PrivDPR: Synthetic Graph Publishing with Deep PageRank under Differential Privacy

Sen Zhang
The Hong Kong Polytechnic University
Hong Kong, China
senzhang@polyu.edu.hk

Haibo Hu
The Hong Kong Polytechnic University
Hong Kong, China
haibo.hu@polyu.edu.hk

Qingqing Ye*
The Hong Kong Polytechnic University
Hong Kong, China
qqing.ye@polyu.edu.hk

Jianliang Xu
Hong Kong Baptist University
Hong Kong, China
xujl@comp.hkbu.edu.hk

Abstract

The objective of privacy-preserving synthetic graph publishing is to safeguard individuals' privacy while retaining the utility of original data. Most existing methods focus on graph neural networks under differential privacy (DP), and yet two fundamental problems in generating synthetic graphs remain open. First, the current research often encounters high sensitivity due to the intricate relationships between nodes in a graph. Second, DP is usually achieved through advanced composition mechanisms that tend to converge prematurely when working with a small privacy budget. In this paper, inspired by the simplicity, effectiveness, and ease of analysis of PageRank, we design PrivDPR, a novel privacy-preserving deep PageRank for graph synthesis. In particular, we achieve DP by adding noise to the gradient for a specific weight during learning. Utilizing weight normalization as a bridge, we theoretically reveal that increasing the number of layers in PrivDPR can effectively mitigate the high sensitivity and privacy budget splitting. Through formal privacy analysis, we prove that the synthetic graph generated by PrivDPR satisfies node-level DP. Experiments on real-world graph datasets show that PrivDPR preserves high data utility across multiple graph structural properties.

CCS Concepts

• **Security and privacy → Data anonymization and sanitization**.

Keywords

Differential Privacy; Graph Synthesis; PageRank

ACM Reference Format:
Sen Zhang, Haibo Hu, Qingqing Ye, and Jianliang Xu. 2025. PrivDPR: Synthetic Graph Publishing with Deep PageRank under Differential Privacy. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709334

*Corresponding author

KDD '25, August 3–7, 2025, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709334

1 Introduction

Numerous real-world applications, such as social networks [22], email networks [23], and voting networks [26], are empowered by graphs and graph analysis [35]. For instance, Facebook leverages social network analysis to offer friend recommendations based on the connections between different users [18]. While the benefits are indisputable, direct publication of graph data potentially results in individual privacy being exposed by different types of privacy attacks [15]. Hence, it is crucial to sanitize graph data before making it publicly available.

Differential privacy (DP) [8] is an extensively studied statistical privacy model thanks to its rigorous mathematical privacy framework. DP can be applied to graph data in two common ways [14]: edge-level DP and node-level DP. Edge-level DP considers two graphs as neighbors if they differ by a single edge, while node-level DP considers two graphs as neighbors if they differ by the edges connected to a single node. Satisfying node-level DP can be challenging as varying one node could result in the removal of $N-1$ edges in the worst case, where N denotes the number of nodes. With the development of differentially private deep learning [1], most existing methods [4, 32, 41, 43, 51] focus on generating synthetic graph data by privatizing deep graph generation models. The advanced composition mechanisms, such as moments accountant (MA) [1], are employed to address excessive splitting of privacy budget during optimization, ensuring that the focus remains on mitigating high sensitivity in graph data. For example, Yang *et al.* [43] propose two solutions, namely differentially private GAN (DPGGAN) and differentially private VAE (DPGVAE), which address high sensitivity by enhancing MA. However, they only achieve weak edge-level DP. Recently, a number of methods [4, 32, 51] focus on Graph Neural Networks (GNNs) under node-level RDP, which often mitigate high sensitivity by bounded-degree strategies. Despite their success, these methods achieve DP based on advanced composition mechanisms and tend to converge prematurely, particularly when the privacy budget is small. This issue results in decreased performance in terms of both privacy and utility.

Over the past two decades, PageRank [28] has been widely used in graph mining and learning to evaluate node rankings, thanks

to its simplicity, effectiveness, and ease of analysis. In this paper, we propose a novel privacy-preserving deep PageRank approach for graph synthesis, namely PrivDPR. A naive method for ensuring privacy is to clip the gradients for all weights and then add noise to them. However, this method often suffers from high sensitivity due to complex node relationships, leading to poor utility. To tackle this issue, inspired by the fact that graph synthesis only requires a specific weight instead of all weights, we explore the relationship between the number of layers and sensitivity. Instead of directly reducing high sensitivity, our core idea is to use weight normalization as a bridge to theoretically demonstrate that increasing the number of layers effectively addresses the challenges associated with high sensitivity and privacy budget splitting. Moreover, through formal privacy analysis, we provide evidence that the synthetic graphs generated by PrivDPR satisfy node-level DP requirements. Extensive experiments on four real graph datasets highlight PrivDPR's ability to effectively preserve essential structural properties of the original graphs.

To summarize, this paper makes three main contributions:

- We present PrivDPR, a novel method for privately synthesizing graphs through the design of a deep PageRank. This method can preserve high data utility while ensuring (ϵ, δ)-node-level DP.
- Instead of directly reducing high sensitivity, we utilize weight normalization as a bridge to explore the relationship between the number of layers and sensitivity. We reveal that increasing the number of layers can effectively address issues arising from high sensitivity and privacy budget splitting.
- Extensive experiments on real graph datasets demonstrate that our solution outperforms existing state-of-the-art methods significantly across eight distinct graph utility metrics and two classical downstream tasks.

The remaining sections of this paper are organized as follows. In Section 2, we provide the preliminaries of our proposed solution. Section 3 presents the problem definition and introduces existing solutions. In Section 4, we discuss the sanitization solution. The privacy and time complexity analysis are presented in Section 5. The comprehensive experimental results are presented in Section 6. We review the related work in Section 7. Finally, we conclude this paper in Section 8.

2 Preliminaries

In this section, we will provide a concise review of the concepts of DP and PageRank. The mathematical notations used throughout this paper are summarized in Table 1.

2.1 Differential Privacy

DP is the prevailing concept of privacy for algorithms on statistical databases. Informally, DP limits the change in output distribution of a mechanism when there is a slight change in its input. In graph data, the concept of neighboring databases is established using two graph datasets, denoted as G and G'. These datasets are considered neighbors if their dissimilarity is limited to at most one edge or node.

Definition 1 (Edge (Node)-Level DP [14]). *A graph analysis mechanism $\mathcal{A}$ achieves (ϵ, δ)-edge (node)-level 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 $O \subseteq Range(\mathcal{A})$, we have $\mathbb{P}[\mathcal{A}(G) \in O] \leq \exp(\epsilon) \cdot \mathbb{P}[\mathcal{A}(G') \in O] + \delta$.*

Table 1: Common Symbols and Definitions

Symbol	Description
ϵ, δ	Differential privacy parameters
$G, \widetilde{G}$	Original and synthetic graphs
G, G'	Any two neighboring graph datasets
d_i^{in}, d_i^{out}	In-degree and out-degree of node i
$\mathcal{A}$	A randomized algorithm
V, E	Set of nodes and edges of G
N	Number of nodes in G
$\mathbf{x}, \mathbf{y}$	Lowercase letters denote vectors
$\mathbf{X}, \mathbf{Y}$	Bold capital letters denote matrices
$\|\mathbf{x}\|_2$, $\|\mathbf{X}\|_2$	ℓ_2-norm and Spectral norm
r	Dimension of low-dimensional vectors
γ	Damping factor of PageRank model

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 [7]. If G' can be obtained by adding or removing a data sample from G, it is termed unbounded DP [6]. The parameter ϵ is referred to as the privacy budget, which is utilized to tune the trade-off between privacy and utility in the algorithm. A smaller value of ϵ indicates a higher level of privacy protection. The parameter δ is informally considered as a failure probability and is typically selected to be very small.

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

Definition 2 (Sensitivity [7]). *Given a function $f : G \to \mathbb{R}^r$, for any neighboring datasets G and G', the ℓ_2-sensitivity of f is defined as $\mathcal{S}_f = \max_{G,G'} \|f(G) - f(G')\|_2$.*

Gaussian mechanism. By utilizing the ℓ_2-sensitivity definition, we can formalize the Gaussian mechanism applied to f as follows:

Theorem 2.1 (Gaussian Mechanism [7]). *For any function $f : G \to \mathbb{R}^r$, the Gaussian mechanism is defined as $\mathcal{A}(G) = f(G) + \mathcal{N}\left(\mathcal{S}_f^2\sigma^2\mathbf{I}\right)$, where $\mathcal{N}\left(\mathcal{S}_f^2\sigma^2\mathbf{I}\right)$ represents a zero-mean Gaussian distribution with $\sigma = \frac{\sqrt{2\log(1.25/\delta)}}{\epsilon}$.*

The concept of sensitivity implies that protecting privacy at the level of individual nodes is more challenging compared to the edge level. This is primarily because modifying a node typically has a considerably greater impact (higher sensitivity) than changing an edge. As a result, a significant amount of noise must be added to ensure privacy for individual nodes.

Important properties. Furthermore, DP possesses two important properties that play a significant role in the implementation of PrivDPR.

