DPGS: Differentially Private Graph Synthesis via Graph Summarization

Xun Ran[1], Qingqing Ye[1,*], Jian Lou[2], Haibo Hu[1]
[1]The Hong Kong Polytechnic University
[2]Sun Yat-sen University
qi-xun.ran@connect.polyu.hk,qqing.ye@polyu.edu.hk,louj5@mail.sysu.edu.cn,haibo.hu@polyu.edu.hk

ABSTRACT

Many real-world systems, such as social and email networks, can be represented as graphs. Analyzing these structures is crucial for various applications but raises significant privacy concerns. Differential privacy (DP) provides a formal framework for protecting sensitive graph data, ensuring minimal impact from changing individual nodes or edges on analysis results. A canonical approach in this context is to generate a synthetic graph under DP constraints, while preserving the characteristics of original graph, to support downstream graph analysis. However, existing graph synthesis solutions either perturb the adjacency matrix directly, leading to high noise sensitivity, or use alternative encoding methods that suffer from significant information loss. To address this, we propose DPGS, a novel framework for Differentially Private Graph Synthesis via graph summarization. By extracting a summarized representation of the original graph, DPGS encodes structural patterns before applying DP, enhancing noise resilience while preserving key properties. Experiments on real-world datasets show that DPGS outperforms existing methods in maintaining structural features under privacy constraints, leading to effective graph analysis with enhanced utility.

1 INTRODUCTION

A wide range of complex systems can be modeled as graphs, including domains such as social interactions [1], scholarly communication networks [2], and recommendation platforms [3]. The study of graph-structured data constitutes a fundamental component of numerous scientific and industrial applications [4]. For example, Amazon recommends products by analyzing co-purchase networks, where users (i.e., nodes) are linked by shared purchases (i.e., edges) [5]. Nevertheless, the sensitive nature of graph data makes direct analysis without privacy safeguards a significant concern.

A traditional approach to preserving privacy in graph analysis is anonymization, which removes identifying information from nodes [6, 7]. However, prior studies have shown that anonymized graphs remain vulnerable to deanonymization attacks when adversaries have access to auxiliary information [8, 9].

As a gold standard in the privacy community, differential privacy (DP) [10–13] has been widely applied to protect graph data [14, 15]. Its central idea is to bound the influence of any single node or edge on the outcome of an analysis. Prior research on differentially private graph analysis has largely focused on task-specific algorithms, including methods for releasing degree distributions [16], counting subgraphs [17], and community detection [18]. However, such solutions are usually task-specific — for a different task, dedicated DP solutions must be designed from scratch. In contrast to task-specific methods, releasing a differentially private synthetic graph that preserves the original graph's semantic structure offers a general, task-agnostic solution. This method is more advantageous than task-specific algorithms because it supports a wide range of downstream graph analysis tasks.

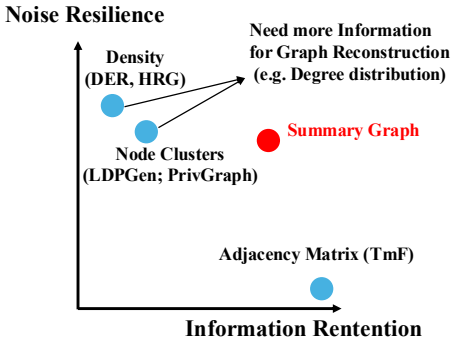


Figure 1: Comparison of Graph Encoding Methods

Despite its advantages, designing effective solutions for accurate synthetic graph generation remains a non-trivial task. The core difficulty lies in the inherent trade-off between information retention and noise resilience. Fine-grained approaches, such as those that perturb the adjacency matrix, preserve rich structural details but are highly sensitive to noise, resulting in degraded accuracy under stringent privacy budgets. In contrast, coarse-grained or aggregated approaches, including clustering and hierarchical models, offer greater robustness to noise but inevitably sacrifice fine-grained information, thereby limiting reconstruction fidelity.

Figure 1 summarizes existing works along these two dimensions. Approaches such as Top-m Filter (TmF) [19], which perturb the adjacency matrix directly, lie at the high-information but low-resilience end of the spectrum. Although these methods preserve rich structural details, perturbing individual edges introduces substantial noise effects. Density-based exploration and reconstruction (DER) [20] enhances noise resilience by grouping dense regions and applying quadtree-based aggregation; however, it struggles in sparse regions and is computationally intensive. Hierarchical Random Graph (HRG) [21] further improves noise resilience by encoding the graph into a hierarchical form, though at the cost of structural distortion and additional overhead. Node clustering methods, such as LDPGen [22], aggregate graph information before noise injection, thereby improving robustness but introducing approximation errors that limit reconstruction accuracy. More recent approaches, such as PrivGraph [23], additionally leverage external statistics (e.g., degree distributions), which may create further privacy risks and distortions.

Our Proposal. In this work, we propose DPGS, a novel approach that balances noise resilience and structural preservation through

graph summarization [24], which represents a graph using supernodes and superedges, thereby simplifying its structure while retaining essential connectivity patterns. Figure 2 shows an example of transforming an input graph (Figure 2(a)) into a summary graph (Figure 2(b)). By perturbing this compressed representation, our method reduces the dimensionality and thus the noise impact, while maintaining sufficient structural information for high-quality synthetic graph generation. Graph summarization is widely used across social network analysis, biological networks, and recommender systems, making it a promising avenue for privacy-preserving graph synthesis.

Representing a graph at a higher level of abstraction by summarizing its structural patterns, graph summarization allows us to introduce less perturbation to satisfy DP. Specifically, by grouping nodes into supernodes and aggregating connections into superedges, summarization reduces the total number of elements (compared to the original adjacency matrix), which directly lowers the function's global sensitivity—the maximum change caused by modifying a single edge or node. This reduction enables the addition of less noise to achieve the same level of privacy.

Moreover, summarization preserves key structural information by capturing both intra-group and inter-group relationships, ensuring that essential connectivity patterns (such as community structures, dense regions, and cross-group links) are retained. As a result, even though fine-grained details are abstracted, the summary still holds sufficient information to support accurate reconstruction of graph properties or generate synthetic graphs that reflect meaningful global patterns.

Designing a method that balances utility and privacy remains challenging. Many existing graph summarization approaches, such as random search [23, 24], operate solely on structural signals and struggle to incorporate node features effectively. In contrast, GNNs naturally integrate both structure and features through message aggregation, enabling more meaningful node representations and better initial partitions. However, this advantage comes with privacy risks, as aggregation induces data dependencies that can amplify noise under DP constraints [25]. To address this trade-off, we propose a hybrid approach: we first leverage GNNs to obtain feature-aware initial clusterings, then refine them via random search. The GNN-induced noise is mitigated through a spectral clustering module, while the refinement phase employs the Exponential Mechanism to reduce noise accumulation and accelerate convergence.

Following the idea, DPGS first extracts a summarized representation of the graph, capturing both local and global structures. We then perturb this summary instead of the raw adjacency matrix, allowing for a more stable and noise-resilient encoding. To ensure fidelity, DPGS separately processes dense substructures and sparse interconnections, applying different perturbation strategies tailored to each. Finally, a post-processing step refines the synthetic graph, ensuring consistency and enhancing utility.

Contributions. In summary, our main contributions are threefold:

- We propose DPGS, a novel differentially private synthetic graph generation framework that leverages **graph summarization** techniques to improve noise resilience while preserving structural integrity.
- We propose a private selection strategy that uses the exponential mechanism to achieve differentially private graph summarization while balancing utility and privacy.
- We conduct extensive experiments across multiple datasets and evaluation metrics to demonstrate the effectiveness of DPGS.

The remainder of this paper is organized as follows. In Section 2 and 3, we introduces preliminaries and the studied problem. Section 4 presents the proposed scheme and Section 5 gives the theoretical analysis. The experimental results are reported and analyzed in Section 6. The related works are analyzed in Section 7. In the Section 8, the conclusion is drawn.

2 PRELIMINARIES

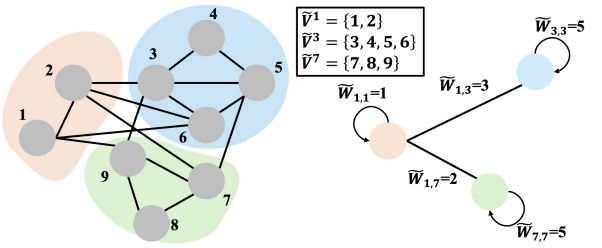
In this section, we introduce preliminaries of our study. We first present graph summarization techniques, which serve as the building block of this work, followed by the background of differential privacy and graph neural networks.

2.1 Graph Summarization

Input graph: Consider an undirected graph $G = (V, E, \boldsymbol{W}, \boldsymbol{X})$ where $V = \{v_1, v_2, \ldots, v_p\}$ is the set of p node, $E \subseteq V \times V$ is the edge set and $\boldsymbol{W}$ is the adjacency (weight) matrix. We assume that G is undirected without self-loops: $\boldsymbol{W}_{i,j} > 0$, if $(i, j) \in E$ and $\boldsymbol{W}_{i,j} = 0$ if $(i, j) \notin E$. Finally, $\boldsymbol{X} \in \mathbb{R}^{p \times n} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p\right]^T$ is the feature matrix, where each row vector $\boldsymbol{x}_i \in \mathbb{R}^n$ is the feature vector associated with one of p nodes in the graph G. A graph can be conveniently represented by Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{W}\boldsymbol{1})$ is the degree matrix. The graph Laplacian is well established as a foundation for embedding, clustering, and semi-supervised learning. Consequently, Laplacian-based representations provide particularly suitable building blocks for graph algorithms. We refer to the nodes and edges in G as subnodes and subedges, respectively, in order to distinguish them from the supernodes and superedges that appear in the summary graphs discussed below.

