define the ℓ_2 norm of $\mathbf{\Pi}_v$ as $\|\mathbf{\Pi}_v\|_2 = (\sum_{u\in\mathcal{V}}(\mathbf{\Pi}(v,u))^2)^{1/2}$ and the ℓ_∞ norm as $\|\mathbf{\Pi}_v\|_\infty = \max_{u\in\mathcal{V}}|\mathbf{\Pi}(v,u)|$. For any dimension $j \in \{1,\ldots,d\}$, according to Equations (2) and (12), we have $z_{v,j} = \sum_{u\in\mathcal{V}}\mathbf{\Pi}(v,u)\cdot x_{u,j}$ and $\tilde{z}_{v,j} = \sum_{u\in\mathcal{V}}\mathbf{\Pi}(v,u)\cdot \tilde{x}_{u,j}$. The following theorem establishes the estimation error of the PrivGE framework.

Theorem 3.3. *Given $\delta > 0$, for any node v, with probability at least $1-\delta$, we have $\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j} - z_{v,j}| = O(\log(d/\delta))$.*

Proof. The variable $\tilde{z}_{v,j} = \sum_{u\in\mathcal{V}} t_u$ is the sum of $|\mathcal{V}|$ independent random variables, where $t_u = \mathbf{\Pi}(v,u)\cdot\tilde{x}_{u,j}$. According to the Algorithm 2, we have $t_u \in [a_u, b_u]$, where $a_u = \mathbf{\Pi}(v,u)\cdot(-b-1)$ and $b_u = \mathbf{\Pi}(v,u)\cdot(b+1)$. Observe that $b_u - a_u \le 2(b+1)$ for any node u. Then by Bernstein's inequality, we have

$$\begin{aligned}&\Pr[|\tilde{z}_{v,j} - \sum\nolimits_{u\in\mathcal{V}}\mathbb{E}[t_u]| > \tau]\\ &< 2\cdot\exp(-\frac{\tau^2}{2\sum_{u\in\mathcal{V}}\mathrm{Var}[t_u] + \frac{4}{3}\tau(b+1)}),\end{aligned} \tag{13}$$

where $\mathbb{E}[t_u] = \mathbf{\Pi}(v,u)\cdot\mathbb{E}[\tilde{x}_{u,j}]$ and $\mathrm{Var}[t_u] = (\mathbf{\Pi}(v,u))^2\cdot\mathrm{Var}[\tilde{x}_{u,j}]$. The asymptotic expressions involving ϵ are evaluated in $\epsilon\to 0$, which yields $b = \frac{\epsilon/k\cdot e^{\epsilon/k} - e^{\epsilon/k}+1}{e^{\epsilon/k}(e^{\epsilon/k}-\epsilon/k-1)} = O(1)$ and $\mathrm{Var}[\tilde{x}_{u,j}] = O(k/d)$. Therefore, we have

$$\begin{aligned}&\Pr[|\tilde{z}_{v,j} - \sum\nolimits_{u\in\mathcal{V}}\mathbb{E}[t_u]| > \tau]\\ &< 2\cdot\exp(-\frac{\tau^2}{O(k/d)\cdot\sum_{u\in\mathcal{V}}(\mathbf{\Pi}(v,u))^2 + \tau\cdot O(1)}).\end{aligned} \tag{14}$$

There exists $\tau = O(\log(d/\delta))$ such that the following inequality holds with at least $1-\delta/d$ probability

$$|\tilde{z}_{v,j} - \sum\nolimits_{u\in\mathcal{V}}\mathbb{E}[t_u]| \le \tau. \tag{15}$$

Observe that

$$\begin{aligned}&|\tilde{z}_{v,j} - \sum\nolimits_{u\in\mathcal{V}}\mathbb{E}[t_u]|\\ &= |\tilde{z}_{v,j} - z_{v,j} + z_{v,j} - \frac{k}{d}\cdot\frac{b(e^{\epsilon/k}-1)}{be^{\epsilon/k}+1}\cdot z_{v,j}|\\ &\ge |\tilde{z}_{v,j} - z_{v,j}| - |z_{v,j} - \frac{k}{d}\cdot\frac{b(e^{\epsilon/k}-1)}{be^{\epsilon/k}+1}\cdot z_{v,j}|.\end{aligned} \tag{16}$$

Since $x_{u,j}\in[-1,1]$ and $\sum_{u\in\mathcal{V}}\mathbf{\Pi}(v,u) = 1$, combining (16) in (15), we have

$$|\tilde{z}_{v,j} - z_{v,j}| \le \tau + |z_{v,j} - \frac{k}{d}\cdot\frac{b(e^{\epsilon/k}-1)}{be^{\epsilon/k}+1}\cdot z_{v,j}| = O(\log(d/\delta)).$$

By the union bound, $\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j} - z_{v,j}| \le O(\log(d/\delta))$ holds with at least $1-\delta$ probability. □

Next, we present a series of theorems that delineate the utility of alternative mechanisms.