Theorem 2.2 (Sequential Composition [8]). *If $\mathcal{A}_1$ ensures (ϵ_1, δ_1)-DP, and $\mathcal{A}_2$ ensures (ϵ_2, δ_2)-DP, then the composition $(\mathcal{A}_1 \circ \mathcal{A}_2)$ guarantees $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

Theorem 2.3 (Post-Processing [8]). *If $\mathcal{A}$ is an algorithm that achieves (ϵ, δ)-DP, then the sequential composition $\mathcal{B}(\mathcal{A}(\cdot))$ with any other algorithm $\mathcal{B}$ that does not have direct or indirect access to the private database also satisfies (ϵ, δ)-DP.*

2.2 PageRank

The Internet and social networks can be seen as vast graph structures. PageRank [28] is a well-known algorithm used for analyzing the links in a graph, making it a representative method for graph link analysis. It operates as an unsupervised learning approach on graph data. The core concept of PageRank involves establishing a random walk model on a directed graph, which can be viewed as a first-order Markov chain. This model describes the behavior of a walker randomly visiting each node along the directed edges of the graph. By meeting certain conditions, the probability of visiting each node during an infinitely long random walk converges to a stationary distribution. At this point, the stationary probability assigned to each node represents its PageRank value, indicating its significance. PageRank is defined recursively, and its calculation is typically performed using an iterative algorithm. The formal definition of PageRank is as follows:

Lemma 2.4. *Consider a graph G. The PageRank score of a node j, denoted as PR_j, represents the probability of reaching node j through random walks. The value of PR_j can be calculated by summing up the ranking scores of its direct predecessors i, weighted by the reciprocal of their out-degree d_i^{out}. Mathematically, we have:*

$$PR_j = \gamma \left(\sum_{i \in P_j} \frac{PR_i}{d_i^{out}} \right) + \frac{1-\gamma}{N}, \quad j = 1, 2, \cdots, N \tag{1}$$

where γ is a damping factor commonly set to 0.85, P_j represents the set of direct predecessors of node j, and the second term ensures that $PR_j > 0$ for $j = 1, 2, \cdots, N$, with $\sum_{j=1}^{N} PR_j = 1$.

3 Problem Definition and Existing Solutions

3.1 Problem Definition

In this paper, we consider a directed and unweighted graph $G = (V, E)$, where V represents the set of nodes and E represents the set of edges. An undirected graph is a special case of the graph that we have defined and is included in our problem definition. Our primary objective is to address the following problem: *given a graph G, how can we generate a synthetic graph $\widetilde{G}$ that possesses similar graph properties as the original graph G, while ensuring node-level DP.*

Definition 3 (Graph Synthesis under Bounded DP[1]). *A graph synthesis model $\mathscr{L}$ satisfies (ϵ, δ)-node-level DP if two neighboring graphs G and G', which differ in only a node and its corresponding edges, satisfy the following condition for all possible $\widetilde{G}_s \subseteq Range(\mathscr{L})$:*

$$\mathbb{P}(\mathscr{L}(G) \in \widetilde{G}_s) \leq \exp(\epsilon) \cdot \mathbb{P}\left(\mathscr{L}\left(G'\right) \in \widetilde{G}_s\right) + \delta,$$

where $\widetilde{G}_s$ denotes the set comprising all possible $\widetilde{G}$.

[1]In this paper, we achieve node-level privacy protection for graph synthesis under bounded DP. This implies that the synthesized graph can have the same number of nodes as the original graph.

The generated synthetic graph can be utilized for various downstream graph analysis tasks without compromising privacy, thanks to the post-processing property of DP.

Theorem 3.1. *Let $\mathscr{L}$ be an (ϵ, δ)-node-level private graph synthesis model, and f is an arbitrary graph query whose input is a simple graph. Then, $f \circ \mathscr{L}$ satisfies (ϵ, δ)-node-level DP.*

In this scenario, even though an attacker has knowledge of the differentially private protocol, encompassing the methods of data encoding and perturbation, they are unable to deduce the original information accurately.

3.2 Existing Solutions

Existing solutions for graph generation include two tracks: differentially private shallow graph models and differentially private deep graph learning models. As our focus is on the latter one, we will defer the introduction of private shallow models to Section 7. In traditional differentially private deep learning [1, 11], the advanced composition mechanisms (i.e., MA) are employed to address excessive splitting of privacy budget during optimization. DP optimizers for *non-graph data* typically update on the summed gradient with Gaussian noise:

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

where $\mathbf{g}(x_i)$ denotes the gradient for each example x_i, $Clip(\cdot)$ is the clipping function defined as $Clip(\mathbf{g}(x_i)) = \mathbf{g}(x_i) / \max\left(1, \frac{\|\mathbf{g}(x_i)\|_2}{C}\right)$, C is the clipping threshold, and B is the batch size. In graph learning, individual examples no longer compute their gradients independently because changing a single node or edge in the graph may affect all gradient values. To address this issue, three types of solutions have been proposed:

- *Private GAN model.* DPGGAN [43] is a differentially private GAN for graph synthesis. It improves the sensitivity by enhancing the MA. DPGGAN proves that the noised clipped gradient $\tilde{\mathbf{g}}$ applied as above guarantees that the learned graph generation model to be edge-level DP, with a different condition from that in Theorem 2.1 due to the nature of graph generation.
- *Private VAE model.* DPGVAE [43] is a differentially private VAE designed for graph synthesis. It achieves the same level of privacy as DPGGAN under the same conditions.
- *Private GNN model.* The features of one node can influence the gradients of other nodes in the network. The sensitivity for $\sum_{i=1}^{B} Clip(\mathbf{g}(x_i))$ may reach BC under node-level DP. Several solutions [4, 27, 32, 41, 51] have been proposed. The classic and advanced approach is GAP [32], which uses aggregation perturbation to achieve RDP and introduces a new GNN architecture tailored for private learning over graphs, resulting in improved trade-offs between privacy and accuracy.

Limitations. Despite the usefulness of these approaches, two limitations in generating synthetic graphs have yet to be solved: 1) DPGGAN and DPGVAE only achieve weak edge-level DP, and 2) MA and RDP typically require a sufficient privacy budget to estimate the privacy guarantee. Thus, these private models mentioned above tend to converge prematurely with a small privacy budget. We explain the second issue in detail using Algorithm 1.

In the deep learning with DP framework, described in the algorithm from Line 7 to Line 9, there is a potential issue of premature termination when working with a limited overall privacy budget, such as $\epsilon \leq 0.5$. This occurs because the privacy loss metric, $\delta_{Accountant}$, converges rapidly towards the desired privacy level δ. As a result, the algorithm may stop prematurely before achieving the desired level of privacy. This premature termination poses a challenge for existing deep learning models that utilize this framework. To mitigate this issue and improve the utility of the trained models, it is often necessary to set a relatively large value for the privacy budget parameter, ϵ. However, this approach introduces a trade-off between privacy and utility since a larger privacy budget carries a higher risk to data privacy.

Algorithm 1: Deep Learning with DP

```
1  for epochs = 1, ··· , T do
2      Take a batch sample set with sampling probability p;
3      Apply clipping to per-sample gradients;
4      Add Gaussian noise to the sum of clipped gradients;
5      Update weights by any optimizer on private gradients
         with learning rate η;
6      Compute δ_Accountant given the target ϵ;
7      if δ_Accountant(ϵ, σ, p, T) ≥ δ then
8          Break;
9      end
10 end
```

4 Our Proposal: PrivDPR

To tackle the limitations outlined in the previous section, we propose a node-level differentially private deep PageRank for graph synthesis, inspired by the simplicity, effectiveness, and ease of analysis of PageRank in Section 2.2. First, we provide an overview of the approach. Next, we describe how we construct the deep PageRank and achieve gradient perturbation. Finally, we present the complete training algorithm.

4.1 Overview

The workflow of PrivDPR is shown in Figure 1, which consists three stages: deep PageRank, gradient perturbation, and graph reconstruction.

- **Deep PageRank.** We design a deep PageRank that serves as the foundation for analyzing the correlation between the number of layers, high sensitivity, and privacy budget splitting. (see Section 4.2)
- **Gradient Perturbation.** We achieve private deep PageRank by gradient perturbation. Instead of directly reducing high sensitivity, we first reveal theoretically that we can preset a desired small sensitivity and achieve it by slightly increasing the number of layers. We then show that the theorem can be extended to resist privacy budget splitting. (see Section 4.3)
- **Graph Reconstruction.** We reconstruct the graph by examining the co-occurrence counts of nodes using the acquired representations during optimization. This process entails creating a transition count matrix, and deriving an edge probability matrix to produce a binary adjacency matrix that represents the synthesized graph. (see **Appendix A**)

4.2 Deep PageRank

We fuse multiple layers into the PageRank, resulting in a modified form of Eq. (1) given by:

$$\min_{\mathbf{\Theta}} \mathscr{L} = \sum_{j \in V} \left(\gamma \left(\sum_{i \in P_j} \frac{f(v_i; \mathbf{\Theta})}{d_i^{out}} \right) + \frac{1-\gamma}{N} - f(v_j; \mathbf{\Theta}) \right)^2, \quad (3)$$

where $f(v; \mathbf{\Theta})$ is a fully connected neural network in the following form:

$$f(v; \mathbf{\Theta}) = \phi_{L+1} \left(\mathbf{W}_{L+1} \phi_L \left(\mathbf{W}_L \left(\phi_{L-1} \left(\mathbf{W}_{L-1} \left(\dots \phi_1 \left(\mathbf{V} \mathbf{W}_1 \right) \dots \right) \right) \right) \right) \right). \quad (4)$$

Figure 1 shows the detailed architecture of deep PageRank (i.e., Eq. (3)), in which graph data is fed into the neural network through a virtual one-hot encoding of nodes as $\mathbf{V}$. The set of learning parameters is denoted as $\Theta = \{\mathbf{V}, \mathbf{W}_1, \cdots, \mathbf{W}_l, \mathbf{W}_{L+1}\}$, with $\mathbf{V}$ in dimensions of $N \times r$, $\mathbf{W}_1$ in dimensions of $r \times d$, $\mathbf{W}_l$ in dimensions of $d \times d$, and $\mathbf{W}_{L+1}$ in dimensions of $d \times 1$. The activation function employed in each layer is denoted as ϕ. For simplicity, the bias terms of each layer are omitted.