Summary graph: A summary graph $\tilde{G} = (\tilde{V}, \tilde{E}, \tilde{\boldsymbol{W}}, \tilde{\boldsymbol{X}})$ of an original graph $G = (V, E, \boldsymbol{W}, \boldsymbol{X})$ is defined by a set of supernodes $\tilde{V}$, a set of superedges $\tilde{E}$, a weight matrix $\tilde{\boldsymbol{W}}$, and a transformed feature matrix $\tilde{\boldsymbol{X}}$. Each superedge $(\tilde{V}^u, \tilde{V}^v) \in \tilde{E}$ connects two supernodes $\tilde{V}^u, \tilde{V}^v \in \tilde{V}$. When $u = v$, the edge $(\tilde{V}^v, \tilde{V}^v)$ represents a self-loop at supernode $\tilde{V}^v$. We denote by $\Pi_{\tilde{V}} := \{(\tilde{V}^u, \tilde{V}^v) : \tilde{V}^u, \tilde{V}^v \in \tilde{V}\}$ the set of all unordered pairs of supernodes, and $\tilde{E} \subseteq \Pi_{\tilde{V}}$. For each superedge $(\tilde{V}^u, \tilde{V}^v) \in \tilde{E}$, the corresponding entry of the weight matrix is defined by the number of subedges between the two supernodes, i.e., $\tilde{\boldsymbol{W}}_{i,j} := \left|\{(u, v) \in E : u \in \tilde{V}^i, v \in \tilde{V}^j\}\right|$. An illustrative example of a summary graph is provided in Figure 2.

Reconstructed graph: Given a summary graph $\tilde{G}$, we reconstruct a graph $\hat{G} = (\hat{V}, \hat{E}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{X}})$ following the conventional approach in [26, 27]. The set of subnodes $\hat{V}$ is obtained by taking the union of all supernodes in $\tilde{V}$, while the reconstructed edge set $\hat{E}$ includes all inter- and intra-supernode connections, defined as $\hat{E} := \{(u, v) \in \hat{V} \times \hat{V} : u \neq v,\ (\tilde{V}^u, \tilde{V}^v) \in \tilde{E}\}$. The reconstructed adjacency weights can be computed as: $\hat{\boldsymbol{W}}_{u,v} = \frac{\tilde{\boldsymbol{W}}_{u,v}}{|\Pi_{\tilde{V}^u, \tilde{V}^v}|}$, where $\Pi_{\tilde{V}^u, \tilde{V}^v} := \{(u, v) :$



(a) Input Graph G (b) Summary Graph $\tilde{G}$

Figure 2: Illustration of graph summarization. Subfigure (a) presents an example graph G. From the corresponding summary graph $\tilde{G}$ in (b), the original graph can be reconstructed. Each subnode in G is assigned to exactly one supernode $\tilde{V}^i$ in $\tilde{G}$, and the weight of each superedge $\tilde{W}_{i,j}$ represents the number of subedges connecting the two associated supernodes.

$u \neq v,\ u \in \tilde{V}^u,\ v \in \tilde{V}^v\}$ denotes the set of all possible subedges between the two corresponding supernodes.

2.2 Differential Privacy

Differential privacy (DP) [10] is a rigorous framework for privacy-preserving data analysis in the curator (trusted-server) model: a trusted curator collects individuals' data and releases randomized outputs, typically by adding calibrated noise to query answers. Intuitively, DP ensures that the presence or absence of any single record has only a limited effect on the released output, thereby protecting individual privacy. Formally, we adopt the widely used relaxed definition known as (ε, δ)-differential privacy:

Definition 1 ((ε, δ)-Differential Privacy). *An algorithm $\mathcal{A}$ satisfies (ε, δ)-differential privacy, where $\varepsilon > 0$ and $\delta \in (0, 1)$, if for any two neighboring datasets D and D', we have*

$$\forall T \subseteq Range(\mathcal{A}):\ \Pr\left[\mathcal{A}(D) \in T\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathcal{A}(D') \in T\right] + \delta,$$

where $Range(\mathcal{A})$ denotes the set of all possible outputs of algorithm $\mathcal{A}$.

We consider two datasets D and D' to be *neighbors*, denoted $D \simeq D'$, if they differ in only one record. In graph settings, two graphs G and G' are considered neighbors under edge-level privacy if one can be obtained from the other by adding or removing a single edge.

Definition 2 (Edge-level (ε, δ)-Differential Privacy). *A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-Edge-DP if, for any pair of neighboring graphs G and G' and any subset of outputs $\mathcal{Z} \subseteq Range(\mathcal{M})$, the following holds:*

$$\Pr\left[\mathcal{M}(G) \in \mathcal{Z}\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathcal{M}(G') \in \mathcal{Z}\right] + \delta.$$

Gaussian Mechanism. Gaussian mechanism (GM) achieves (ε, δ)-differential privacy by adding random Gaussian noise to the function output. The noise scale depends on the *global sensitivity* Δ, defined as

$$\Delta = \max_{D \simeq D'} \left\| f(D) - f(D') \right\|_2,$$

where f is the query function, and $D \simeq D'$ are neighboring datasets that differ in a single record. The mechanism $\mathcal{A}$ adds noise sampled from a zero-mean Gaussian distribution with variance calibrated to Δ:

$$\mathcal{A}_f(D) = f(D) + \mathcal{N}\left(0, \sigma^2 \Delta^2\right),$$

where $\sigma \geq \frac{\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$ ensures (ε, δ)-DP. For vector-valued outputs, independent Gaussian noise is added to each component.

Exponential Mechanism. The Gaussian mechanism is applicable when the output of a function f is a real-valued vector. In contrast, the Exponential Mechanism (EM) [28] is designed for scenarios where the output is selected from a finite set. EM selects outputs with probabilities proportional to the exponential of their quality scores, favoring more accurate outcomes. Given an input dataset v, EM samples an output $o \in O$ according to a predefined quality function $q(v, o)$, which evaluates how suitable o is given the input v. This mechanism requires careful design of the quality function q, which maps each pair (v, o) to a real-valued utility score. The privacy analysis of EM depends on the global sensitivity of q, defined as:

$$\Delta_q = \max_{o} \max_{v \simeq v'} \left| q(v, o) - q(v', o) \right|.$$

$\mathcal{A}$ satisfies ε-differential privacy under the following equation.

$$\Pr\left[\mathcal{A}_q(v) = o\right] = \frac{\exp\left(\frac{\varepsilon}{2\Delta_q} q(v, o)\right)}{\sum_{o' \in O} \exp\left(\frac{\varepsilon}{2\Delta_q} q(v, o')\right)}$$

Composition Properties of (ε, δ)-DP. The following properties characterize how (ε, δ)-differential privacy behaves when multiple mechanisms are combined, enabling the design of complex private algorithms based on modular components.

(1) Sequential Composition. If a sequence of k randomized algorithms $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ is applied to the same dataset, and each $\mathcal{A}_i$ satisfies $(\varepsilon_i, \delta_i)$-DP, then their combination satisfies $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.
(2) Parallel Composition. Suppose the dataset is partitioned into k disjoint subsets, and each subset is processed independently by a mechanism satisfying $(\varepsilon_i, \delta_i)$-DP. Then, the overall privacy guarantee is given by $(\max_i \varepsilon_i, \max_i \delta_i)$.
(3) Post-processing Invariance. Any data-independent transformation of the output of a differentially private mechanism does not weaken the privacy guarantee. That is, if $\mathcal{A}$ satisfies (ε, δ)-DP, then for any (possibly randomized) function g, the output $g(\mathcal{A}(D))$ also satisfies (ε, δ)-DP.

2.3 Graph Neural Networks

Given a graph $G = (V, E, \boldsymbol{W}, \boldsymbol{X})$, a r-layer GNN is a parametric function that can be represented by the following operations:

$$\boldsymbol{H} = \mathrm{GNN}(\boldsymbol{X}, \boldsymbol{W}; \Theta_{\mathrm{GNN}}),$$

where $\boldsymbol{H} \in \mathbb{R}^{p \times d'}$ is the node representations. GNNs can effectively capture local topological and feature-based patterns. The resulting representations $\boldsymbol{H}$ are then used to compute soft cluster assignments, which serve as the foundation for downstream summary construction and noise injection. This not only improves the quality of the summary graph but also ensures that important graph semantics are preserved in the differentially private synthetic graph generation pipeline.

3 PROBLEM STATEMENT

A common approach to privacy-preserving graph analysis is to design dedicated DP mechanisms to specific analysis tasks, such as community discovery and node classification [21]. However, many

tasks involve iterative graph queries, each requiring allocating a portion of the privacy budget. This becomes unsustainable as the number of queries grows, since the accumulation of privacy budget consumption eventually depletes the total budget, leading to poor utility in final results.

To overcome this limitation, we adopt an alternative paradigm. Instead of designing DP mechanisms separately to each analysis task or allocating privacy budgets to excessive queries, we focus on generating a differentially private *synthetic graph* that retains the structural properties of the original graph while satisfying Edge-DP. By doing so, any downstream graph analysis task can be considered as post-processing, without consuming additional privacy budget. This enables a more scalable and practical solution for privacy-preserving graph analytics.

Threat model. In differentially private graph synthesis, we consider an adversary with unrestricted access to the released synthetic graph who attempts to infer the presence of a particular edge in the source graph. For example, given a synthetic social network, the adversary may try to decide whether two users were connected in the original data.

Given a graph G, our objective is to generate a synthetic graph $\hat{G}$ that preserves key structural properties of G while satisfying Edge-DP. Instead of directly perturbing the adjacency matrix, we leverage *graph summarization* technique to encode structural information at a higher level, enhancing noise resilience and privacy protection. The synthetic graph $\hat{G}$ enables downstream graph analysis tasks without incurring any additional privacy loss.

By summarizing the graph before applying DP, our approach can enhance noise resilience while preserving these essential structural properties. To validate the superiority of our solution, we will follow prior work [15, 19, 20] and evaluate the similarity between $\hat{G}$ and G across five metrics widely used in graph analytics, including community structure, node attributes, degree distribution, path-based connectivity, and topology.