Theorem 3.4. *Assume that the perturbation function is the Laplace mechanism. Given $\delta > 0$, for any node $v\in\mathcal{V}$ with probability as least $1-\delta$, we have*

$$\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j} - z_{v,j}| = \begin{cases} O(\frac{d\log(d/\delta)}{\epsilon}), \delta < 2de^{-\frac{\|\mathbf{\Pi}_v\|_2^2}{2\|\mathbf{\Pi}_v\|_\infty^2}},\\ O(\frac{d\sqrt{\log(d/\delta)}}{\epsilon}), \delta \ge 2de^{-\frac{\|\mathbf{\Pi}_v\|_2^2}{2\|\mathbf{\Pi}_v\|_\infty^2}}.\end{cases}$$

To prove Theorem 3.4, our initial step entails demonstrating the sub-exponential nature of the Laplace random variable. Thus, we present the definition of sub-exponential random variables.

Definition 3.5 (Sub-exponential distirbutions)**.** A random variable η is said to be sub-exponential with parameter ν (denoted $\eta \sim subE(\nu)$) if $\mathbb{E}[\eta] = 0$, and its moment generating function (MGF) satisfies

$$\mathbb{E}[e^{s\eta}] \le e^{s^2\nu^2/2}, \forall |s| \le 1/\nu. \tag{17}$$

Then, the following lemma confirms that the Laplace distribution is sub-exponential.

Lemma 3.6. *If a random variable η obeys the Laplace distribution with parameter λ, then η is sub-exponential: $\eta\sim subE(2\lambda)$.*

Proof. Without loss of generality, we consider a centered random variable $\eta\sim Lap(1)$. Its moment generating function (MGF) is given by $\mathbb{E}[e^{s\eta}] = \frac{1}{1-s^2}$ for $|s|<1$. Notably, one of the upper bounds on the MGF is $\mathbb{E}[e^{s\eta}] \le e^{2s^2}$ for $|s| < \frac{1}{2}$. This indicates that $\eta\sim subE(2)$.

Next, we extend the result to the Laplace distribution with parameter λ. Note that if $\eta\sim Lap(1)$, then $\lambda\eta\sim Lap(\lambda)$. As a result, the upper bound on the MGF becomes $\mathbb{E}[e^{s\lambda\eta}] \le e^{2\lambda^2 s^2}$ for $|s| < \frac{1}{2\lambda}$, from which we conclude that the distribution $Lap(\lambda)$ is sub-exponential with parameter $\nu = 2\lambda$. □

Now, we prove Theorem 3.4.

Proof. If the perturbation function is the Laplace mechanism, we have $\tilde{z}_{v,j} - z_{v,j} = \sum_{u\in\mathcal{V}}\mathbf{\Pi}(v,u)\cdot\eta_u$, where $\eta_u\sim Lap(2d/\epsilon)$. Since the random variable $\eta_u\sim subE(4d/\epsilon)$, according to the Bernstein's inequality we have

$$\begin{aligned}\Pr[|\tilde{z}_{v,j} - z_{v,j}| > \tau] &= \Pr[|\sum\nolimits_{u\in\mathcal{V}}\mathbf{\Pi}(v,u)\cdot\eta_u| > \tau]\\ &< \begin{cases} 2e^{-\tau^2\epsilon^2/32d^2\|\mathbf{\Pi}_v\|_2^2}, & \text{if } 0\le\tau\le\frac{4d\|\mathbf{\Pi}_v\|_2^2}{\epsilon\|\mathbf{\Pi}_v\|_\infty},\\ 2e^{-\tau\epsilon/8d\|\mathbf{\Pi}_v\|_\infty}, & \text{if } \tau > \frac{4d\|\mathbf{\Pi}_v\|_2^2}{\epsilon\|\mathbf{\Pi}_v\|_\infty}.\end{cases}\end{aligned} \tag{18}$$

First, consider the case where $0\le\tau\le\frac{4d\|\mathbf{\Pi}_v\|_2^2}{\epsilon\|\mathbf{\Pi}_v\|_\infty}$. Let $\delta/d = 2e^{-\tau^2\epsilon^2/32d^2\|\mathbf{\Pi}_v\|_2^2}$. Solving for τ, we obtain $\tau = \frac{4d}{\epsilon}\cdot\sqrt{2log(2d/\delta)}\cdot\|\mathbf{\Pi}_v\|_2$. In this case, δ must satisfy the condition $\delta\ge 2de^{-\frac{\|\mathbf{\Pi}_v\|_2^2}{2\|\mathbf{\Pi}_v\|_\infty^2}}$. Utilizing the union bound and evaluating the asymptotic expressions for small ϵ (i.e., $\epsilon\to 0$), we can deduce that when $\delta\ge 2de^{-\frac{\|\mathbf{\Pi}_v\|_2^2}{2\|\mathbf{\Pi}_v\|_\infty^2}}$, there exists $\tau = O(d\sqrt{log(d/\delta)}/\epsilon)$ such that the inequality $\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j} - z_{v,j}|\le\tau$ holds with at least $1-\delta$ probability.