However, in the backpropagation of deep PageRank, applying stochastic gradient descent (SGD) to update $\mathscr{L}$ becomes infeasible, since the squared loss term in Eq. (3) involves a summation over all nodes i pointing to node j, denoted as $\sum_{i \in P_j}$. This means that each squared loss term aggregates information from multiple links pointing to the same node j, which contradicts the standard SGD assumption (i.e., $\sum_{(i,j) \in E} L_{ij}$) and is thus non-decomposable. To address this challenge, we alternatively establish an upper bound for $\mathscr{L}$.

Lemma 4.1. *By applying the Cauchy-Schwarz inequality to Eq. (3), an upper bound of the objective function is:*

$$\min_{\mathbf{\Theta}} \mathscr{L} \leq \sum_{(i,j) \in E} d_j^{in} \gamma^2 \left(\frac{f(v_i; \mathbf{\Theta})}{d_i^{out}} - \frac{f(v_j; \mathbf{\Theta})}{d_j^{in} \gamma} \right)^2 + \sum_{(i,j) \in E} \left(\frac{f(v_i; \mathbf{\Theta})}{d_i^{out}} - \frac{f(v_j; \mathbf{\Theta})}{d_j^{in} \gamma} \right) \frac{2\gamma(1-\gamma)}{N} + \sum_{(i,j) \in E} \frac{(1-\gamma)^2}{d_j^{in} N^2}.$$

Proof. Please refer to **Appendix B**. □

According to Lemma 4.1, the objective function for each edge (i, j) is given by:

$$\mathcal{L}(v_i, v_j; \mathbf{\Theta}) = d_j^{in} \gamma^2 \left(\frac{f(v_i; \mathbf{\Theta})}{d_i^{out}} - \frac{f(v_j; \mathbf{\Theta})}{d_j^{in} \gamma} \right)^2 + \left(\frac{f(v_i; \mathbf{\Theta})}{d_i^{out}} - \frac{f(v_j; \mathbf{\Theta})}{d_j^{in} \gamma} \right) \frac{2\gamma(1-\gamma)}{N} + \frac{(1-\gamma)^2}{d_j^{in} N^2}. \quad (5)$$

The proof of this upper bound on the approximation ratio is currently a subject for future research. However, it is important to note that the effectiveness of this upper bound has been demonstrated through experiments, which will be discussed and presented later on.

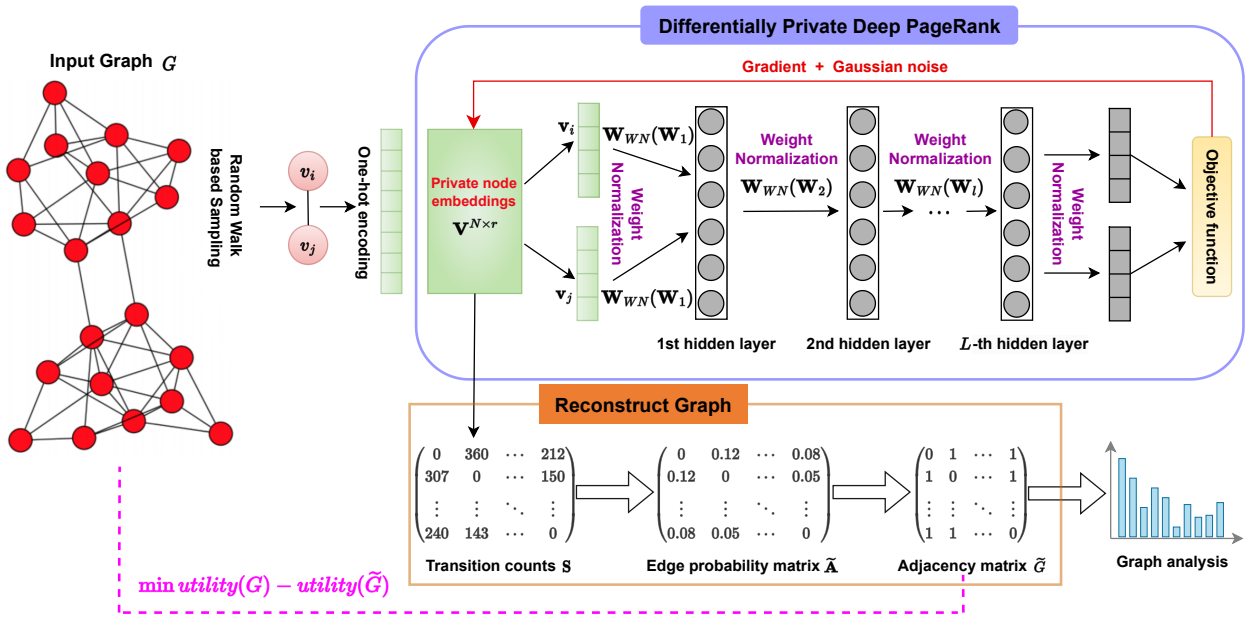

Figure 1: Framework of our proposed PrivDPR

4.3 Gradient Perturbation

4.3.1 ***How to Resist High Sensitivity?*** To yield a private embedding matrix $\mathbf{V}$, a naive method is to first clip $\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}$ and then inject noise into this gradient. This results in the following expression:

$$\widetilde{\nabla}_{\mathbf{V}}\mathcal{L} \leftarrow \frac{1}{B}\left(\sum_{(v_i,v_j)\in E_B} Clip\left(\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}\right) + \mathcal{N}\left(\mathcal{S}_\nabla^2\sigma^2\mathbf{I}\right)\right), \quad (6)$$

where the sensitivity of $\sum_{(v_i,v_j)\in E_B} Clip\left(\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}\right)$, denoted as $\mathcal{S}_\nabla$, can reach up to BC, as modifying one node could potentially impact all gradients in Eq. (6).

As shown in Figure 1, we design an optimizable matrix $\mathbf{V}$ as the input to the neural network $f(v;\Theta)$. Since our goal is to privatize $\mathbf{V}$ for graph synthesis, we only need to add noise to the gradient of $\mathbf{V}$, rather than to the gradients of all the weights. Inspired by this, we explore the use of weight normalization in $f(v;\Theta)$, which enables us to naturally bound the gradient of $\mathbf{V}$ and further reveal the relationships among the model's parameters. Using weight normalization (see Figure 1), not $Clip(\cdot)$, as a bridge to bound $\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}$, we can rewrite Eq. (6) as follows:

$$\widetilde{\nabla}_{\mathbf{V}}\mathcal{L} \leftarrow \frac{1}{B}\left(\sum_{(v_i,v_j)\in E_B} \frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}} + \mathcal{N}\left(\mathcal{S}_\nabla^2\sigma^2\mathbf{I}\right)\right). \quad (7)$$

Theorem 4.2 reveals that we can preset a desired small $\mathcal{S}_\nabla$ for $\sum_{(v_i,v_j)\in E_B} \frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}$ and achieve it by slightly increasing the number of layers.

Theorem 4.2. *Given $\Theta = \{\mathbf{V}, \mathbf{W}_l\}_{l=1}^{L+1}$ with weight normalization using $\mathbf{W}_{WN}(\mathbf{W}) = \frac{\mathbf{W}}{\|s\mathbf{W}\|_2}$ where $s > 1$, we have $\left\|\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}\right\|_2 \leq M\left(\frac{1}{s}\right)^{L+1}$, in which $M = \left(2(N-1)\gamma^2 + 2\gamma + \frac{2\gamma(1-\gamma)}{N}\right)\left(1+\frac{1}{\gamma}\right)$. By presetting a sensitivity $\mathcal{S}_\nabla$, we can determine the maximum number of layers L using $\log_{\frac{1}{s}}\frac{\mathcal{S}_\nabla}{BM} - 1 \leq L$ under node-level DP.*

To prove Theorem 4.2, we will need the following lemmas.

Lemma 4.3. *For activation functions with bounded derivatives, such as Sigmoid, the upper bound of $\frac{\partial f(v;\Theta)}{\partial \mathbf{V}}$ is given by:*

$$\left\|\frac{\partial f(v;\Theta)}{\partial \mathbf{V}}\right\|_2 \leq \prod_{l=1}^{L+1}\|\mathbf{W}_l\|_2, \quad (8)$$

where $\frac{\partial f(v;\Theta)}{\partial \mathbf{V}} = \frac{\partial f(v;\Theta)}{\partial \mathbf{v}}$ holds because the input layer uses a one-hot-encoded vector and $\mathbf{v}$ denotes a vector from $\mathbf{V}$.

Proof. Please refer to **Appendix C**. □

Next, we give the upper bound of $\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}$.

Lemma 4.4. *The upper bound of $\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}$ is*

$$\left\|\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}\right\|_2 \leq M\prod_{l=1}^{L+1}\|\mathbf{W}_l\|_2, \quad (9)$$

where $M = \left(2(N-1)\gamma^2 + 2\gamma + \frac{2\gamma(1-\gamma)}{N}\right)\left(1+\frac{1}{\gamma}\right)$.

Proof. Please refer to **Appendix D**. □

We now prove the Theorem 4.2.

Proof of Theorem 4.2. In Lemma 4.4, by normalizing $\mathbf{W}$ using $\mathbf{W}_{WN}(\mathbf{W}) = \mathbf{W}/\|s\mathbf{W}\|_2$ where $s > 1$, we have

$$\left\|\frac{\partial \mathcal{L}(v_i,v_j;\Theta)}{\partial \mathbf{V}}\right\|_2 \leq M\prod_{l=1}^{L+1}\|\mathbf{W}_{WN}(\mathbf{W}_l)\|_2 \leq M\left(\frac{1}{s}\right)^{L+1}. \quad (10)$$

Consider the worst case, where every gradient is affected in Eq. (7) under node-level DP. Then, according to Eq. (10), we have

$$BM\left(\frac{1}{s}\right)^{L+1} \leq \mathcal{S}_\nabla, \tag{11}$$

where $\mathcal{S}_\nabla$ is a desired sensitivity that can be preset. Therefore, we have

$$\log_{\frac{1}{s}} \frac{\mathcal{S}_\nabla}{BM} - 1 \leq L, \tag{12}$$

which shows that slightly increasing the number of layers can effectively overcome the high sensitivity. □

The following is a real example showing the calculation to achieve a preset sensitivity.