4 DPGS: SYNTHETIC GRAPH GENERATION WITH DP

In this section, we present a synthetic graph generation framework DPGS that achieves *differential privacy* through *graph summarization*, avoiding the need to directly perturb the raw graph structure. We will first introduce an overview of DPGS in Section 4.1, with detailed implementation from Sections 4.2 to 4.4.

4.1 An Overview of DPGS

DPGS consists of two main components—**structure summarization** and **feature summarization**—followed by a **graph reconstruction** process. The key idea lies in summarizing information at a higher level of abstraction, which inherently reduces sensitivity and enables accurate reconstruction under stronger privacy guarantees. Figure 3 shows an overview of DPGS.

Structure Summarization. To privately capture structural patterns, we design a GNN-based spectral clustering algorithm with DP guarantees. Unlike traditional methods that cluster solely based on structure, we incorporate node features to enhance clustering quality, leveraging the homophily property in real-world networks. To further improve assignment accuracy without exhausting the privacy budget, we introduce a *refinement step using the exponential mechanism* on a small set of uncertain nodes, amplifying privacy through selective intervention. The output is a noisy but informative *summary edge matrix* $\bar{\boldsymbol{W}}$, representing the connectivity between supernodes with carefully added Gaussian noise. This summarization not only improves the privacy-utility trade-off but also significantly reduces the dimensionality of the graph.

Feature Summarization. To complement the summarized structure, we construct a *compressed feature matrix* $\bar{\boldsymbol{X}}$ that preserves semantic information from the original features. Instead of perturbing raw features or training gradients—which may degrade utility or accumulate noise—we apply *objective perturbation* to a carefully designed convex optimization problem. This guarantees a one-shot DP feature transformation that maintains consistency with the summary graph structure.

Graph Reconstruction. Finally, we reconstruct a synthetic graph based on the summarized outputs. Using the learned cluster assignments, we propagate both structural and feature information from the summary graph back to the original node space. The summary structure and features are projected to obtain node-level similarities, which are then combined to infer the presence of edges. As this reconstruction relies solely on differentially private summary representations, it incurs no additional privacy cost.

To summarize, DPGS achieves differentially private graph synthesis by first compressing and perturbing high-level summaries, and then performing utility-preserving reconstruction. This design minimizes sensitivity and avoids direct noise injection into fine-grained structures, offering improved accuracy under tight privacy constraints.

4.2 Structure Summarization

Spectral Clustering. We propose a differentially private spectral clustering module, DP-SCGNN, which jointly leverages graph topology and node features to generate clustering-aware representations. Motivated by the homophily property of real-world networks [29], our method encourages nodes with similar attributes and strong structural connectivity to be grouped together. Given a graph $G = (V, E, \boldsymbol{W}, \boldsymbol{X})$, to enable privacy-preserving representation for clustering, following the approach proposed in [30], we adopt a multi-scale GNN-based encoder with injected Gaussian noise:

$$\boldsymbol{H} = \texttt{COMBINE}\left(\left\{\text{MLP}^{(k)}(\texttt{Norm}(\boldsymbol{X}^{(k)} + \boldsymbol{\eta}^{(k)}))\right\}_{k=0}^{K}\right), \tag{1}$$

where $\boldsymbol{X}^{(k)} = \left(\boldsymbol{W}^{\top}\right)^{k} \boldsymbol{X}^{(0)}$ denotes the k-hop aggregated node features, and $\boldsymbol{\eta}^{(k)} \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$ is the Gaussian noise added to enforce differential privacy. The operator `Norm` performs ℓ_2 normalization, and `COMBINE` integrates representations across different multi-layer perceptron (MLP) layers.

The embedding matrix $\boldsymbol{H}$ is then fed to a multilayer perceptron with a softmax output layer to produce soft cluster assignments: for each node i, the MLP maps the node embedding $\boldsymbol{h}_i \in \mathbb{R}^n$ to the i-th column of the assignment matrix $\boldsymbol{P} \in \mathbb{R}_{+}^{k \times p}$.

$$\boldsymbol{P} = \text{MLP}_{\text{cluster}}(\boldsymbol{H}; \boldsymbol{\Theta}_{\text{MLP}}), \tag{2}$$

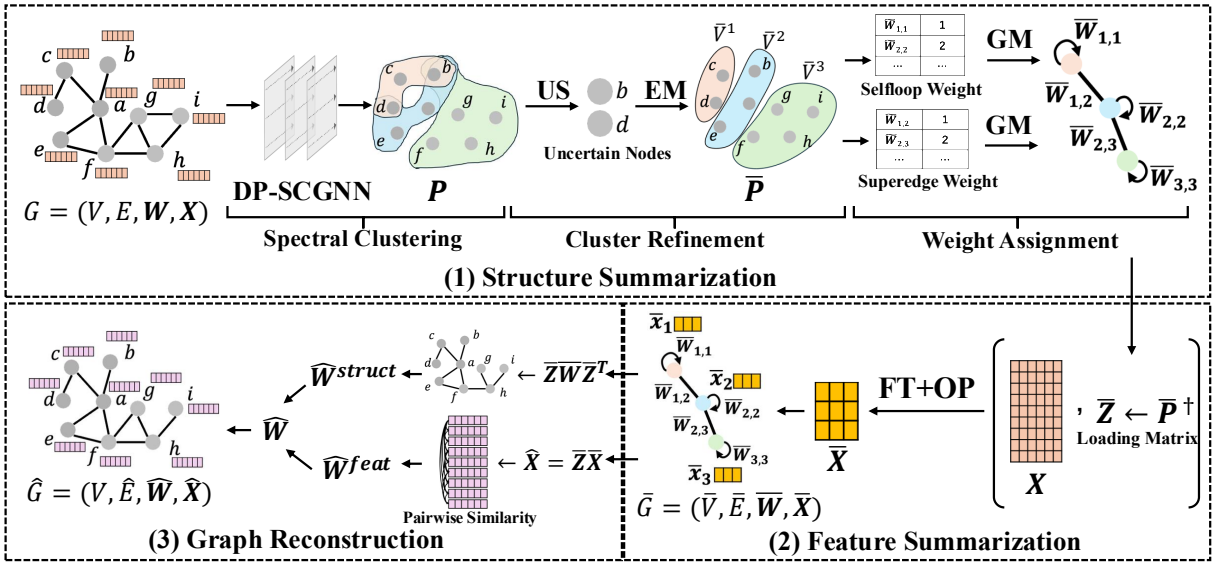


Figure 3: Workflow of DPGS. The framework consists of three components: (1) Structure Summarization, where the input graph $G = (V, E, W, X)$ is clustered using DP-SCGNN, resulting in a soft assignment matrix P. A subset of high-uncertainty nodes (US) is selected and refined using the exponential mechanism (EM), yielding a hard cluster assignment $\bar{P}$. Gaussian noise (GM) is then added to the aggregated intra- and inter-cluster edge weights to obtain the structure summary and the weight matrix $\bar{W}$; (2) Feature Summarization, where node features are transformed via feature transformation (FT) and objective perturbation (OP) into a compressed matrix $\bar{X}$ that aligns with the structure summary. The pseudo-inverse of $\bar{P}$ is used to compute the loading matrix $\bar{Z}$; (3) Graph Reconstruction, where a synthetic graph $\hat{G} = (\hat{V}, \hat{E}, \hat{W}, \hat{X})$ is reconstructed. Structural similarity $\hat{W}^{\text{struct}}$ and feature similarity $\hat{W}^{\text{feat}}$ are combined to infer edge existence.

where the softmax activation of the MLP guarantees that $P_{i,j} \in [0, 1]$ and enforces the constraints $P\mathbf{1}_k = \mathbf{1}_p$. Additionally, a temperature parameter can be introduced to the softmax function to adjust the sharpness of the assignment distribution, thereby controlling the degree to which each column $P_{:,i}$ approximates a one-hot vector—effectively tuning the level of fuzziness in the cluster assignments. We jointly optimize Θ_{GNN} and Θ_{MLP} by minimizing an unsupervised two-term loss $\mathcal{L}_u$ that approximates the relaxed MinCut objective [31]:

$$\mathcal{L}_u = \mathcal{L}_c + \mathcal{L}_o = \underbrace{-\frac{\operatorname{tr}\left(P^T \tilde{W} P\right)}{\operatorname{tr}\left(P^T \tilde{D} P\right)}}_{\mathcal{L}_c} + \underbrace{\left\| \frac{P^T P}{\|P^T P\|_F} - \frac{I_k}{\sqrt{k}} \right\|_F}_{\mathcal{L}_o}$$

where $\tilde{W} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \in \mathbb{R}^{p \times p}$, $\tilde{D}$ denotes the normalized degree matrix D, $\operatorname{tr}(\cdot)$ computes the trace of a matrix, and $\|\cdot\|_F$ indicates the Frobenius norm and k is the number of clusters. Minimizing $\mathcal{L}_c$ promotes the clustering of nodes with strong connectivity. When $\tilde{W}_{i,j}$ is large, the inner product $\langle P_{:,i}, P_{:,j} \rangle$ is encouraged to be large as well, thereby promoting similar cluster assignments for connected nodes. To avoid degenerate minima of $\mathcal{L}_c$, we add an orthogonality term $\mathcal{L}_o$ that enforces near-orthogonal cluster assignments and promotes balanced cluster sizes.

Remark. To enable differentially private training of DP-SCGNN, whose loss function depends on raw graph data, we adopt the DP-SGD mechanism [32]. At each training step, the Gaussian mechanism is applied to privatize the per-example gradients before updating model parameters. However, applying DP-SGD in the GNN context typically requires injecting substantial noise due to the interconnected nature of graph data. In the following sections, we present a formal privacy analysis and show that training the spectral clustering component satisfies differential privacy.