Second, suppose $\tau\ge\frac{4d\|\mathbf{\Pi}_v\|_2^2}{\epsilon\|\mathbf{\Pi}_v\|_\infty}$. Similar to the first case, let $\delta/d = 2e^{-\tau\epsilon/8d\|\mathbf{\Pi}_v\|_\infty}$. Solving the above for τ, we have $\tau = \frac{8d}{\epsilon}\cdot log(2d/\delta)\cdot\|\mathbf{\Pi}_v\|_\infty$. For this case to be valid, δ must satisfy $\delta < 2de^{-\frac{\|\mathbf{\Pi}_v\|_2^2}{2\|\mathbf{\Pi}_v\|_\infty^2}}$. Similarly, according to union bound, when $\delta < 2de^{-\frac{\|\mathbf{\Pi}_v\|_2^2}{2\|\mathbf{\Pi}_v\|_\infty^2}}$, there exists $\tau = O(dlog(d/\delta)/\epsilon)$ such that the inequality $\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j} - z_{v,j}|\le\tau$ holds with at least $1-\delta$ probability. □

Theorem 3.7. *Assume that the perturbation function is the Piecewise mechanism or the Multi-bit mechanism. Given $\delta > 0$, for any node v, with probability at least $1-\delta$, we have $\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j} - z_{v,j}| = O(\frac{d\log(d/\delta)}{\epsilon})$.*

Proof. If the perturbation function is Piecewise mechanism, for any node u, $\mathbf{\Pi}(v,u)\cdot(\tilde{x}_{u,j} - x_{u,j})$ is a zero-mean random variable and its variance is $(\mathbf{\Pi}(v,u))^2\,\mathrm{Var}[\tilde{x}_{u,j}]$. Besides, the inequality

$|\mathbf{\Pi}(v,u)\cdot(\tilde{x}_{u,j}-x_{u,j})| \leq \frac{2de^{\epsilon/2k}}{k(e^{\epsilon/2k}-1)}$ always holds. Then by Bernstein's inequality, we find that

$$\Pr[|\tilde{z}_{v,j}-z_{v,j}|>\tau] = \Pr[|\sum_{u\in\mathcal{V}}\mathbf{\Pi}(v,u)\cdot(\tilde{x}_{u,j}-x_{u,j})|>\tau] < 2\exp(\frac{-\tau^2}{2\sum_{u\in\mathcal{V}}(\mathbf{\Pi}(v,u))^2\,\mathrm{Var}[\tilde{x}_{u,j}]+\frac{\tau\cdot 4de^{\epsilon/2k}}{3k(e^{\epsilon/2k}-1)}}). \quad (19)$$

Note that asymptotic expressions involving ϵ are in the sense of $\epsilon \leftarrow 0$. Thus, we can deduce that $\mathrm{Var}[\tilde{x}_{u,j}] = O(\frac{kd}{\epsilon^2})$ and $\frac{de^{\epsilon/2k}}{k(e^{\epsilon/2k}-1)} = O(\frac{d}{\epsilon})$. Since $\mathbf{\Pi}(v,u)\in[0,1]$ and $\sum_{u\in\mathcal{V}}\mathbf{\Pi}(v,u)=1$, we can obtain that

$$\Pr[|\tilde{z}_{v,j}-z_{v,j}|>\tau] < 2\cdot\exp(\frac{-\tau^2}{O(\frac{kd}{\epsilon^2})+\tau\cdot O(\frac{d}{\epsilon})}). \quad (20)$$

By applying the union bound, we can ensure that $\max_{j\in\{1,\ldots,d\}}|\tilde{z}_{v,j}-z_{v,j}|\leq\tau$ holds with at least $1-\delta$ probability by setting $\delta/d = 2\cdot\exp(\frac{-\tau^2}{O(\frac{kd}{\epsilon^2})+\tau\cdot O(\frac{d}{\epsilon})})$. Solving the above for τ, we have $\tau = O(\frac{d\log(d/\delta)}{\epsilon})$.