Example 4.5. Citeseer [34] is a popular citation network with 3,327 nodes and 4,732 edges, which is used as the input data. Recall $f(v;\mathbf{\Theta})$ with $\mathbf{V} \in \mathbb{R}^{N\times r}$, $\mathbf{W}^l \in \mathbb{R}^{r\times d}$, and $\mathbf{W}^{L+1} \in \mathbb{R}^{d\times 1}$. Using $N = 3,327, \gamma = 0.85$, we have $M \approx 10,464$. Then, with $r = d = 128, \mathcal{S}_\nabla = 5, B = 128, s = 5$, we determine that $7 \leq L$ using Eq. (12).

4.3.2 ***How to Resist Privacy Budget Splitting?*** Given the total privacy parameters ϵ and σ, and the total number of iterations T for model optimization, the sequential composition property of DP in Theorem 2.2 requires dividing both ϵ and δ. Instead of using advanced composition mechanisms, we evenly divide privacy parameters ϵ and σ. In particular, we calculate σ using $\sigma = \frac{\sqrt{2\log(1.25/(\delta/T))}}{\epsilon/T}$. Note that δ is incorporated within a logarithm function, making its impact negligible. Therefore, we only focus on how to resist privacy budget splitting. In fact, Theorem 4.2 provides a perspective to tackle this issue by reducing $\mathcal{S}_\nabla$ in Eq. (11), which is defined as $BM\left(\frac{1}{s}\right)^{L+1} \leq \frac{\mathcal{S}_\nabla}{T}$. This leads to the following inequality:

$$\log_{\frac{1}{s}} \frac{\mathcal{S}_\nabla}{BMT} - 1 \leq L. \tag{13}$$

Note that $\log(\cdot)$ has a slow growth rate, which means that the additional cost of resistance against privacy budget splitting is not significant.

4.4 Model Optimization

The pseudo-code of PrivDPR are presented in Algorithm 2. We first generate a batch set E_B by random walk. Next, we input these samples into the node embedding matrix $\mathbf{V}$ using a one-hot encoded vector with a length of N. The resulting low-dimensional vectors are then fed into a neural network. To constrain the gradient with respect to $\mathbf{V}$, we normalize each weight $\mathbf{W}$ using weight normalization, and then update $\mathbf{W}$. Subsequently, we introduce Gaussian noise to the sum of the gradients for $\mathbf{V}$, and then update $\mathbf{V}$. After each parameter update, we count transitions in score matrix $\mathbf{S}$. After finishing n_{epochs} training, we transform $\mathbf{S}$ into edge probability matrix $\widetilde{\mathbf{A}}$.

Algorithm 2: PrivDPR Algorithm

Input: graph G, privacy parameters ϵ and δ, delay factor γ, preset sensitivity $\mathcal{S}_\nabla$, number of training epochs n_{epochs}, batch size $\mathfrak{b}$, random walk number R_{wn}, random walk length R_{wl}, dimensions r and d, learning rate η.

Output: Synthetic graph $\widetilde{G}$.

Initialize the learning parameters set;
for $i = 1$ *to* n_{epochs} **do**
  for $j = 1$ *to* $\lfloor N/\mathfrak{b} \rfloor$ **do**
    `// Generate batch samples`
    Create a list of indices, *node_list*, starting from $j \cdot \mathfrak{b}$ and ending at $(j+1)\cdot\mathfrak{b} - 1$;
    for *each node_id in node_list* **do**
      Generate a batch of node pairs E_B through random walks with walk number R_{wn} and walk length R_{wl};
    end
    `// See Section 4.3.1 for details`
    Apply weight normalization to each weight $\mathbf{W}$;
    Update $\mathbf{W}$ by Adam optimizer with learning rate η;
    `// See Sections 4.3.1 and 4.3.2 for details`
    Add Gaussian noise to the sum of the gradients for $\mathbf{V}$;
    Update $\mathbf{V}$ by Adam optimizer with learning rate η;
    `// Generate score matrix`
    Sample graphs from $\mathbf{V}\mathbf{V}^\top$ to generate score matrix $\mathbf{S}$;
  end
end
Convert score matrix $\mathbf{S}$ to edge-independent model $\widetilde{\mathbf{A}}$: $\mathbf{S}^\dagger \leftarrow \max\{\mathbf{S}, \mathbf{S}^\top\}, \widetilde{\mathbf{A}} \leftarrow \mathbf{S}^\dagger / sum(\mathbf{S}^\dagger), \widetilde{G} \leftarrow \widetilde{\mathbf{A}}$;
return $\widetilde{G}$;

5 Privacy and Complexity Analysis

5.1 Privacy Analysis

In this section, we provide a privacy analysis for PrivDPR.

Theorem 5.1. *The synthetic graphs generated by PrivDPR satisfies (ϵ, δ)-node-level DP.*

Proof. In Algorithm 2, for each weight parameter that needs optimization, the total number of iterations $T = n_{epochs}\lfloor N/\mathfrak{b} \rfloor$ is fixed a priori, and the desired privacy cost, say ϵ, is split across the iterations: $\epsilon = \epsilon_1 + \cdots + \epsilon_T$. In this work, the privacy budget is evenly split across iterations, so $\epsilon_1 = \cdots = \epsilon_T = \frac{\epsilon}{T}$. Since Gaussian noise is injected into the embedding matrix $\mathbf{V}$, the $\mathbf{V}$ satisfies $(\frac{\epsilon}{T}, \frac{\delta}{T})$-node-level DP for each iteration. In particular, node privacy is satisfied because the sensitivity is calculated according to the definition of node DP in Section 4.3.1. After $n_{epochs}\lfloor N/\mathfrak{b} \rfloor$ iterations, $\mathbf{V}$ naturally satisfies (ϵ, δ)-node-level DP, following the sequential composition property. Also, the resulting graphs obey (ϵ, δ)-node-level DP, as stipulated by the post-processing property of DP. □

5.2 Complexity Analysis

Here we analyze the computational complexity of PrivDPR. The time complexity for initialization is $O(1)$. The time complexity for the outer loop is n_{epochs}. The time complexity for the inner loop is

$\lfloor N/\mathfrak{b} \rfloor$. Random walk generation has a complexity of $O(\mathfrak{b}R_{wn}R_{wl})$ per batch. The complexity of weight normalization and private updates is $O(Ld^2)$. The score matrix has a complexity of $O(N^2r)$. Considering the refined considerations above, the overall time complexity can be expressed as $O(n_{epochs}(\lfloor N/\mathfrak{b} \rfloor(\mathfrak{b}R_{wn}R_{wl} + Ld^2) + N^2r))$. This implies that the time complexity is linear with respect to the number of nodes in the graph, so our method is scalable and can be applied to large-scale graphs.

6 Experiments

In this section, we will answer the following three questions:

- How do the weight normalization parameter s and weight dimension d in neural networks affect the performance of PrivDPR? (see Section 6.1)
- How does the privacy budget ϵ impact on the performance of PrivDPR? (see Section 6.2)
- How scalable is PrivDPR in the context of link prediction and node classification tasks? (see Section 6.3)

Datasets. We run experiments on the five real-world datasets, Cora[2], Citeseer [34], p2p[3], Chicago[4], and Amazon[5]. Cora is a citation network of academic papers with 2,708 nodes, 7 classes, and 5,429 edges. Citeseer is a similar citation network with 3,327 nodes, 6 classes, and 4,732 edges. p2p is a sequence of snapshots from the Gnutella P2P network, consisting of 6,301 nodes and 20,777 edges. Chicago is a directed transportation network of the Chicago area with 12,982 nodes and 39,018 edges. Amazon is a co-purchase network with 410,236 nodes and 3,356,824 edges, representing product connections based on co-purchases. Since we focus on simple graphs in this work, all datasets are pre-processed to remove self-loops.

Baselines. We compare our PrivDPR [6] with four other baselines: GAP [32], DPGGAN [43], DPGVAE [43], and DPR (No DP). GAP represents the current state-of-the-art differentially private GNN model, designed to produce private node embeddings. For a fair comparison, we configure GAP to generate synthetic graphs using the same generation method as PrivDPR. For a fair comparison, *we configure GAP to generate synthetic graphs using the same generation method as PrivDPR.* In this study, we simulate a scenario in which the graphs contain only structural information, whereas GAP depends on node features. To guarantee a fair evaluation, similar to prior research [5], we employ randomly generated features as inputs for GAP. Given that random features do not infringe on privacy, we eliminate the noise perturbation on features in GAP. DPR (No DP) serves as the non-private version of PrivDPR.