Cluster Refinement. To improve the reliability of cluster assignments under differential privacy, we design a refinement procedure that selectively corrects ambiguous assignments while preserving the graph's structural coherence. Instead of applying refinement to all nodes—which would lead to excessive privacy budget consumption—we focus on a small, highly uncertain subset of nodes and reassign them using the exponential mechanism. This design balances utility and privacy through targeted intervention.

As shown in Algorithm 1, we begin by measuring the uncertainty of each node's soft assignment using entropy (Lines 2-4). For node i, with assignment vector $P_{:,i} \in \mathbb{R}^k$, we compute:

$$u_i = -\sum_{c=1}^{k} P_{c,i} \log P_{c,i}. \tag{3}$$

These uncertainty scores are normalized into a probability distribution $\pi_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)}$, from which we sample a subset $\mathcal{R}$ of r nodes without replacement. This sampling concentrates effort on the most ambiguous nodes. For each sampled node $i \in \mathcal{R}$, we identify a small set of plausible candidate clusters by selecting the top-m entries in $P_{:,i}$, denoted as $C_i \subseteq \{1, \ldots, k\}$. To determine the most structurally appropriate cluster for node i, we define a utility score function over each candidate $c \in C_i$, which counts how many of i's neighbors are

confidently assigned to c:

$$s_i(c) \leftarrow \sum_{j:\boldsymbol{W}_{i,j}=1} \mathbb{I}\left[\arg\max_{c'} \boldsymbol{P}_{c',j} = c\right] \tag{4}$$

where $\{j : \boldsymbol{W}_{i,j} = 1\}$ denotes the 1-hop neighbors of node i in the adjacency matrix $\boldsymbol{W}$, and $\mathbb{I}[\cdot]$ is the indicator function (Lines 8-11). We then sample the final cluster using the exponential mechanism:

$$\Pr\left[c_i = c\right] \propto \exp\left(\frac{\epsilon \cdot s_i(c)}{2}\right), \quad \text{for } c \in C_i, \tag{5}$$

where ε is the privacy budget and sensitivity $\Delta s = 1$ (Line 12). To finalize the assignment, we update $\boldsymbol{P}_{:,i}$ by retaining the original score of the selected cluster c_i and setting the remaining entries to zero (Line 13). For nodes not selected for refinement ($i \notin \mathcal{R}$), we similarly retain only the maximum entry. This enforces that the refined matrix $\bar{\boldsymbol{P}}$ is a orthogonal assignment matrix in which each column retains only one non-zero entry (Lines 15-17). This enables more interpretable downstream processing, while ensuring that the privacy budget is spent effectively.

Algorithm 1: ClusterRefinement

Input: Soft assignment matrix $\boldsymbol{P} \in \mathbb{R}^{k\times p}$, adjacency matrix $\boldsymbol{W} \in \{0,1\}^{p\times p}$, refinement size r, candidate size m, privacy budget ϵ
Output: Refined assignment matrix $\bar{\boldsymbol{P}} \in \mathbb{R}^{k\times p}$

1 // **Uncertainty-based sampling**
2 **for** $i = 1$ **to** p **do**
3 $\quad u_i \leftarrow -\sum_{c=1}^{k} \boldsymbol{P}_{c,i} \log \boldsymbol{P}_{c,i}$
4 Normalize: $\pi_i \leftarrow \frac{\exp(u_i)}{\sum_{j=1}^{p}\exp(u_j)}$
5 Sample $\mathcal{R} \subset \{1,\ldots,p\}$ of size r from multinomial($\{\pi_i\}$) without replacement
6 // **Refine uncertain nodes**
7 Set $\bar{\boldsymbol{P}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{k\times p}$
8 **for** *each* $i \in \mathcal{R}$ **do**
9 $\quad C_i \leftarrow$ indices of top-m values in $\boldsymbol{P}_{:,i}$
10 $\quad$ **for** *each* $c \in C_i$ **do**
11 $\quad\quad s_i(c) \leftarrow \sum_{j:\boldsymbol{W}_{i,j}=1} \mathbb{I}\left[\arg\max_{c'} \boldsymbol{P}_{c',j} = c\right]$
12 $\quad$ Sample $c_i \in C_i$ with probability $\propto \exp\left(\frac{\epsilon\cdot s_i(c)}{2}\right)$
13 $\quad$ Set $\bar{\boldsymbol{P}}_{c_i,i} \leftarrow \boldsymbol{P}_{c_i,i}$
14 // **Orthogonalize assignments**
15 **for** *each* $i \notin \mathcal{R}$ **do**
16 $\quad c_i \leftarrow \arg\max_c \boldsymbol{P}_{c,i}$
17 $\quad$ Set $\bar{\boldsymbol{P}}_{c_i,i} \leftarrow \boldsymbol{P}_{c_i,i}$
18 **return** $\bar{\boldsymbol{P}}$

Summary Weight Assignment. Based on the node clustering results $\bar{\boldsymbol{P}}$, our aim is to operate on a summary graph $\bar{G}$ constructed from clustered nodes. In this graph, edges are categorized into intra-supernode (within a cluster) or inter-supernode (across clusters) edges. Since intra-supernode edges often dominate, treating these two types separately mitigates noise amplification. Instead of perturbing each edge, we add Gaussian noise to aggregated edge counts: self-loop weights for intra-supernode connections and superedge weights for inter-supernode links. This aggregation reduces sensitivity and yields a compact, noise-resilient representation of the original graph structure.

As shown in Algorithm 2, we begin by grouping nodes into supernodes based on the cluster assignment matrix $\bar{\boldsymbol{P}} \in \mathbb{R}_{+}^{k\times p}$. Each node is assigned to the cluster u with the highest membership score:

$$\bar{V}^u = \left\{v_i \mid \arg\max_{c'} \bar{\boldsymbol{P}}_{c',i} = u\right\}, \quad \forall u \in \{1,\ldots,k\}.$$

This results in a partition of the node set into k disjoint clusters, which serve as the supernodes of the summary graph (Lines 1-2).

To define the connectivity between supernodes, we compute the normalized edge count between each pair (u, v) based on the original adjacency matrix $\boldsymbol{W} \in \{0,1\}^{p\times p}$. To ensure differential privacy, we perturb the raw edge counts using Gaussian noise. For $u \neq v$, the edge weight is defined as:

$$\bar{\boldsymbol{W}}_{u,v} = \frac{\sum_{v_i\in\bar{V}^u, v_j\in\bar{V}^v} \boldsymbol{W}_{i,j} + \mathcal{N}(0,\sigma^2)}{|\bar{V}^u|\cdot|\bar{V}^v|}.$$

For self-loops (intra-cluster connectivity), the weight is:

$$\bar{\boldsymbol{W}}_{u,u} = \frac{2\sum_{i<j,\, v_i,v_j\in\bar{V}^u} \boldsymbol{W}_{i,j} + \mathcal{N}(0,4\sigma^2)}{|\bar{V}^u|(|\bar{V}^u|-1)}.$$

Note that the sensitivity of counting edges is 1, since adding or removing a single edge affects at most one pairwise count (Lines 3-8).

To preserve symmetry, we mirror each entry across the diagonal: $\bar{\boldsymbol{W}}_{v,u} \leftarrow \bar{\boldsymbol{W}}_{u,v}$ (Line 9). Finally, to ensure validity under differential privacy, we clip any negative weights to zero (Line 10):

$$\bar{\boldsymbol{W}}_{u,v} \leftarrow \max(0, \bar{\boldsymbol{W}}_{u,v}).$$

The resulting matrix $\bar{\boldsymbol{W}} \in \mathbb{R}^{k\times k}$ defines a connected, undirected, and non-negative weighted summary graph that can serve as the structural backbone for downstream graph reconstruction or representation tasks.

Algorithm 2: SummaryWeightAssignment

Input: Adjacency matrix $\boldsymbol{W} \in \{0,1\}^{p\times p}$, cluster assignment matrix $\bar{\boldsymbol{P}} \in \mathbb{R}_{+}^{k\times p}$, Gaussian noise scale σ
Output: Noisy summary graph weight matrix $\bar{\boldsymbol{W}} \in \mathbb{R}^{k\times k}$

1 **for** $u = 1$ **to** k **do**
2 $\quad \bar{V}^u \leftarrow \{v_i \mid \arg\max_{c'} \bar{\boldsymbol{P}}_{c',i} = u\}$
3 **for** $u = 1$ **to** k **do**
4 $\quad$ **for** $v = u$ **to** k **do**
5 $\quad\quad$ **if** $u \neq v$ **then**
6 $\quad\quad\quad \bar{\boldsymbol{W}}_{u,v} \leftarrow \frac{\sum_{v_i\in\bar{V}^u, v_j\in\bar{V}^v} \boldsymbol{W}_{i,j} + \mathcal{N}(0,\sigma^2)}{|\bar{V}^u|\cdot|\bar{V}^v|}$
7 $\quad\quad$ **else**
8 $\quad\quad\quad \bar{\boldsymbol{W}}_{u,u} \leftarrow \frac{2\sum_{i<j, v_i,v_j\in\bar{V}^u} \boldsymbol{W}_{i,j} + \mathcal{N}(0,4\sigma^2)}{|\bar{V}^u|(|\bar{V}^u|-1)}$
9 $\quad\quad \bar{\boldsymbol{W}}_{v,u} \leftarrow \bar{\boldsymbol{W}}_{u,v}$; // Symmetrize
10 $\quad\quad \bar{\boldsymbol{W}}_{u,v} \leftarrow \max(0, \bar{\boldsymbol{W}}_{u,v})$; // Normalize
11 **return** $\bar{\boldsymbol{W}}$

4.3 Feature Summarization

Our objective then is to derive a suitable transformation F such that $\tilde{\boldsymbol{X}} = F(\boldsymbol{X})$, i.e., it generates the summary graph's feature matrix from the original graph's features. It is essential that $\tilde{\boldsymbol{X}}$ remains similar to $\boldsymbol{X}$ with only bounded information loss. We adopt the method of Kumar et al. (2023) [33] to learn a feature transformation

while bounding the information loss. This is done by minimizing the objective function:

$$\min_{\tilde{X}} f(\tilde{X}) = \operatorname{tr}\left(\tilde{X}^T Z^T L Z \tilde{X}\right) + \frac{\alpha}{2}\|Z\tilde{X} - X\|_F^2. \tag{6}$$

where $Z \in \mathbb{R}^{p\times k}$ is the pseudo inverse of the assignment matrix P, known as the loading matrix. Here, the first term of the objective function imposes the smoothness property on the summary graph [33], and the second term acts as a regularizer. This regularization summarizes the feature matrix of a larger graph $X \in \mathbb{R}^{p\times n}$ into a smaller one $\tilde{X} \in \mathbb{R}^{k\times n}$.