Given that the Multi-bit mechanism belongs to the same category as the Piecewise mechanism (both being bounded mechanisms) and the perturbed value is also unbiased, the analysis process of the Multi-bit mechanism closely resembles that of the Piecewise mechanism. Due to space constraints, we omit the proof of the Multi-bit mechanism. □

According to the utility analysis presented in Theorem 3.3 and Theorem 3.4, we observe that the HDS mechanism can yield a higher utility than the Laplace mechanism. Furthermore, as indicated by Theorem 3.3 and Theorem 3.7, the proposed HDS mechanism offers superior error bounds compared to the Piecewise and Multi-bit mechanisms, reducing them from $O(\frac{d\log(d/\delta)}{\epsilon})$ to $O(\log(d/\delta))$.

4 Experiments

4.1 Experimental Setup

Dataset. Our experiments use five real-world datasets: two citation networks (i.e., Cora and Pubmed [20]), one social networks (i.e., LastFM [32]) and two web graphs (i.e., Facebook and Wikipedia [31]). Table 1 summarizes the statistics of the datasets used for evaluation. Following previous works [24, 32], we randomly split each dataset into three portions for node classification: 50% for training, 25% for validation, and 25% for testing. Since the labels of nodes in the LastFM dataset are highly imbalanced, we restrict the classes to the top 10 with the most samples. For link prediction, the edges are divided into 85% for training, 10% for testing, and 5% for validation. And we sample the same number of non-existing links in each group as negative links. Note that the LDP mechanisms perturb the node features of all training, validation, and test sets.

Model Implementation Details. For node classification, we first obtain the node embedding matrix $\tilde{\mathbf{Z}}$ via the PrivGE framework. Then the matrix $\tilde{\mathbf{Z}}$ is fed into a model composed of a multi-layer perceptron (MLP) followed by a softmax function for prediction:

$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathrm{MLP}(\tilde{\mathbf{Z}};\mathbf{\Theta})), \quad (21)$$

where $\hat{\mathbf{Y}}$ represents the posterior class probabilities and $\mathbf{\Theta}$ is the learnable model parameters. Moreover, we adopt the cross-entropy loss as the optimization function to train the classification module.

To perform link prediction, we first extract edge features based on node representations. Specifically, for each node pair $(u,v)\in\mathcal{E}$, we combine the node vectors $\tilde{\mathbf{z}}_u$ and $\tilde{\mathbf{z}}_v$ via the Hadamard product to compose the edge vector. Then, we take the constructed edge feature vectors as input and train a logistic regression classifier to predict the presence or absence of edges. Following existing works [25, 38], we use the pairwise objective loss function to optimize the model.

Table 1: Dataset Statistics

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,278	1,433	7
Pubmed	19,717	44,324	500	3
Lastfm	7,083	25,814	7,842	10
Facebook	22,470	170,912	4,714	4
Wikipedia	11,631	170,845	13,183	2

For each task, we train the model on the training set and tune the hyperparameters based on the model's performance in the validation set. First, we define the search space of hyperparameters. Then we fix the privacy budget $\epsilon = 1.0$ and utilize NNI [26], an automatic hyperparameter optimization tool, to determine the optimal choices. We use the Adam optimizer to train the models and select the best model to evaluate the testing set.

Competitors and Evaluation Metric. In our experiments, we compare the performance of the HDS mechanism with the Laplace mechanism (LP) [9], the Piecewise mechanism (PM) [37] and the Multi-bit mechanism (MB) [32]. Following existing works [20, 32], we use *accuracy* to evaluate the node classification performance and *AUC* to evaluate the link prediction performance. Unless otherwise stated, we run each algorithm ten times for both tasks and report the mean and standard deviation values.

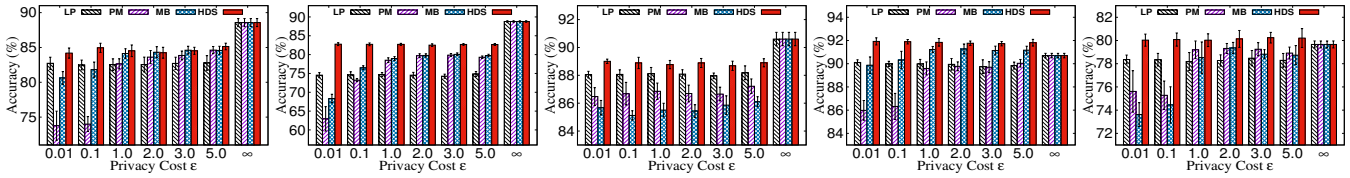
Software and Hardware. We implement our algorithm in PyTorch and C++. All the experiments are conducted on a machine with an NVIDIA GeForce RTX 3080Ti, an AMD Ryzen Threadripper PRO 3995WX CPU, and 256GB RAM. Our code is available online[3].