Parameter Settings. In PrivDPR, we vary the privacy budget ϵ from $\{0.1, 0.2, 0.4, 0.8, 1.6, 3.2\}$ while keeping the privacy parameter δ fixed at 10^{-5}. The learning rate is set to $\eta = 1 \times 10^{-3}$, which is consistent with the settings used in DPGGAN and DPGVAE. We limit the maximum number of training epochs to $n_{epochs} = 5$. The embedding dimension is chosen as $r = 128$. Note that we do not specifically show the effect of r as its setting is commonly used in various network embedding methods [5, 21, 29, 38, 39, 50]. The delay factor is set to $\gamma = 0.85$. The batch size is $\mathfrak{b} = 16$. Also, we use $R_{wn} = 2$ random walks with a length of $R_{wl} = 16$. These values are one-fifth of the recommended values in DeepWalk [29]. In Section 6.1, we investigate the impact of adjusting the weight normalization parameter s on PrivDPR, while considering the value of $\mathcal{S}_\nabla$ as 5, which is determined based on DPGGAN. Note that we do not modify the number of layers L in PrivDPR, since it is calculated dynamically based on s using Eq. (13). Also, in Section 6.1, we assess the influence of varying the dimension of the hidden layer weight d on PrivDPR. To ensure consistency with the original papers, we employ the official GitHub implementations for GAP, DPGGAN, and DPGVAE. We replicate the experimental setup described in those papers to maintain consistency and comparability.

Graph Utility Metrics. To evaluate the similarity between G and $\widetilde{G}$, we use eight graph topological metrics: triangle count (TC), wedge count (WC), claw count (CC), relative edge distribution entropy (REDE), characteristic path length (CPL), Diameter, size of the largest connected component (LCC), and degree distribution.

We evaluate the accuracy of PrivDPR in the aforementioned graph metrics patterns over all datasets against the baselines. The accuracy of each method $\mathcal{A}$ on graph G is measured by the mean relative error (MRE) [36], namely $\text{MRE} = \frac{1}{|\mathcal{A}|}\sum_{\mathcal{A}_i \in \mathcal{A}} \left| \frac{\hat{\mathcal{A}}_i(G) - \mathcal{A}_i(G)}{\mathcal{A}_i(G)} \right|$, where $\mathcal{A}_i(G)$ denotes the true query result in input graph G, and $\hat{\mathcal{A}}_i(G)$ denotes the differentially private query result in G. Each result reported is averaged over five repeated runs, that is $|\mathcal{A}| = 5$. A lower MRE indicates a lower error and thus a higher data utility.

The degree distribution is measured with Kolmogorov-Smirnov (KS) [17, 19], which quantifies the maximum distance between the two-degree distributions. Let F and F' denote the cumulative distribution functions estimated from the sorted degree sequences of the original and synthetic graphs, respectively. Then $KS_D = \max_d |F(d) - F'(d)|$. The smaller this statistic value, the closer (more similar) the degree distributions between the synthetic and original graphs. For KS, we also report the average performance over five independent runs.

6.1 Impact of Parameters

Parameter s. In this experiment, we investigate the impact of the parameter s on the performance of PrivDPR, focusing on graph metrics such as TC, REDE, CPL, and KS. For the datasets Cora, Citeseer, p2p, and Chicago, we consider different values of s, namely, 2, 4, 6, 8. As shown in Table 2, we analyze the standard deviation (SD) of the MRE and KS_D values for PrivDPR across various s. Remarkably, we consistently observe that the SD remains consistently not more than $3.5122E-02$ across all datasets. This result highlights the robustness of PrivDPR against variations in s. Based on these findings, we have made the decision to set s as a constant value of 8 in subsequent experiments.

Parameter d. In this experiment, we investigate the impact of the weight dimension parameter d on the performance of PrivDPR in terms of various graph metrics, including TC, REDE, CPL, and KS_D. Specifically, we consider different values of d, namely, 64, 128, 256, 512 for datasets Cora, Citeseer, p2p, and Chicago. As illustrated in Table 3, we observe that the SD of the MRE and KS_D values for PrivDPR with different d is consistently not more than $5.8030E-02$ across all datasets. This finding indicates that PrivDPR exhibits

[2] https://linqs.org/datasets/
[3] http://snap.stanford.edu/data/p2p-Gnutella08.html
[4] http://konect.cc/networks/tntp-ChicagoRegional/
[5] https://snap.stanford.edu/data/amazon0505.html
[6] Our code is available at https://github.com/sunnerzs/PrivDPR.

robustness to variations in d. Consequently, in subsequent experiments, we fix d at 64 as a constant value.

Table 2: Summary of MRE and KS_D with different s, given $\epsilon = 3.2$ and $\mathcal{S}_\nabla = 5$. SD represents the standard deviation of the values in each row.

Dataset	Statistics	Parameter s				SD
		2	4	6	8	
Cora	TC	0.9926	0.9914	0.9908	0.9893	**1.4029E-03**
	REDE	0.0244	0.0245	0.0242	0.0245	**1.6113E-04**
	CPL	0.1135	0.1106	0.1155	0.1162	**2.4788E-03**
	KS	0.5231	0.5223	0.5224	0.5356	**6.4809E-03**
Citeseer	TC	0.9923	0.9944	0.9919	0.9936	**1.1799E-03**
	REDE	0.0168	0.0182	0.0172	0.0165	**7.5261E-04**
	CPL	0.3105	0.3068	0.3116	0.3198	**5.4899E-03**
	KS	0.5792	0.5249	0.5830	0.6084	**3.5122E-02**
p2p	TC	0.9591	0.9593	0.9566	0.9557	**1.7917E-03**
	REDE	0.0381	0.0379	0.0382	0.0382	**1.5189E-04**
	CPL	0.0247	0.0232	0.0227	0.0227	**9.5005E-04**
	KS	0.1322	0.1325	0.1224	0.1177	**7.3605E-03**
Chicago	TC	0.9913	0.9901	0.9950	0.9938	**2.2624E-03**
	REDE	0.0078	0.0076	0.0078	0.0080	**1.7852E-04**
	CPL	0.8118	0.8114	0.8124	0.8121	**4.3182E-04**
	KS	0.2854	0.2854	0.2854	0.2778	**3.8314E-03**

Table 3: Summary of MRE and KS_D with different d, given $\epsilon = 3.2$ and $s = 8$. SD represents the standard deviation of the values in each row.

Dataset	Statistics	Weight Dimension d				SD
		64	128	256	512	
Cora	TC	0.9893	0.9920	0.9905	0.9914	**1.2012E-03**
	REDE	0.0245	0.0252	0.0248	0.0234	**7.8200E-04**
	CPL	0.1162	0.1068	0.1101	0.1174	**5.0061E-03**
	KS	0.5356	0.5216	0.5227	0.5388	**8.7785E-03**
Citeseer	TC	0.9936	0.9923	0.9944	0.9936	**8.8327E-04**
	REDE	0.0165	0.0172	0.0179	0.0170	**5.7647E-04**
	CPL	0.3198	0.3128	0.3041	0.3159	**6.6596E-03**
	KS	0.6084	0.5507	0.4888	0.6120	**5.8030E-02**
p2p	TC	0.9557	0.9532	0.9547	0.9616	**3.6798E-03**
	REDE	0.0382	0.0381	0.0383	0.0383	**1.0559E-04**
	CPL	0.0227	0.0236	0.0205	0.0226	**1.3138E-03**
	KS	0.1177	0.1572	0.1276	0.1271	**1.7174E-02**
Chicago	TC	0.9938	0.9926	0.9857	0.9932	**3.7517E-03**
	REDE	0.0080	0.0077	0.0077	0.0077	**1.5630E-04**
	CPL	0.8121	0.8114	0.8115	0.8116	**3.2051E-04**
	KS	0.2778	0.2689	0.3032	0.2854	**1.4568E-02**

6.2 Impact of Privacy Budget on Graph Statistics Preservation

The privacy budget ϵ is a critical parameter in the context of DP, as it determines the level of privacy provided by the algorithm. We conduct experiments to evaluate the impact of ϵ on the performance of each private algorithm. Specifically, we present the results on Cora in Figure 2. The results on Citeseer, p2p and Chicago datasets can be found in **Appendix E**. From these figures, PrivDPR consistently outperforms both DPGGAN and DPGVAE with weak edge-level DP guarantees. The reasons why the results of DPGGAN and DPGVAE are poor are twofold: 1) they use the MA mechanism and tend to converge prematurely under small ϵ; and 2) they use a threshold-based method for reconstructing synthetic graphs, which potentially generates numerous disconnected subgraphs within the synthesized graph. Moreover, it is worth noting that in most cases, PrivDPR achieves comparable results to DPR (No DP) and surpasses GAP, even with a small privacy budget of $\epsilon = 0.1$. This phenomenon can be attributed to two factors. First, our designed deep PageRank effectively captures the structural properties of the input graph. Second, the theorem presented in Section 4.3 addresses challenges such as high sensitivity and excessive splitting on the privacy budget, further enhancing the performance of PrivDPR.

6.3 Link Prediction and Node Classification

For the link prediction task, the existing links in each dataset are randomly divided into a training set (80%) and a test set (20%). 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. We measure performance using the area under the ROC curve (AUC). The AUC results and analysis for all methods are presented in Tables 4 and 5, with $\epsilon = 0.1$ and $\epsilon = 3.2$, in **Appendix F**. In summary, PrivDPR consistently achieves the highest AUC among all privacy-preserving algorithms and maintains high stability.

For the node classification task, we randomly sample 90% of the nodes as training data and randomly sample 10% of the nodes outside the training set as test data. We follow the procedure of [10] and evaluate our embeddings using Micro-F1 score. We report the results in Table 6 in **Appendix G**, with $\epsilon = 0.1$ and $\epsilon = 3.2$. To summarize, PrivDPR outperforms other privacy methods across various datasets in terms of Micro-F1 and SD. This indicates PrivDPR is an outstanding off-the-shelf method for different graph downstream tasks.

7 Related Work

Unlike traditional privacy-preserving methods [16, 24, 37], differential privacy (DP) and its variant, local differential privacy (LDP), offer strong privacy guarantees and robustness against adversaries with prior knowledge and have been widely applied across various fields [44–47, 52]. The related work of this paper covers differentially private shallow and deep graph generation models.