Objective Perturbation. According to the objective function in Eq. (6), directly solving it would involve dependencies on the raw data, such as the loading matrix Z, the Laplacian matrix L, and the feature matrix X, all of which are derived from the original graph. It is thus necessary to consider how to incorporate DP mechanism to learn differentially private feature matrix $\bar{X}$. To avoid potential structural damage caused by input perturbation and noise accumulation resulting from gradient perturbation, we perturb the objective funcion as

$$\min_{\bar{X}} f(\bar{X}) = \operatorname{tr}(\bar{X}^T \bar{L}_s \bar{X}) + \frac{\alpha}{2}\|\bar{Z}\bar{X} - X\|_F^2 + \operatorname{tr}(B^T \bar{X}). \tag{7}$$

Here, $\bar{L}_s$ and $\bar{Z}$ are the Laplacian matrix and loading matrix derived from the summarized graph structure $\bar{W}$ and the private assignment matrix $\bar{P}$, respectively. Since the structure summarization process already satisfies DP, these two matrices are used to replace the structure-related data in Eq. (6). To ensure the privacy of the feature matrix, $\operatorname{tr}(B^T\bar{X})$ is introduced. The third term of the private objective function adds noise into $\bar{X}$, where $B \sim \mathcal{N}(0, \Delta_B^2\sigma^2\mathbb{I})$ is the noise matrix. After feature normalization, we have $\Delta_B \le \max_i \|X_i\|^2 \le 2k$.

Optimization. As $\bar{L}_s$ and $\bar{Z}^T\bar{Z}$ are the positive semi-definite and definite matrices, the Hessian of $\bar{X}$, i.e., $\nabla^2 f(\bar{X}) = 2\bar{L}_s + \alpha\bar{Z}^T\bar{Z}$ is a positive definite matrix. This implies that Eq. (7) defines a strongly convex optimization problem. Consequently, a closed-form solution can be obtained by setting its gradient to zero, i.e., $2\bar{L}_s\bar{X} + \alpha\bar{Z}^T(\bar{Z}\bar{X} - \bar{X}) + \frac{1}{\alpha}B = 0$. Then we have

$$\bar{X} = \left(\frac{2}{\alpha}\bar{L}_s + \bar{Z}^T\bar{Z}\right)^{-1}\left(\bar{Z}^T X - \frac{1}{\alpha}B\right). \tag{8}$$

This optimization allows us to inject noise only once while directly solving for the optimal solution, effectively avoiding noise accumulation and preventing direct perturbation to the graph structure. Then, by integrating the summarized structure with the corresponding summarized features, the summary graph $\bar{G} = (\tilde{V}, \bar{E}, \bar{W}, \bar{X})$ is obtained. The whole graph summarization procedure is presented in Algorithm 3, which includes three key steps, namely structure summarization (Lines 2-4), feature summazization (Lines 6-10), and returning summary graph (Lines 11-12).

4.4 Graph Reconstruction

Given the noisy summary weight matrix $\bar{W} \in \mathbb{R}^{k\times k}$, the compressed node features $\bar{X} \in \mathbb{R}^{k\times d}$, and the loading matrix $\bar{Z} \in \mathbb{R}^{p\times k}$, we reconstruct the original graph structure without requiring any additional information. The key idea is to propagate structural and semantic signals from the summary graph back to the original node space, thereby recovering the adjacency matrix in a privacy-preserving manner. The workflow is shown in Algorithm 4, which

Algorithm 3: GraphSummarization

Input: Adjacency matrix $W \in \{0,1\}^{p\times p}$, feature matrix $X \in \mathbb{R}^{p\times n}$, number of clusters k, noise scale σ, privacy budget ε

Output: Private summary graph $\bar{G} = (\tilde{V}, \bar{E}, \bar{W}, \bar{X})$

1 // **Structure Summarization.**
2 $P \leftarrow$ DP-SCGNN(W, X, k, ε_1) ; `// GNN-based soft clustering`
3 $\bar{P} \leftarrow$ ClusterRefinement$(P, W, r, m, \varepsilon_2)$; `// Entropy sampling + Exponential Mechanism`
4 $\bar{W} \leftarrow$ SummaryWeightAssignment$(W, \bar{P}, \sigma)$; `// Noisy edge aggregation`
5 // **Feature Summarization.**
6 Normalize feature matrix X: column-wise:$X_{:,j} \leftarrow X_{:,j}/\|X_{:,j}\|_2, \quad \forall j \in \{1,\dots,n\}$
7 Compute Laplacian matrix: $\bar{L}_s \leftarrow$ Laplacian$(\bar{W})$;
8 Compute loading matrix: $\bar{Z} \leftarrow \bar{P}^{\dagger}$;
9 Generate noise: $B \sim \mathcal{N}(0, \Delta_B^2\sigma^2\mathbb{I})$;
10 Compute the summarized feature matrix:

$$\bar{X} = \left(\frac{2}{\alpha}\bar{L}_s + \bar{Z}^T\bar{Z}\right)^{-1}\left(\bar{Z}^T X - \frac{1}{\alpha}B\right)$$

// **Return Summary Graph.**
11 Set $\tilde{V} = \{1,\dots,k\}$, $\bar{E} = \{(u,v) \mid \bar{W}_{u,v} > 0\}$;
12 **return** $\bar{G} = (\tilde{V}, \bar{E}, \bar{W}, \bar{X})$;

involves computing structure-based similarity (Line 2), feature-based similarity (Lines 4-7), and fusion of scores (Lines 11-19), as detailed below.

Structure-based similarity. We first reconstruct pairwise structural similarity among original nodes by projecting the noisy supernode weights back to the fine-grained space:

$$\hat{W}^{\text{struct}} = \bar{Z}\bar{W}Z^{\top}.$$

This step propagates the differentially private structure information to the original graph level.

Feature-based similarity. We then reconstruct the node features:

$$\hat{X} = \bar{Z}\bar{X},$$

and compute pairwise feature similarity via cosine similarity:

$$\hat{W}^{\text{feat}}_{i,j} = \frac{\hat{X}_i^{\top}\hat{X}_j}{\|\hat{X}_i\| \cdot \|\hat{X}_j\| + \xi},$$

where $\xi > 0$ is a small constant to ensure numerical stability. This captures semantic relationships between nodes that may not be reflected in the structural signal.

Fusion of scores. We combine both sources of similarity into a unified score matrix:

$$\hat{W}^{\text{score}} = \beta \cdot \hat{W}^{\text{struct}} + (1-\beta) \cdot \hat{W}^{\text{feat}},$$

where $\beta \in [0,1]$ controls the trade-off between structure and features.

Then, we threshold the fused scores to obtain the reconstructed binary adjacency matrix:

$$\hat{W}_{i,j} = \begin{cases} 1 & \text{if } \hat{W}^{\text{score}}_{i,j} \ge \tau, \\ 0 & \text{otherwise,} \end{cases}$$

where τ is a tunable threshold hyperparameter. We define the reconstructed node set as $V = \{v_1,\dots,v_p\}$, where p is the number of rows in Z. The predicted edge set is given by:

$$\hat{E} = \{(v_i, v_j) \mid \hat{W}_{i,j} = 1\}.$$

This reconstruction procedure leverages only the available summary outputs, ensuring that no additional privacy budget is consumed. The integration of structure and feature signals enables accurate and utility-preserving recovery of the original graph topology.

Algorithm 4: GraphReconstruction

Input: Summary edge matrix $\bar{\boldsymbol{W}} \in \mathbb{R}^{k\times k}$, summary feature matrix $\bar{\boldsymbol{X}} \in \mathbb{R}^{k\times d}$, loading matrix $\boldsymbol{Z} \in \mathbb{R}^{p\times k}$, fusion weight $\beta \in [0,1]$, threshold τ
Output: Reconstructed graph $\hat{G}$

// **Structure-based similarity**
$\hat{\boldsymbol{W}}^{\text{struct}} \leftarrow \boldsymbol{Z} \cdot \bar{\boldsymbol{W}} \cdot \boldsymbol{Z}^{\top}$
// **Feature-based similarity**
$\hat{\boldsymbol{X}} \leftarrow \boldsymbol{Z} \cdot \bar{\boldsymbol{X}}$
for $i = 1$ **to** p **do**
  for $j = 1$ **to** p **do**
    $\hat{\boldsymbol{W}}^{\text{feat}}_{i,j} \leftarrow \frac{\bar{\boldsymbol{X}}_i^{\top}\bar{\boldsymbol{X}}_j}{\|\bar{\boldsymbol{X}}_i\|\cdot\|\bar{\boldsymbol{X}}_j\|+\xi}$; // Cosine similarity
// **Fusion**
$\hat{\boldsymbol{W}}^{\text{score}} \leftarrow \beta \cdot \hat{\boldsymbol{W}}^{\text{struct}} + (1-\beta)\cdot \hat{\boldsymbol{W}}^{\text{feat}}$
// **Thresholding (binary reconstruction)**
$\hat{E} \leftarrow \emptyset$
for $i = 1$ **to** p **do**
  for $j = 1$ **to** p **do**
    if $\hat{\boldsymbol{W}}^{score}_{i,j} \geq \tau$ **then**
      $\hat{\boldsymbol{W}}_{i,j} \leftarrow 1$
      Add (v_i, v_j) to $\hat{E}$; // Add predicted edge
    else
      $\hat{\boldsymbol{W}}_{i,j} \leftarrow 0$
$V = \{v_i \mid \exists\, u \in \{1,\ldots,k\},\ \bar{\boldsymbol{P}}_{u,i} \neq 0\}$.
return $\hat{G} = (V, \hat{E}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{X}})$