4.2 Experimental Results

Node Classification. In the first set of experiments, we evaluate the performance of different mechanisms under different privacy budgets for node classification. The privacy budget varies over the range $\{0.01, 0.1, 1.0, 2.0, 3.0, 5.0, \infty\}$, and we report the accuracy of each method under each privacy budget, as shown in Figure 2. Note that the error bars in the Figure 2 represent the standard deviation. In addition, the case where $\epsilon = \infty$ is provided for comparison with non-private baselines, where node features are directly employed without any perturbation.

The experimental results lead us to conclude that our proposed method demonstrates robustness to the perturbations and achieves comparable performance to the non-private baseline across all datasets. For instance, on the LastFM dataset, the HDS mechanism achieves an accuracy of about 89.0% at $\epsilon = 0.01$, with only a 1.6% decrease compared to the non-private method ($\epsilon = \infty$). On the Cora dataset, the accuracy of HDS mechanism at $\epsilon = 0.01$ is approximately 84.2%, just 4.3% lower than the non-private method. Similarly, on the Pubmed dataset, when $\epsilon = 0.01$, our method loses less than 6.0% in accuracy over the non-private baseline. Interestingly, on the Facebook and Wikipedia datasets, our framework

[3] https://github.com/Zening-Li/PrivGE



(a) Cora (b) Pubmed (c) LastFM (d) Facebook (e) Wikipedia

Figure 2: Trade-offs between privacy and accuracy under different LDP mechanisms in node classification. Note that the error bars represent the standard deviation and the results for ∞ denote the accuracy of the non-private baselines.

Table 2: Trade-offs between privacy and AUC under different LDP mechanisms in link prediction. Note that the results for ∞ denote the AUC of the non-private baselines.

Dataset	Mech.	$\epsilon = 1.0$	$\epsilon = 2.0$	$\epsilon = 3.0$	$\epsilon = 5.0$
Cora	LP	75.2 ± 3.1	76.0 ± 3.5	75.0 ± 3.7	75.2 ± 3.6
	PM	73.2 ± 3.2	78.2 ± 1.1	77.8 ± 1.7	75.2 ± 2.1
$\epsilon = \infty$	MB	76.0 ± 1.8	78.5 ± 1.4	78.5 ± 1.1	76.6 ± 0.9
93.1 ± 0.3	HDS	**82.4 ± 1.5**	**82.7 ± 0.8**	**82.5 ± 0.9**	**82.3 ± 0.8**
Pubmed	LP	62.8 ± 0.6	63.2 ± 1.2	64.3 ± 0.8	64.4 ± 0.5
	PM	79.0 ± 1.0	77.0 ± 1.2	76.7 ± 1.5	74.9 ± 0.5
$\epsilon = \infty$	MB	79.4 ± 0.8	77.9 ± 1.0	78.4 ± 0.7	77.2 ± 0.7
97.5 ± 0.1	HDS	**79.5 ± 1.5**	**79.6 ± 1.6**	**80.1 ± 0.2**	**79.6 ± 1.5**
Lastfm	LP	74.4 ± 6.3	75.4 ± 6.0	77.0 ± 4.8	73.8 ± 3.5
	PM	67.9 ± 0.9	71.7 ± 6.2	72.0 ± 3.9	80.1 ± 1.4
$\epsilon = \infty$	MB	75.7 ± 3.1	78.2 ± 3.3	78.9 ± 4.6	78.0 ± 3.3
95.9 ± 0.1	HDS	**91.7 ± 1.3**	**92.0 ± 0.1**	**92.0 ± 0.1**	**92.0 ± 0.1**
Facebook	LP	81.92 ± 3.2	84.7 ± 2.7	85.6 ± 1.1	86.0 ± 1.2
	PM	92.1 ± 0.5	93.2 ± 0.5	93.3 ± 0.3	89.9 ± 0.3
$\epsilon = \infty$	MB	92.6 ± 3.8	92.8 ± 2.3	93.1 ± 2.2	87.7 ± 0.9
95.6 ± 0.1	HDS	**96.7 ± 0.1**	**96.7 ± 0.1**	**96.6 ± 0.1**	**96.6 ± 0.1**
Wikipedia	LP	72.2 ± 2.8	74.9 ± 4.7	76.1 ± 2.5	75.7 ± 3.5
	PM	76.9 ± 5.3	76.6 ± 4.6	76.8 ± 3.5	77.9 ± 1.7
$\epsilon = \infty$	MB	76.6 ± 3.6	77.2 ± 1.4	78.2 ± 1.3	78.4 ± 1.1
98.5 ± 0.1	HDS	**99.1 ± 0.1**	**99.1 ± 0.1**	**99.1 ± 0.1**	**99.1 ± 0.1**

outperforms the non-private setting by about 1.2 and 0.4 percentage points, respectively. We conjecture that the observed experimental results are mainly due to the fact that the injected noise provides the model with strong generalization capabilities. Additionally, the small standard deviation in our experimental results indicates that our method maintains stable performance under various privacy budgets. It consistently exhibits similar accuracy across distinct privacy budget values from 0.01 to 1.0, which serves as a powerful demonstration of its robustness to perturbations.