7.1 Private Shallow Graph Generation Models

Several research efforts have been dedicated to achieving differentially private publication for social graph data. One approach is to generate representative synthetic graphs using the Kronecker graph model, as explored by Mir and Wright [25]. They estimate the model parameters from the input graph under DP. Another approach is the Pygmalion model proposed by Sala *et al.*, which utilizes the dK-series of the input graph to capture the distribution of observed degree pairs on edges [33]. This model has been combined with smooth sensitivity to construct synthetic graphs [40].

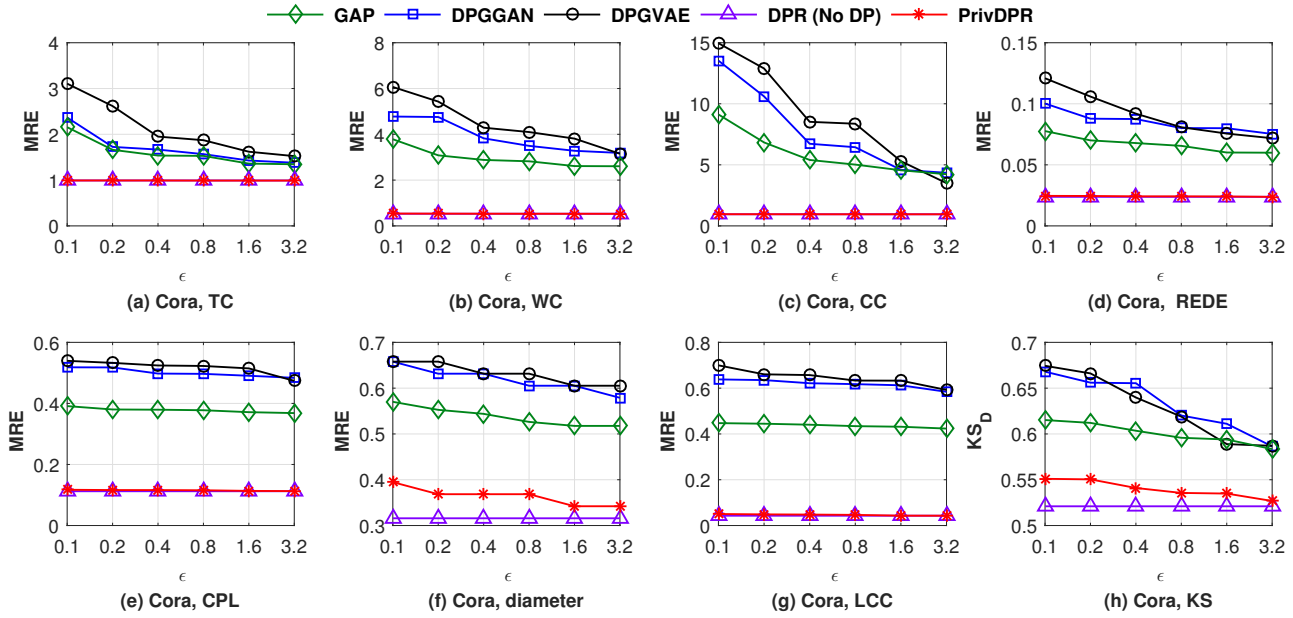

Figure 2: Privacy budget on Cora

Xiao *et al.* encode the graph structure through private edge counting queries under the hierarchical random graph model and report improved results compared to the dK-series approach [42]. Chen *et al.* employ the exponential mechanism to sample an adjacency matrix after clustering the input graph [3]. Proserpio *et al.* suggest down-weighting the edges of a graph non-uniformly to mitigate high global sensitivity arising from very high degree nodes. They demonstrate this approach, combined with MCMC-based sampling, for generating private synthetic graphs [30]. Gao and Li propose a private scheme to preserve both the adjacency matrix and persistent homology, specifically targeting the persistence structures in the form of holes [12]. These methods, along with subsequent research on private graph release [9, 13, 49], typically guarantee weak edge-level DP.

7.2 Private Deep Graph Generation Models

With the rapid development of deep learning, numerous advanced deep graph generation models have emerged in recent times. These models employ various powerful neural networks in a learn-to-generate manner [2, 20, 48]. For instance, NetGAN [2] converts graphs into biased random walks, learns to generate walks using GAN, and then constructs graphs from the generated walks. GraphRNN [48], on the other hand, treats graph generation as a sequence of node and edge additions, and models it using a heuristic breadth-first search scheme and hierarchical RNN. These deep learning models can generate graphs with richer properties and flexible structures learned from real-world networks. However, existing research on deep graph generation has not thoroughly examined the potential privacy threats associated with training and generating graphs using powerful models. A recent related solution proposed by Yang *et al.* introduces DPGGAN and DPGVAE models for graph synthesis [43]. They improve MA [1], which is an effective strategy for computing privacy loss after multiple queries, but only achieves weak edge-level DP. Recent research endeavors [4, 32, 41, 51] have been dedicated to the advancement of node-level differentially private GNNs. These efforts aim to address the issue of high sensitivity, but achieving an optimal balance between privacy and utility remains a significant challenge. This is primarily because they often employ advanced composition mechanisms to manage privacy budget splitting and yet tend to converge prematurely when working with a small privacy budget.

8 Conclusion

This paper focuses on synthetic graph generation under DP. The underlying highlights lie in the following two aspects. First, we design a novel privacy-preserving deep PageRank for graph synthesis, called PrivDPR, which achieves DP by adding noise to the gradient for a specific weight during learning. Second, we theoretically show that increasing the number of layers can effectively overcome the challenges associated with high sensitivity and privacy budget splitting. Through privacy analysis, we prove that the generated synthetic graph satisfies (ϵ, δ)-node-level DP. Extensive experiments on real-world graph datasets show that our solution substantially outperforms state-of-the-art competitors. Our future focus is on developing more graph generation techniques for node embeddings to enhance the utility of synthetic graphs.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No: 62372122, 92270123 and 62072390), and the Research Grants Council, Hong Kong SAR, China (Grant No: 15224124, 25207224, C2004-21GF and C2003-23Y). We also thank Dr. Nan Fu and Dr. Lihe Hou for helpful discussions.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. NetGAN: Generating graphs via random walks. In *International Conference on Machine Learning*. 610–619.

[3] Rui Chen, Benjamin C Fung, Philip S Yu, and Bipin C Desai. 2014. Correlated network data publication via differential privacy. *The VLDB Journal* 23, 4 (2014), 653–676.

[4] 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).

[5] Lun Du, Xu Chen, Fei Gao, Qiang Fu, Kunqing Xie, Shi Han, and Dongmei Zhang. 2022. Understanding and improvement of adversarial training for network embedding from an optimization perspective. In *ACM International Conference on Web Search and Data Mining*. 230–240.

[6] Cynthia Dwork. 2006. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*. 1–12.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*. 265–284.

[8] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*. 9, 3–4 (2014), 211–407.

[9] Marek Eliáš, Michael Kapralov, Janardhan Kulkarni, and Yin Tat Lee. 2020. Differentially private release of synthetic graphs. In *Annual ACM-SIAM Symposium on Discrete Algorithms*. 560–578.

[10] Alessandro Epasto, Vahab Mirrokni, Bryan Perozzi, Anton Tsitsulin, and Peilin Zhong. 2022. Differentially private graph learning via sensitivity-bounded personalized pagerank. In *International Conference Neural Information Processing Systems*. 22617–22627.

[11] Jie Fu, Qingqing Ye, Haibo Hu, Zhili Chen, Lulu Wang, Kuncan Wang, and Xun Ran. 2024. DPSUR: Accelerating differentially private stochastic gradient descent using selective update and release. In *Proceedings of the VLDB Endowment*. 1200–1213.

[12] Tianchong Gao and Feng Li. 2019. Preserving persistent homology in differentially private graph publications. In *IEEE INFOCOM Conference on Computer Communications*. 2242–2250.

[13] Anupam Gupta, Aaron Roth, and Jonathan Ullman. 2012. Iterative constructions and private data release. In *Theory of Cryptography Conference*. 339–356.

[14] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *IEEE International Conference on Data Mining*. 169–178.

[15] Michael Hay, Kun Liu, Gerome Miklau, Jian Pei, and Evimaria Terzi. 2011. Privacy-aware data management in information networks. In *ACM SIGMOD Conference on Management of Data*. 1201–1204.

[16] Haibo Hu, Jianliang Xu, Xizhong Xu, Kexin Pei, Byron Choi, and Shuigeng Zhou. 2014. Private search on key-value stores with hierarchical indexes. In *IEEE International Conference on Data Engineering*. 628–639.

[17] Xun Jian, Yue Wang, and Lei Chen. 2021. Publishing graphs under node differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 4164–4177.

[18] Fan Jiang, Carson K Leung, and Adam GM Pazdor. 2016. Big data mining of social networks for friend recommendation. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 921–922.

[19] Zach Jorgensen, Ting Yu, and Graham Cormode. 2016. Publishing attributed social graphs with formal privacy guarantees. In *ACM SIGMOD Conference on Management of Data*. 107–122.

[20] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[21] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin. 2017. PRUNE: Preserving proximity and global ranking for network embedding. In *International Conference Neural Information Processing Systems*. 5257–5266.

[22] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Signed networks in social media. In *International Conference on Human Factors in Computing Systems*. 1361–1370.

[23] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph Evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery From Data* 1, 1 (2007), 2–es.

[24] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. *l*-diversity: Privacy beyond *k*-anonymity. *ACM Transactions on Knowledge Discovery from Data* 1, 1 (2007), 3–es.

[25] Darakhshan J. Mir and Rebecca N. Wright. 2009. A differentially private graph estimator. In *IEEE International Conference on Data Mining Workshops*. 122–129.

[26] Peter J Mucha and Mason A Porter. 2010. Communities in multislice voting networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20, 4 (2010).