5 PRIVACY ANALYSIS

Recall in Figure. 3, DPGS consists of two main components: graph summarization and graph reconstruction. Among them, only the summarization components involve direct access to the sensitive input data. Therefore, to analyze the privacy guarantee of DPGS, it suffices to analyze the privacy properties of the graph summarization algorithm. The Structure Summarization consists of three components, i.e., DP-SCGNN, cluster refinement and summary weight assignment, which consume privacy budgets of ε_{11}, ε_{12}, ε_{13}, respectively. The Feature Summarization consume privacy budget ε_2. Based on the composition of differential privacy, we can establish the following theorem:

Theorem 1 (Privacy Guarantee of DPGS). *Given a graph $G = (V, E, \boldsymbol{W}, \boldsymbol{X})$, where E is the edge set and $\boldsymbol{X}$ is the feature matrix, DPGS satisfies differential privacy if and only if its* GraphSummarization *algorithm satisfies differential privacy. Specifically:*

- *Algorithm 3* GraphReconstruction*, composed of structural clustering, weight assignment, and feature summarization, satisfies $(\varepsilon_{11}+\varepsilon_{12}+\varepsilon_{13}+\varepsilon_2, \delta)$-Edge-DP.*
- *In Algorithm 3* GraphReconstruction*, for the node features $\boldsymbol{X}$ as the sensitive dataset, the feature summarization step alone satisfies (ε_2, δ)-DP under user-level changes.*

Therefore, the Graph Summarization module of DPGS simultaneously protects edge-level structure and node-level features, and the overall mechanism preserves differential privacy via composition.

Proof. For brevity, we provide only a proof sketch; the full proof is given in the appendix. Under the edge-level definition where two neighboring graphs differ in exactly one edge, each components in Algorithm 3 with guassian mechanism is differentially private. In particular, the Exponential Mechanism in Cluster Refinement is applied to utility functions whose sensitivity (i.e., $\Delta_s(i)$) is 1 with respect to a single edge change. This guarantees that the selection probability distribution changes by at most a multiplicative factor of $e^{\varepsilon_{12}}$. it can be safely combined with approximate (ε, δ)-DP mechanisms via composition. The feature module consumes privacy budget ε_2 and injects Gaussian noise calibrated to the sensitivity of the feature data. Here, the sensitivity is defined as the maximum change in one row of the feature matrix (i.e., $\max_i \|\boldsymbol{X}_i\|^2$), corresponding to a single user's feature vector. Hence, by the definition of differential privacy, the Gaussian mechanism ensures (ε_2, δ)-DP under row-level adjacency, providing user-level privacy protection for node features.

As shown in Section 2, by the sequential composition theorem, the combined mechanism of structure and feature summarization satisfies $(\varepsilon_{11}+\varepsilon_{12}+\varepsilon_{13}+\varepsilon_2, \delta)$-DP. Since GraphReconstruction is a post-processing step, the same bound applies to the overall DPGS mechanism. □

We then present the privacy accounting method for training DP-SCGNN. A common challenge when training GNNs under differential privacy lies in the dependencies among subgraphs in the training set. These dependencies amplify the sensitivity of the aggregation process, often leading to larger noise requirements and complicating privacy accounting. To address this issue, we adopt the accounting method in [35], which reduces inter-node dependency and improves the overall privacy-utility trade-off. This is formally proven using Rényi Differential Privacy (Rényi-DP) [36] as follows.

Theorem 2 (Privacy Guarantee for any r-Layer GNN). *Let N be the number of training instances V_{tr}, N_k be the maximum degree of the input graph, r be the number of GNN layers, and m be the batch size during training. For any choice of the noise standard deviation $\sigma > 0$ and clipping threshold η, every iteration t of DP-SGD is (α_ρ, γ) node-level Rényi DP, where:*

$$\gamma = \frac{1}{\alpha_\rho - 1} \ln \mathbb{E}_\rho \left[\exp\left(\alpha_\rho(\alpha_\rho - 1) \cdot \frac{2\rho^2\eta^2}{\sigma^2} \right) \right],$$

$$\rho \sim \text{Hypergeometric}\left(N, \frac{N_k^{r+1} - 1}{N_k - 1}, m \right).$$

Hypergeometric *denotes the standard hypergeometric distribution [37]. By the standard composition theorem for Rényi Differential Privacy [36], over T iterations, spectral clustering is $(\alpha_\rho, \gamma T)$-Edge-Rényi DP.*

Proof. Please refer to the appendix. □

6 EXPERIMENTAL EVALUATION

In this section, we first introduce our experiment setup in Section 6.1, and then present the experimental results and our analysis in Section 6.2.

6.1 Experiment Setup

6.1.1 Datasets. Table 1 characterizes the 6 real-world datasets used to benchmark DPGS.

Table 1: Summary statistics of used datasets.

Name	#Nodes	#Edges	#Features	#Classes
Cora	2708	5278	1433	7
CiteSeer	3327	9104	3703	6
PubMed	19717	44324	500	3
Coauthor	18333	163788	6805	15
DBLP	17716	105734	1639	4

6.1.2 Experimental Design. We evaluate the performance of our approach against a range of representative baselines across three categories. For **graph summarization baselines**: we include four state-of-the-art methods: Local Variation Neighbourhood (LVN), proposed in [38]; SSuMM, massive graph summarization method proposed in [39]; Kron, which applies Schur complement-based reduction of the Laplacian matrix [40]; and Algebraic Distance (AD), which leverages connection strength between nodes [41]. For **Edge-DP GNN baselines**, we compare with DP-GNN [35], DPGCN [42], GAP [30], LPGNet [43], and Eclipse [44]. For **differentially private graph synthesis baselines**: For the private synthesis methods that do not incorporate node features, we consider five baselines: PrivHRG [15], DER [20], TmF [19], LDPGen [22], and PrivGraph [23]. To ensure a fair comparison, we adopt hyperparameter configurations recommended in the original studies. Since LDPGen was originally designed for the local differential privacy setting, we adapt its implementation to the standard DP model for consistency across all methods. For the privacy budget, we allocate the total budget evenly across all components.

6.1.3 Evaluation Indicators. **Synthesis Quality.** To comprehensively evaluate the quality of the synthetic graphs, we adopt two representative metrics that capture different aspects of structural fidelity. Specifically, we use *relative error (RE)* of modularity and clustering coefficient to assess the preservation of **global topological structure**, and *eigenvector centrality (EVC)*-based metrics to evaluate the retention of **node-level importance**.

The RE of modularity and clustering coefficient are computed as:

$$\mathrm{RE}_{\mathrm{Mod}} = \frac{|\hat{Q} - Q|}{\max(\xi, Q)}, \quad \mathrm{RE}_{\mathrm{CC}} = \frac{|\hat{Y} - Y|}{\max(\xi, Y)},$$

where Q and $\hat{Q}$ denote the modularity of the original and synthetic graphs, Y and $\hat{Y}$ denote their average clustering coefficients, and ξ is a small constant to prevent division by zero.

To evaluate node importance preservation, we rank all nodes by EVC and compute two comparison metrics: (1) the proportion of overlapping nodes among the top 1% most influential nodes, and (2) the mean absolute error (MAE) of their centrality scores:

$$\mathrm{Overlap}_{\mathrm{Node}} = \frac{|N \cap \hat{N}|}{|N|}, \quad \mathrm{MAE}_{\mathrm{EVC}} = \frac{1}{T}\sum_{i=1}^{T} |\hat{s}_i - s_i|,$$

where N and $\hat{N}$ are the sets of top 1% nodes ranked by EVC in the original and synthetic graphs, and s_i, $\hat{s}_i$ are their corresponding EVC scores. This combination of metrics allows us to jointly assess structural and semantic fidelity of the generated graphs. Please refer to the appendix for more evaluation metrics and results.

Downstream Task Performance. To evaluate the utility of the synthetic graphs beyond structural similarity, we assess their effectiveness on two representative downstream graph learning tasks: node classification and link prediction. These tasks reflect how well the synthetic graph supports semantic inference and connectivity modeling, and are widely used to gauge the practical value of graph representations. Strong performance in these settings indicates that the synthetic graphs preserve not only topological patterns but also task-relevant signals essential for learning meaningful representations. We use accuracy to evaluate node classification and the Area Under the ROC Curve (AUC) to assess link prediction. Accuracy quantifies the proportion of correctly predicted node labels, while AUC measures the model's ability to distinguish between positive and negative links.

6.2 Experimental Results and Analysis

6.2.1 Synthesis Quality. **Topological Structure.** Figures 4 and 5 report the relative error of modularity and clustering coefficients on Cora,CiteSeer, PubMed and Coauthor datasets, capturing how well each method preserves structural properties under varying privacy budgets ε. Overall, DPGS achieves the lowest RE across both datasets and metrics, indicating superior preservation of topological features.

On CiteSeer, DPGS reduces RE of Modularity from over 0.8 (typical for TmF, DER, and LDPGen) to below 0.5 at $\varepsilon = 1.0$, with further improvements as ε increases. Similar trends are observed in clustering coefficients, where DPGS consistently outperforms all baselines across privacy levels. In contrast, methods such as TmF and PrivHRG show minimal sensitivity to ε, likely due to their direct matrix perturbation or rigid modeling assumptions.