Second, our HDS mechanism consistently outperforms the other mechanisms in almost all cases, particularly under smaller privacy budgets, which is consistent with the theoretical analysis in Theorem 3.3. For instance, at $\epsilon = 0.01$, HDS achieves approximately 8.2% higher accuracy than the best competitor, the Laplace mechanism, on the Pubmed dataset. Similarly, at $\epsilon = 0.01$, our approach outperforms LP, PM, and MB by 1.7%, 4.4% and 6.4%, respectively, on the Wikipedia dataset. This remarkable performance advantage can be attributed to the fact that our proposed perturbation mechanism can provide more concentrated perturbations compared to the Laplace, Piecewise, and Multi-bit mechanisms. Additionally, we can observe that even the simplest method, the Laplace mechanism, sometimes outperforms the Piecewise and Multi-bit mechanisms, especially when a smaller privacy budget is allocated.

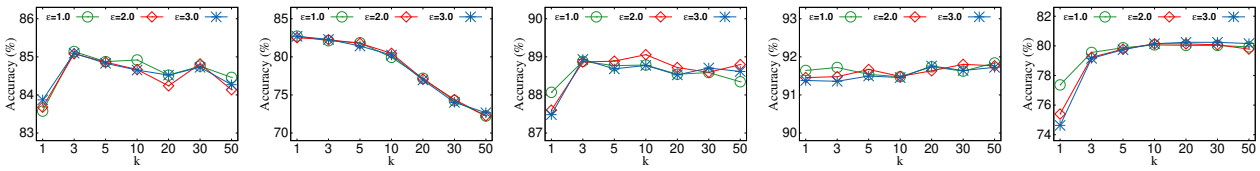
Link Prediction. In the second set of experiments, we shift our focus to another important task: link prediction. To examine the performance of PrivGE under different privacy budgets, we vary ϵ from 1.0 to 5.0. The experimental results are summarized in Table 2.

Similar to the node classification task, the HDS mechanism outperforms the other three methods in terms of the AUC score. At $\epsilon = 1.0$, the AUC scores of our proposed mechanism outperform the best competitors by about 6.5, 16.0, and 4.0 on the datasets Cora, LastFM, and Facebook, respectively. In addition, even considering strong privacy guarantees, such as $\epsilon = 1.0$, the technique still achieves acceptable AUC scores on all five datasets, especially on the LastFM, Facebook and Wikipedia datasets, where the AUC scores are above 90%. These outcomes underline the effectiveness of the proposed method for the link prediction task. Similar to the node classification task, our perturbation mechanism yields higher AUC scores than the non-private baseline on both Facebook and Wikipedia datasets, which further confirms the efficiency of the proposed solution. Unlike the observations obtained from the node classification, in the link prediction task, we notice that the Laplace mechanism is inferior to the Piecewise and Multi-bit mechanisms in most cases. In addition, the performance of the three competitors becomes very unstable since a lot of noise is injected. As a result, in some cases, the performance with a small privacy budget yet outperforms when the privacy budget is large. In summary, the experimental results emphasize that our approach can achieve a better trade-off between privacy and utility.

Parameter Analysis. In this experiment, we investigate the impact of the parameter k on the performance of our proposed method. The parameter k is a hyperparameter that controls the size of the subset of dimensions that are randomly perturbed in the HDS mechanism. We vary k within $\{1, 3, 5, 10, 20, 30, 50\}$, and evaluate the results under different privacy budgets $\epsilon \in \{1.0, 2.0, 3.0\}$ for both node classification and link prediction tasks.

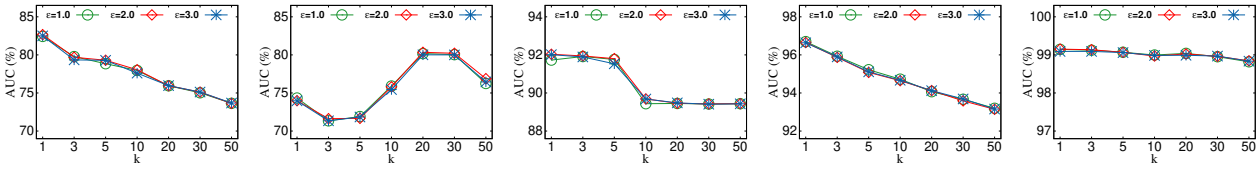
As for node classification, the experimental results are illustrated in Figure 3. We observe that the performance of our mechanism remains relatively stable for all values of k on Cora, LastFM, and Facebook datasets, varying up and down by no more than 2.0%. However, on the Pubmed dataset, the approach performs better when k is smaller, especially when $k \leq 5$. In contrast, on the Wikipedia dataset, the performance of HDS improves as the number of samples increases, particularly when $k \geq 10$. This indicates that the performance of HDS is sensitive to the value of k on the Pubmed and Wikipedia datasets, whereas it is relatively insensitive to k on the other three datasets.