[27] Iyiola E Olatunji, Thorben Funke, and Megha Khosla. 2023. Releasing graph neural networks with differential privacy guarantees. *Transactions on Machine Learning Research* (2023), 2835–8856.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. *The PageRank citation ranking: Bring order to the web.* Technical Report. Stanford InfoLab.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 701–710.

[30] Davide Proserpio, Sharon Goldberg, and Frank McSherry. 2014. Calibrating data to sensitivity in private data analysis. In *Proceedings of the VLDB Endowment*. 637–648.

[31] Luca Rendsburg, Holger Heidrich, and Ulrike Von Luxburg. 2020. NetGAN without GAN: From random walks to low-rank approximations. In *International Conference on Machine Learning*. 8073–8082.

[32] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2023. GAP: Differentially private graph neural networks with aggregation perturbation. In *USENIX Security Symposium*. 3223–3240.

[33] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. 2011. Sharing graphs using differentially private graph models. In *ACM SIGCOMM Internet Measurement Conference*. 81–98.

[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* 29, 3 (2008), 93–106.

[35] Dushyant Sharma, Rishabh Shukla, Anil Kumar Giri, and Sumit Kumar. 2019. A brief review on search engine optimization. In *International Conference on Cloud Computing, Data Science and Engineering*. 687–692.

[36] 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 *ACM SIGSAC Conference on Computer and Communications Security*. 703–717.

[37] Latanya Sweeney. 2002. *k*-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[38] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *International Conference on World Wide Web*. 1067–1077.

[39] Cunchao Tu, Xiangkai Zeng, Hao Wang, Zhengyan Zhang, Zhiyuan Liu, Maosong Sun, Bo Zhang, and Leyu Lin. 2018. A unified framework for community detection and network representation learning. *IEEE Transactions on Knowledge and Data Engineering* 31, 6 (2018), 1051–1065.

[40] Yue Wang and Xintao Wu. 2013. Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy* 6, 2 (2013), 127.

[41] Zihang Xiang, Tianhao Wang, and Di Wang. 2024. Preserving node-level privacy in graph neural networks. In *IEEE Symposium on Security and Privacy*. 4714–4732.

[42] Qian Xiao, Rui Chen, and Kian-Lee Tan. 2014. Differentially private network data release via structural inference. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 911–920.

[43] Carl Yang, Haonan Wang, Ke Zhang, Liang Chen, and Lichao Sun. 2021. Secure deep graph generation with link differential privacy. In *International Joint Conference on Artificial Intelligence*. 3271–3278.

[44] Qingqing Ye, Haibo Hu, Kai Huang, Man Ho Au, and Qiao Xue. 2023. Stateful Switch: Optimized time series release with local differential privacy. In *IEEE INFOCOM Conference on Computer Communications*. 1–10.

[45] Qingqing Ye, Haibo Hu, Ninghui Li, Xiaofeng Meng, Huadi Zheng, and Haotian Yan. 2021. Beyond Value Perturbation: Local differential privacy in the temporal setting. In *IEEE INFOCOM Conference on Computer Communications*. 1–10.

[46] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. 2019. PrivKV: Key-value data collection with local differential privacy. In *IEEE Symposium on Security and Privacy*. 317–331.

[47] Qingqing Ye, Haibo Hu, Xiaofeng Meng, Huadi Zheng, Kai Huang, Chengfang Fang, and Jie Shi. 2021. PrivKVM*: Revisiting key-value statistics estimation with local differential privacy. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2021), 17–35.

[48] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*. 5708–5717.

[49] Quan Yuan, Zhikun Zhang, Linkang Du, Min Chen, Peng Cheng, and Mingyang Sun. 2023. PrivGraph: Differentially private graph data publication by exploiting community information. In *USENIX Security Symposium*. 3241–3258.

[50] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *International Conference Neural Information Processing Systems*. 5165–5175.

[51] Qiuchen Zhang, Hong kyu Lee, Jing Ma, Jian Lou, Carl Yang, and Li Xiong. 2024. DPAR: Decoupled graph neural networks with node-level differential privacy. In *Proceedings of the ACM on Web Conference*. 1170–1181.

[52] Yuemin Zhang, Qingqing Ye, Rui Chen, Haibo Hu, and Qilong Han. 2023. Trajectory data collection with local differential privacy. In *Proceedings of the VLDB Endowment*. 2591–2604.

Appendix A Graph Synthesis

In what follows, we describe the specific details of generating graphs as discussed in previous works [2, 31]. Once training is complete, we employ the node embeddings to create a score matrix $\mathbf{S}$ that records transition counts. Since we intend to analyze this synthetic graph, we convert the raw counts matrix $\mathbf{S}$ into a binary adjacency matrix. Initially, $\mathbf{S}$ is symmetrized by setting $s_{ij} = s_{ji} = \max(s_{ij}, s_{ji})$. However, since we have no explicit control over the starting node of the random walks generated by G, high-degree nodes are likely to be overrepresented. Consequently, a simple strategy such as thresholding or selecting the top-k entries for binarization may exclude low-degree nodes and create isolated nodes. To address this concern, we ensure that every node i has at least one edge by sampling a neighbor j with a probability of $p_{ij} = \frac{s_{ij}}{\sum_v s_{iv}}$. If an edge has already been sampled, we repeat this procedure. To ensure the graph is undirected, we include (j, i) for every edge (i, j). We continue sampling edges without replacement, using the probability $p_{ij} = \frac{s_{ij}}{\sum_{u,v} s_{uv}}$ for each edge (i, j), until we reach the desired number of edges (e.g., determined by applying the Sigmoid function with the threshold value 0.5 to the score matrix). Additionally, to ensure the graph is undirected, we include (j, i) for every edge (i, j).

Appendix B Proof of Lemma 4.1

Proof. Let P_j denote the set of direct predecessors of node j, and d_i^{out} represent the out-degree of node i. According to Eq. (3), the objective function can be expressed as follows:

$$\begin{aligned}
\min_{\Theta} \mathscr{L} &= \sum_{j\in V} \gamma^2 \left(\sum_{i\in P_j} \frac{f(v_i;\Theta)}{d_i^{out}} + \frac{1-\gamma}{\gamma N} - \frac{f(v_j;\Theta)}{\gamma} \right)^2 \\
&\overset{(1)}{=} \sum_{j\in V} \gamma^2 \left(\sum_{i\in P_j} \frac{f(v_i;\Theta)}{d_i^{out}} - \frac{f(v_j;\Theta)}{\gamma} \right)^2 \\
&\quad + \sum_{j\in V} 2\gamma^2 \left(\sum_{i\in P_j} \frac{f(v_i;\Theta)}{d_i^{out}} - \frac{f(v_j;\Theta)}{\gamma} \right) \frac{1-\gamma}{\gamma N} + \sum_{j\in V} \frac{(1-\gamma)^2}{N^2} \\
&\overset{(2)}{\leq} \sum_{(i,j)\in E} d_j^{in}\gamma^2 \left(\frac{f(v_i;\Theta)}{d_i^{out}} - \frac{f(v_j;\Theta)}{d_j^{in}\gamma} \right)^2 \\
&\quad + \sum_{(i,j)\in E} \left(\frac{f(v_i;\Theta)}{d_i^{out}} - \frac{f(v_j;\Theta)}{d_j^{in}\gamma} \right) \frac{2\gamma(1-\gamma)}{N} + \sum_{(i,j)\in E} \frac{(1-\gamma)^2}{d_j^{in}N^2},
\end{aligned}$$

where the step (2) holds because the Cauchy-Schwarz inequality is applied to the first term in the step (1). □

Appendix C Proof of Lemma 4.3

Proof. The Lipschitz constant of a function and the norm of its gradient are two sides of the same coin. We define $\|f\|_{\text{Lip}}$ as the smallest value ρ such that $\|f(\mathbf{x}) - f(\mathbf{x}')\|/\|\mathbf{x} - \mathbf{x}'\| \leq \rho$ for any $\mathbf{x}, \mathbf{x}'$, with the norm being the ℓ_2-norm. We can use the inequality $\|f_1 \circ f_2\|_{\text{Lip}} \leq \|f_1\|_{\text{Lip}} \cdot \|f_2\|_{\text{Lip}}$ to observe the following bound on $\|f\|_{\text{Lip}}$:

$$\begin{aligned}
\|f\|_{\text{Lip}} &\leq \|\phi_{L+1}\|_{\text{Lip}} \cdot \|\mathbf{W}_{L+1}\mathbf{V}_L\|_{\text{Lip}} \cdot \|\phi_L\|_{\text{Lip}} \cdot \|\mathbf{W}_L\mathbf{V}_{L-1}\|_{\text{Lip}} \\
&\quad \cdots \|\phi_1\|_{\text{Lip}} \cdot \|\mathbf{W}_1\mathbf{V}_0\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|\phi_l\|_{\text{Lip}} \cdot \|\mathbf{W}_l\mathbf{V}_{l-1}\|_{\text{Lip}} \\
&\overset{(i)}{\leq} \prod_{l=1}^{L+1} \|\mathbf{W}_l\mathbf{V}_{l-1}\|_{\text{Lip}} \overset{(ii)}{=} \prod_{l=1}^{L+1} \max_{\mathbf{V}\neq 0} \frac{\|\mathbf{W}_l\mathbf{V}\|_2}{\|\mathbf{V}\|_2} = \prod_{l=1}^{L+1} \|\mathbf{W}_l\|_2,
\end{aligned}$$

where the step (i) holds since the *Sigmoid* function is used, and the step (ii) holds because weight normalization is implemented through spectral normalization. The *Sigmoid* function can be substituted with other activation functions, such as *ReLU* and *leaky ReLU*, while the inequalities above still hold.