PrivGraph does not perform as well as DPGS, particularly on CiteSeer. Although it benefits from community-level modeling, its hard clustering strategy in the community adjustment phase forces a full re-clustering of all nodes at each iteration. This introduces cumulative noises and optimization instability, often leading to suboptimal convergence and inconsistent structural recovery. These limitations are reflected in its relatively high RE in both modularity and clustering coefficient metrics, especially under small ε.

On PubMed — a larger and sparser dataset, DPGS maintains RE(Modularity) below 0.4 for $\varepsilon \geq 2$, while all other methods, including PrivGraph, exceed 0.6. Its RE on clustering coefficients is also the lowest and most stable, demonstrating robustness to both graph size and sparsity.

Node-level Importance. Figures 6 and 7 present the comparison results of node-level importance preservation using two complementary metrics: the overlap of eigenvector centrality nodes between the original and synthetic graphs, and the MAE of their EVC scores. Across four datasets, DPGS consistently outperforms all baselines once the privacy budget exceeds $\varepsilon = 1.5$. This improvement can be attributed to its summarization-based design:

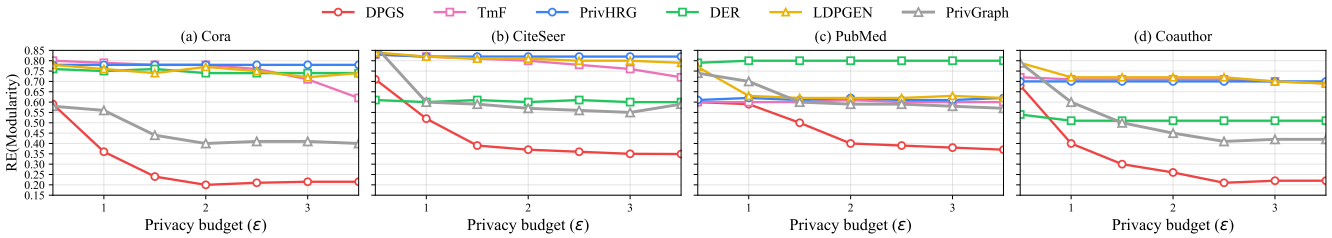


Figure 4: Overall results on RE(Modularity).

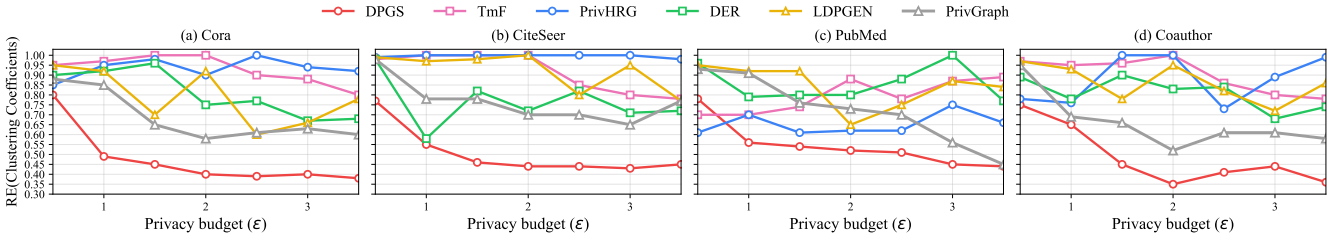


Figure 5: Overall results on RE(Clustering Coefficients).

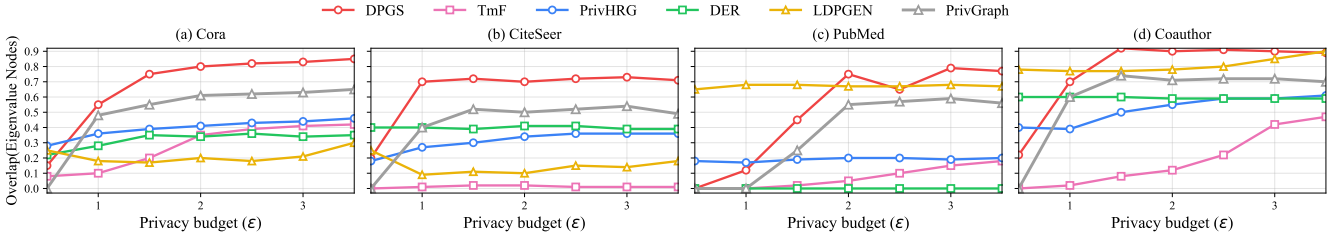


Figure 6: Overall results on Overlap(Eigenvalue Nodes).

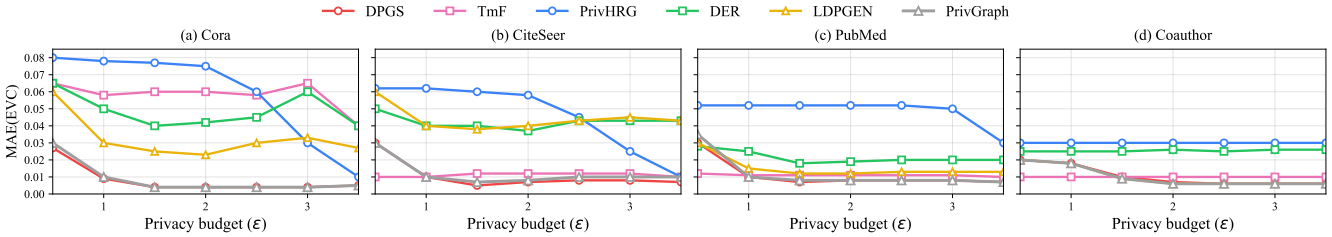


Figure 7: Overall results on MAE(EVC).

by clustering structurally similar nodes and performing edge reconstruction at the community level, DPGS mitigates the effect of perturbation and better maintains global spectral properties. As a result, it yields significantly higher node overlap (e.g., > 0.7 on both CiteSeer and PubMed for $\varepsilon \geq 2$) and lower MAE compared to TmF, PrivHRG, and DER.

At low privacy budgets ($\varepsilon = 0.5$), the overlap scores of DPGS remain modest as expected, since noisy clustering leads to degraded reconstruction fidelity. However, the performance significantly improves with increasing ε, while other baselines remain flat or improve marginally. This trend suggests that DPGS makes the best use of a larger budget to enhance structural recovery.

In contrast, PrivGraph and LDPGen, while competitive on certain datasets, exhibit limited adaptability. For example, PrivGraph's overlap plateaus on PubMed, likely due to its reliance on fixed statistics and hard clustering, which restrict flexibility under noise. LDPGen performs relatively well on PubMed but struggles on CiteSeer, possibly because the EVC score differences between top nodes are less pronounced, making ranking sensitive to noise.

TmF and PrivHRG, which directly perturb edge presence or rely on coarse probabilistic hierarchies, consistently underperform due to accumulated noise and limited expressiveness. DER, although data-adaptive, shows unstable results across datasets and generally falls short of DPGS's performance.

6.2.2 Downstream Task Performance. **Node Classification.** To evaluate the utility of DPGS under different privacy constraints, we conduct node classification experiments across four benchmark datasets with privacy budgets $\varepsilon \in 0.1, 3.5$ under the Edge-DP setting. As reported in Table 2, DPGS consistently achieves the highest accuracy across all datasets and privacy levels. Although the performance margin over strong baselines (e.g., Eclipse and LPGNet) is narrowed down compared to earlier settings, DPGS remains the

Table 2: Node classification performance across datasets under different ε values.

Dataset	ε	DPGCN	Eclipse	LPGNet	DP-GNN	GAP	DPGS
Cora	0.1	36.95 ± 0.42	65.11 ± 0.38	48.14 ± 0.21	27.36 ± 1.08	34.66 ± 0.25	**67.50 ± 0.20**
	3.5	70.24 ± 1.00	65.32 ± 0.61	70.50 ± 0.91	39.54 ± 0.71	57.40 ± 0.20	**69.68 ± 0.38**
CiteSeer	0.1	35.23 ± 1.06	63.07 ± 0.07	49.38 ± 0.87	28.00 ± 0.29	32.87 ± 0.47	**64.14 ± 0.15**
	3.5	61.17 ± 0.68	63.68 ± 0.48	63.20 ± 0.52	31.20 ± 0.43	56.41 ± 0.41	**67.80 ± 0.35**
PubMed	0.1	54.65 ± 0.62	71.72 ± 0.24	66.82 ± 0.88	62.24 ± 0.64	59.99 ± 0.30	**72.17 ± 0.25**
	3.5	61.06 ± 0.97	71.55 ± 0.82	74.07 ± 0.46	65.70 ± 1.13	72.01 ± 0.16	**77.73 ± 0.28**
Coauthor	0.1	63.26 ± 0.30	88.01 ± 0.14	67.46 ± 0.64	52.90 ± 0.77	80.21 ± 0.63	**90.09 ± 0.12**
	3.5	86.03 ± 0.13	89.01 ± 0.12	89.77 ± 0.10	59.01 ± 0.78	90.47 ± 0.20	**91.11 ± 0.06**

most robust method, particularly under low ε values, where most other methods suffer significant performance degradation.

For instance, on the Cora and CiteSeer datasets with $\varepsilon = 0.1$, DPGS attains almost 67% and 64% accuracy respectively. This demonstrates DPGS's effectiveness in preserving task-relevant information even under strict privacy constraints. On larger graphs like PubMed and Coauthor, DPGS maintains its superiority, achieving over 72% and 91% accuracy under $\varepsilon = 0.1$, respectively — outperforming all competitors.

The effectiveness of DPGS can be attributed to its two components: it first summarizes the input graph into low-sensitivity structural and semantic representations, and then applies privacy-preserving mechanisms at this intermediate level. By avoiding direct noise injection into the input graph or training gradients, DPGS significantly reduces utility loss. In contrast, baselines such as DP-GNN and DPGCN suffer from accumulated noise across multiple stages of training or inference, leading to noticeable degradation in predictive performance.