As for link prediction, the experimental results are illustrated in Figure 4. Our framework performs better for small values of k



(a) Cora (b) Pubmed (c) LastFM (d) Facebook (e) Wikipedia

Figure 3: Effect of the sampling parameter k on the performance of PrivGE for node classification.



(a) Cora (b) Pubmed (c) LastFM (d) Facebook (e) Wikipedia

Figure 4: Effect of the sampling parameter k on the performance of PrivGE for link prediction.

on Cora, LastFM, and Facebook datasets, while it achieves better results for large k values on the Pubmed dataset. On the Wikipedia dataset, our method is stable across all values of k. Generally, the optimal value of k depends on the specific dataset and task.

5 Related work

DP on Graph Analysis. DP has become the standard for privacy protection in various data analysis tasks [7, 18]. In centralized DP, most graph analysis tasks revolve around the computation of diverse statistics such as degree distributions and subgraph counts [3, 6, 17, 19, 28, 42]. In addition, research extends to other graph problems under DP, such as node subset release for vertex cover [12] and densest subgraph [4, 27], and synthetic graph generation [16]. However, all these centralized DP methods require possession of all user data, which suffers from the data breach.

Local DP assumes an untrusted data collector and recently has attracted much attention in graph analysis. Ye et al. [40] propose a method to estimate graph metrics. Qin et al. [29] develop a multi-phase approach to generate synthetic decentralized social networks under the notion of LDP. In addition, a series of studies have focused on subgraph counting [13, 14, 34]. Sun et al. [34] develop a multi-phase framework under decentralized DP, which assumes that each user allows her friends to see all her connections. Imola et al. [13] introduce an additional round of interaction between users and the data collector to reduce the estimation error, and [14] employs edge sampling to improve the communication efficiency.

DP on Graph Learning. To address the privacy concerns in graph learning, DP has been widely used to protect sensitive data. For instance, Xu et al. [39] propose a DP algorithm for graph embedding that uses the objective perturbation mechanism on the loss function of the matrix factorization. Zhang et al. [43] devise a perturbed gradient descent method based on the Lipschitz condition. Epasto et al. [10] develop an approximate method for computing personalized PageRank vectors with differential privacy and extend this algorithm to graph embedding. In recent years, DP has also been used to provide formal privacy assurances for Graph Neural Networks [2, 22, 33]. All these methods assume that there is a trusted data curator, making them susceptible to data leakage issues and unsuitable for decentralized graph analysis applications.

To tackle these issues, some studies have focused on leveraging LDP to train Graph Neural Networks [15, 24, 32, 44]. To be specific, Zhang et al. [44] propose an algorithm for recommendation systems, which utilizes LDP to prevent users from attribute inference attacks. LPGNN [32] assumes that node features are private and the data curator has access to the graph structure, where the scenario is similar to ours. In this setting, each user perturbs their features through the Multi-bit mechanism. However, this mechanism introduces much noise to the data that can degrade the performance. Analogously, [15, 24] also utilize the Multi-bit mechanism to preserve the privacy of node features. Unlike these works, we propose an improved mechanism to achieve LDP for graph embedding and provide a detailed utility analysis for our method.

6 Conclusion

In this paper, we propose PrivGE, a privacy-preserving graph embedding framework based on local differential privacy. To this end, we propose an LDP mechanism called HDS to protect the privacy of node features. Then, to avoid neighborhood explosion and over-smoothing problems, we decouple the feature transformation from graph propagation and employ personalized PageRank as a proximity measure to learn node representations. Importantly, we perform a novel and comprehensive theoretical analysis of the privacy and utility of the PrivGE framework. Extensive experiments conducted on real-world datasets demonstrate that our proposed method establishes state-of-the-art performance and achieves better privacy-utility trade-offs on both node classification and link prediction tasks. In future work, we plan to extend our work to protect the privacy of the graph structure as well. Another future direction is to develop a graph embedding algorithm to protect the edge privacy for non-attributed graphs that contain only node IDs and edges.

Acknowledgments

This work was partially supported by (i) the National Key Research and Development Program of China 2021YFB3301301,(ii) NSFC-Grants U2241211and 62072034. Rong-Hua Li is the corresponding author of this paper.

References

[1] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *KDD*. 2464–2473.

[2] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2021. Node-level differentially private graph neural networks. *arXiv preprint arXiv:2111.15521* (2021).