According to $\|f\|_{\text{Lip}}$, it is possible to show that $\left\| \frac{\partial f(v;\Theta)}{\partial \mathbf{V}} \right\|_2 \leq \prod_{l=1}^{L+1} \|\mathbf{W}_l\|_2$. This fact allows us to bound the weight $\mathbf{W}_l$, instead of the output of the gradient function, and obtain a bound on the ℓ_2-sensitivity of the gradient. □

Appendix D Proof of Lemma 4.4

Proof. With Lemma 4.3, we have

$$\begin{aligned}
\left\| \frac{\partial \mathcal{L}(v_i, v_j; \Theta)}{\partial \mathbf{V}} \right\|_2 &= \left\| \left(\frac{2d_j^{in}\gamma^2 f(v_i;\Theta)}{d_i^{out}} - 2\gamma f(v_j;\Theta) + \frac{2\gamma(1-\gamma)}{N} \right) \right. \\
&\quad \left. \left(\frac{\partial f(v_i;\Theta)/\partial \mathbf{V}}{d_i^{out}} - \frac{\partial f(v_j;\Theta)/\partial \mathbf{V}}{d_j^{in}\gamma} \right) \right\|_2 \\
&\overset{(1)}{\leq} \left\| \left(\frac{2d_j^{in}\gamma^2}{d_i^{out}} + 2\gamma + \frac{2\gamma(1-\gamma)}{N} \right) \left(\frac{\partial f(v_i;\Theta)/\partial \mathbf{v}_i}{d_i^{out}} - \frac{\partial f(v_j;\Theta)/\partial \mathbf{v}_j}{d_j^{in}\gamma} \right) \right\|_2 \\
&\leq \left(\frac{2d_j^{in}\gamma^2}{d_i^{out}} + 2\gamma + \frac{2\gamma(1-\gamma)}{N} \right) \left(\frac{1}{d_i^{out}} \left\| \prod_{l=1}^{L+1} \mathbf{W}_l \right\|_2 + \frac{1}{d_j^{in}\gamma} \left\| \prod_{l=1}^{L+1} \mathbf{W}_l \right\|_2 \right) \\
&\leq \left(2(N-1)\gamma^2 + 2\gamma + \frac{2\gamma(1-\gamma)}{N} \right) \left(1 + \frac{1}{\gamma} \right) \prod_{l=1}^{L+1} \|\mathbf{W}_l\|_2,
\end{aligned}$$

where the step (1) holds since the $f(v;\Theta)$ with *Sigmoid* activation in Eq. (4) is less than 1, and triangle inequality is used. □

Appendix E Graph Statistics

The findings for link prediction across the Citeseer, p2p and Chicago datasets are displayed in Figures 4 and 5.

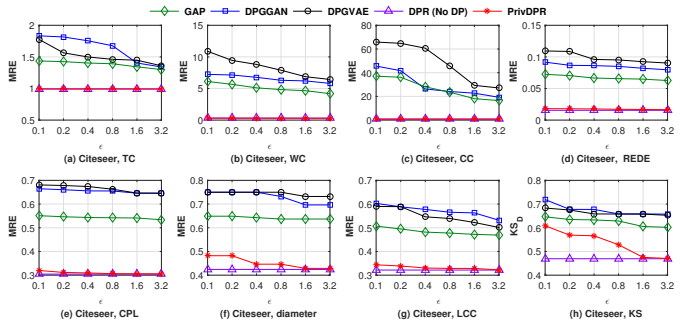

Figure 3: Privacy budget on Citeseer

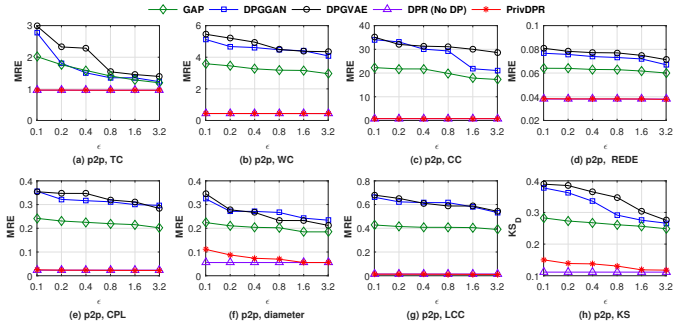

Figure 4: Privacy budget on p2p

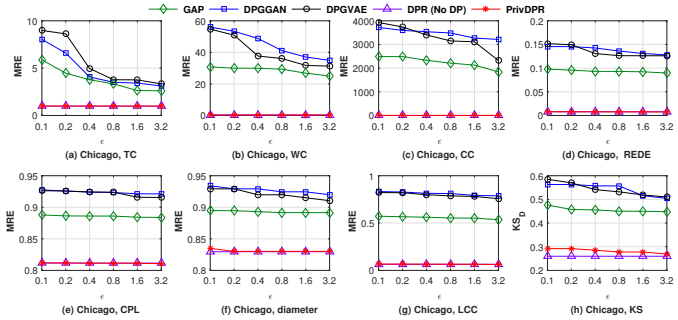

Figure 5: Privacy budget on Chicago

Table 4: AUC Scores for Link Prediction under $\epsilon = 0.1$. Bold: best

Algorithms	AUC	Datasets				
		Cora	Citeseer	p2p	Chicago	Amazon
GAP	Mean	0.4873	0.4781	0.4859	0.4915	0.4792
	SD	0.0371	0.0213	0.0092	0.0093	0.0127
DPGGAN	Mean	0.4914	0.4926	0.5005	0.4980	0.4978
	SD	0.0157	0.0108	0.0115	0.0091	0.0058
DPGVAE	Mean	0.4893	0.4977	0.4982	0.5014	0.4813
	SD	0.0083	0.0156	0.0091	0.0064	0.0063
PrivDPR	Mean	**0.5066**	**0.5099**	**0.5084**	**0.5020**	**0.5009**
	SD	**0.0070**	**0.0100**	**0.0052**	**0.0054**	**0.0025**

Appendix F Link Prediction

A higher value of AUC indicates better utility. From Tables 4 and 5, PrivDPR consistently shows the highest AUC scores across all datasets. This indicates that it is the most effective algorithm for link prediction while maintaining node-level privacy. DPGGAN and DPGVAE, which both offer edge-level privacy, exhibit similar performance levels with $\epsilon = 0.1$ and $\epsilon = 3.2$. GAP has the lowest AUC scores compared to the other algorithms, indicating that it is the least effective for link prediction at both $\epsilon = 0.1$ and $\epsilon = 3.2$. In the context of SD, a smaller value reflects greater stability in the algorithm. PrivDPR exhibits the lowest SD across all datasets under $\epsilon = 0.1$, and also demonstrates the lowest SD for the Cora and Amazon datasets under $\epsilon = 3.2$. This suggests that PrivDPR possesses strong stability. DPGVAE and DPGGAN exhibit relatively low SD in most datasets, signifying stable performance. Although GAP exhibits significant stability on the Chicago dataset under $\epsilon = 3.2$, its overall performance is comparatively lower.

Table 5: AUC Scores for Link Prediction under $\epsilon = 3.2$. Bold: best

Algorithms	AUC	Datasets				
		Cora	Citeseer	p2p	Chicago	Amazon
GAP	Mean	0.4978	0.4897	0.4973	0.5001	0.4913
	SD	0.0138	0.0173	0.0214	**0.0078**	0.0071
DPGGAN	Mean	0.5189	0.5027	0.5079	0.5031	0.5041
	SD	0.0136	**0.0102**	0.0060	0.0112	0.0040
DPGVAE	Mean	0.5068	0.5139	0.5045	0.5139	0.5012
	SD	0.0063	0.0152	**0.0035**	0.0152	0.0017
PrivDPR	Mean	**0.5535**	**0.5658**	**0.5986**	**0.5738**	**0.5917**
	SD	**0.0057**	0.0110	0.0052	0.0094	**0.0013**

Table 6: Micro-F1 Scores for Node Classification under $\epsilon = 0.1$ and $\epsilon = 3.2$. Bold: best

Algorithms	Micro-F1	$\epsilon = 0.1$		$\epsilon = 3.2$	
		Cora	Citeseer	Cora	Citeseer
GAP	Mean	0.3097	0.2094	0.3151	0.2238
	SD	0.0250	**0.0010**	**0.0011**	0.0150
DPGGAN	Mean	0.2690	0.1855	0.2849	0.2042
	SD	0.0150	0.0161	0.0096	0.0080
DPGVAE	Mean	0.2915	0.2075	0.3026	0.2172
	SD	0.0088	0.0149	0.0148	**0.0030**
PrivDPR	Mean	**0.3185**	**0.2242**	**0.3188**	**0.2266**
	SD	**0.0087**	0.0144	0.0170	0.0099

Appendix G Node Classification

A higher Micro-F1 score indicates better utility. PrivDPR achieves the highest average Micro-F1 scores on the Cora and Citeseer datasets under both $\epsilon = 0.1$ and $\epsilon = 3.2$. This indicates PrivDPR is an effective algorithm for privacy-preserving node classification. DPGVAE follows with slightly poor Micro-F1 scores but shows stable performance. GAP scores slightly lower than PrivDPR and DPGVAE, and yet performs well especially with larger privacy budgets ($\epsilon = 3.2$). DPGGAN obtains the lowest Micro-F1 scores, particularly struggling with smaller privacy budgets ($\epsilon = 0.1$). In terms of SD, under $\epsilon = 0.1$, it is observed that PrivDPR achieves the most stable result on Cora, while GAP achieves the most stable result on Citeseer. Additionally, both DPGVAE and PrivDPR demonstrate relatively stable performance across both datasets. Under $\epsilon = 3.2$, GAP achieves the most stable result on Cora, while DPGVAE achieves the most stable result on Citeseer. Nevertheless, DPGGAN displays relatively stable performance across both Cora and Citeseer datasets.