[3] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *SIGMOD*. 123–138.

[4] Laxman Dhulipala, Quanquan C Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, and Shangdi Yu. 2022. Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs. In *FOCS*. 754–765.

[5] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *NeurIPS*. 3571–3580.

[6] Xiaofeng Ding, Shujun Sheng, Huajian Zhou, Xiaodong Zhang, Zhifeng Bao, Pan Zhou, and Hai Jin. 2021. Differentially private triangle counting in large graphs. *TKDE* (2021).

[7] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *FOCS*. 429–438.

[8] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2018. Minimax optimal procedures for locally private estimation. *J. Amer. Statist. Assoc.* 113, 521 (2018), 182–201.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.

[10] Alessandro Epasto, Vahab Mirrokni, Bryan Perozzi, Anton Tsitsulin, and Peilin Zhong. 2022. Differentially private graph learning via sensitivity-bounded personalized pagerank. *NeurIPS* (2022), 22617–22627.

[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*. 1054–1067.

[12] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially private combinatorial optimization. In *SODA*. 1106–1125.

[13] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2021. Locally differentially private analysis of graph statistics. In *USENIX Security*. 983–1000.

[14] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-efficient triangle counting under local differential privacy. In *USENIX Security*. 537–554.

[15] Hongwei Jin and Xun Chen. 2022. Gromov-wasserstein discrepancy with local differential privacy for distributed structural graphs. In *IJCAI*. 2115–2121.

[16] Zach Jorgensen, Ting Yu, and Graham Cormode. 2016. Publishing attributed social graphs with formal privacy guarantees. In *SIGMOD*. 107–122.

[17] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *VLDB* 4, 11 (2011), 1146–1157.

[18] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SICOMP* 40, 3 (2011), 793–826.

[19] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *TCC*. 457–476.

[20] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.

[22] Aashish Kolluri, Teodora Baluta, Bryan Hooi, and Prateek Saxena. 2022. LPGNet: Link private graph networks for node classification. In *CCS*. 1813–1827.

[23] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. 2020. Estimating numerical distributions under local differential privacy. In *SIGMOD*. 621–635.

[24] Wanyu Lin, Baochun Li, and Cong Wang. 2022. Towards private learning on decentralized graphs with local differential privacy. *TIFS* 17 (2022), 2936–2946.

[25] Jingsong Lv, Zhao Li, Hongyang Chen, Yao Qi, and Chunqi Wu. 2022. Path-aware siamese graph neural network for link prediction. *arXiv preprint arXiv:2208.05781* (2022).

[26] Microsoft. 2021. *Neural Network Intelligence*. https://github.com/microsoft/nni

[27] Dung Nguyen and Anil Vullikanti. 2021. Differentially private densest subgraph detection. In *ICML*. 8140–8151.

[28] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *STOC*. 75–84.

[29] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *CCS*. 425–438.

[30] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. Netsmf: Large-scale network embedding as sparse matrix factorization. In *WWW*. 1509–1520.

[31] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[32] Sina Sajadmanesh and Daniel Gatica-Perez. 2021. Locally private graph neural networks. In *CCS*. 2130–2145.

[33] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2023. Gap: Differentially private graph neural networks with aggregation perturbation. In *USENIX Security*.

[34] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui Wang, and Ting Yu. 2019. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *CCS*. 703–717.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[36] Hanzhi Wang, Mingguo He, Zhewei Wei, Sibo Wang, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. Approximate graph propagation. In *KDD*. 1686–1696.

[37] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *ICDE*. 638–649.

[38] Zhitao Wang, Yong Zhou, Litao Hong, Yuanhang Zou, and Hanjing Su. 2021. Pairwise learning for neural link prediction. *arXiv preprint arXiv:2112.02936* (2021).

[39] Depeng Xu, Shuhan Yuan, Xintao Wu, and HaiNhat Phan. 2018. DPNE: Differentially private network embedding. In *PAKDD*. 235–246.

[40] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. 2020. Towards locally differentially private generic graph metric estimation. In *ICDE*. 1922–1925.

[41] Yuan Yin and Zhewei Wei. 2019. Scalable graph embeddings via sparse transpose proximities. In *KDD*. 1429–1437.

[42] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *SIGMOD*. 731–745.

[43] Sen Zhang and Weiwei Ni. 2019. Graph embedding matrix sharing with differential privacy. *IEEE Access* 7 (2019), 89390–89399.

[44] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Lizhen Cui, and Xiangliang Zhang. 2021. Graph embedding for recommendation against attribute inference attacks. In *WWW*. 3002–3014.

[45] Xingyi Zhang, Kun Xie, Sibo Wang, and Zengfeng Huang. 2021. Learning based proximity matrix factorization for node embedding. In *KDD*. 2243–2253.