Link Prediction. Since most private learning methods are primarily designed for node classification, we evaluate the effectiveness of DPGS on the link prediction task, comparing it with several summarization-based baselines, including SSumM, LVN, Kron, and AD. Following established practice, we vary the summarization ratio from 5% to 30% for Cora and CiteSeer, and from 1% to 5% for the larger PubMed dataset to assess scalability under compression.

As shown in Table 3, DPGS consistently achieves the highest AUC across all datasets and summarization levels. On Cora, for instance, DPGS reaches an AUC of 82.42% at a 30% summary size and maintains a strong lead even when compressed to 5%. Similarly, on CiteSeer and PubMed, DPGS outperforms all baselines by a notable margin, especially under extreme compression (e.g., 1% summary on PubMed), where it still achieves 72.17% AUC compared to AD's 68.15%.

These improvements can be attributed to DPGS's capability of jointly summarizing both structural patterns and node features into a compact representation with reduced sensitivity. Privacy-preserving noise is applied only once during the summarization phase, and no additional noise is introduced during reconstruction. In contrast, competing methods either disregard node attributes (e.g., LVN, Kron) or apply heuristics that lead to cumulative distortions. DPGS effectively balances utility and privacy, especially under aggressive compression, making it more robust and scalable in practice.

6.2.3 Robustness against MIA Attacks. We evaluate the resilience of DPGS to membership inference attacks (MIA) under the setting of [45], where the adversary aims to determine whether a given node was part of the target model's training set. Such attacks typically exploit overfitting by using confidence scores from a shadow model trained on a similar distribution. As shown in Table 4, all methods yield AUC values close to 50%, indicating that the attack model performs no better than random guessing. Notably, DPGS maintains consistently low AUC values across all datasets and privacy levels, with minimal variation as ε increases. This demonstrates that DPGS effectively limits information leakage, offering strong membership privacy even under relaxed privacy budgets.

6.2.4 The impact of β. We further investigate the impact of varying the weighting parameter β in the score matrix under a fixed privacy budget of $\epsilon = 3.5$. The results shown in Figure 8 exhibit a consistent *bell-shaped trend* across all datasets: accuracy increases from relatively low values at $\beta = 0$, reaches a peak at an intermediate value of β, and subsequently decreases as β approaches 1. This observation confirms that an appropriate balance between structural and feature information is crucial for maximizing utility under differential privacy.

Several dataset-specific patterns can be observed. On **Cora**, the peak occurs at $\beta = 0.5$, indicating that structural and feature information contribute almost equally to the utility. In contrast, **CiteSeer** achieves its highest accuracy when β is closer to the feature side ($\beta = 0.2$), which aligns with the higher quality of its node features. This demonstrates the importance of feature information in this dataset. **PubMed** reaches its maximum accuracy around $\beta = 0.55$, suggesting a slightly stronger reliance on structural information, while still benefiting from feature signals. Finally, **Coauthor** achieves the highest performance when β is shifted toward the structural side ($\beta = 0.8$), reflecting the dataset's well-defined community structures. Importantly, at $\beta = 1$, Coauthor still maintains the best performance among all datasets, highlighting the robustness of structural information for this graph. Guided by the above analysis, we fix $\beta = 0.5$ in all experiments, which strikes a balance among fairness, privacy, and consistency while preserving utility.

7 RELATED WORK

Differentially Private Graph Neural Networks. Integrating differential privacy (DP) into Graph Neural Networks (GNNs) has gained traction due to the sensitive nature of graph data. Various methods aim to enforce edge- or node-level DP during training.

Table 3: Link prediction performance across datasets under different summary size.

Dataset	Size	Ground Truth	SSumM	LVN	Kron	AD	DPGS
Cora	30%	—	68.35 ± 0.15	70.20 ± 0.58	77.10 ± 0.24	78.02 ± 0.30	**82.42 ± 0.04**
	10%	84.31 ± 0.75	66.08 ± 0.56	67.82 ± 0.22	75.13 ± 0.13	77.02 ± 0.36	**80.21 ± 0.13**
	5%	—	63.32 ± 1.05	63.00 ± 0.95	73.28 ± 0.24	75.05 ± 0.13	**77.89 ± 0.03**
CiteSeer	30%	—	72.05 ± 0.47	71.45 ± 0.25	73.01 ± 0.14	75.13 ± 0.17	**76.86 ± 0.03**
	10%	78.62 ± 0.62	69.71 ± 0.38	69.53 ± 0.58	69.83 ± 0.36	73.88 ± 0.30	**75.66 ± 0.03**
	5%	—	62.89 ± 0.82	63.93 ± 0.64	68.13 ± 0.39	72.00 ± 0.45	**72.29 ± 0.04**
PubMed	5%	—	61.30 ± 0.83	62.10 ± 0.90	67.05 ± 0.40	77.12 ± 0.33	**81.19 ± 0.04**
	3%	83.28 ± 0.57	60.66 ± 0.58	62.13 ± 0.57	66.28 ± 0.22	72.30 ± 0.23	**75.85 ± 0.06**
	1%	—	57.35 ± 0.58	61.60 ± 0.23	66.22 ± 0.38	68.15 ± 0.45	**72.17 ± 0.02**

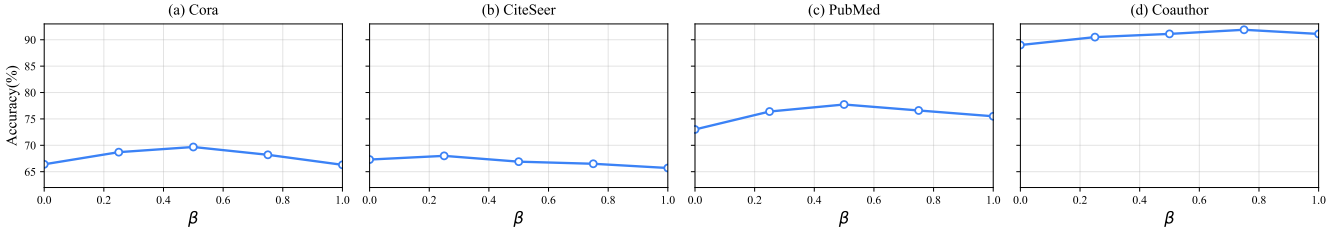


Figure 8: Overall results on β tuning.

Table 4: MIA performance across datasets.

Dataset	Method	$\varepsilon = 0.1$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 3$
Cora	GAP	50.12 ± 0.03	50.23 ± 0.04	50.42 ± 0.02	51.10 ± 0.04
	DP-GNN	50.24 ± 0.04	50.25 ± 0.02	50.20 ± 0.01	50.22 ± 0.03
	DPGS	50.18 ± 0.02	50.22 ± 0.02	50.39 ± 0.03	50.61 ± 0.04
CiteSeer	GAP	50.02 ± 0.09	50.18 ± 0.03	50.30 ± 0.05	51.58 ± 0.01
	DP-GNN	50.06 ± 0.02	50.05 ± 0.02	50.62 ± 0.04	50.67 ± 0.05
	DPGS	50.08 ± 0.03	50.08 ± 0.02	50.28 ± 0.03	50.46 ± 0.05
PubMed	GAP	50.08 ± 0.05	50.35 ± 0.04	51.12 ± 0.02	51.29 ± 0.03
	DP-GNN	50.06 ± 0.03	50.06 ± 0.04	50.35 ± 0.02	50.66 ± 0.02
	DPGS	50.29 ± 0.02	50.30 ± 0.02	50.50 ± 0.06	50.96 ± 0.02
Coauthor	GAP	50.03 ± 0.05	50.18 ± 0.04	50.52 ± 0.05	51.48 ± 0.02
	DP-GNN	50.00 ± 0.02	50.08 ± 0.04	50.57 ± 0.02	50.93 ± 0.06
	DPGS	50.23 ± 0.02	50.30 ± 0.07	50.33 ± 0.04	50.44 ± 0.05

GAP [30] perturbs the neighborhood aggregation step, supporting both edge and node DP with calibrated noise to preserve utility. DPDGC [46] decouples graph convolution into structural and feature components, enabling fine-grained gradient sensitivity control under DP-SGD.

ProGAP [47] introduces progressive noise injection to stabilize training and improve the privacy-utility trade-off. LPGNet [43] modifies message-passing rules to protect edge information without excessive noise. Eclipse [44] applies low-rank SVD to the adjacency matrix before perturbation, reducing sensitivity while retaining structure. Works such as LinkTeller [42] demonstrates that many graph model defenses remain vulnerable to inference attacks, underscoring the need for stronger protections and refined threat models.

Privacy Attacks and Threat Models on Graphs. Graph data is highly vulnerable to privacy attacks due to its rich relational structure. Adversaries may infer hidden edges via structure-based attacks, which exploit topological patterns [25, 48], or feature-based attacks, which rely solely on node attributes [49, 50]. De-anonymization attacks [51, 52] further attempt to re-identify nodes using auxiliary data and matching algorithms. DPGS mitigates such risks by disrupting correlations between observed signals and sensitive edges, reducing susceptibility to both structure- and feature-based attacks. Nevertheless, attacks outside the DP framework remain an open challenge, calling for alternative privacy notions and structural obfuscation techniques.

Graph Summarization and Coarsening. Graph summarization and coarsening reduce graph size while preserving key properties, enabling efficient storage [53], scalable analytics [26], and visualization [54]. Traditional methods rely on heuristics, whereas recent approaches adopt information-theoretic or optimization-based formulations. SSumM [39] uses the Minimum Description Length (MDL) principle to merge nodes and sparsify edges, achieving compression with fidelity guarantees. Kumar et al. [33] coarsen both structure and features by optimizing a feature-aware similarity objective, preserving semantic consistency for downstream tasks. These techniques not only improve efficiency but also complement DP by reducing data sensitivity before noise injection, making them well-suited for privacy-preserving graph analytics.

8 CONCLUSION

We propose DPGS, a novel framework for differentially private graph synthesis that integrates structure and feature summarization to enable accurate and privacy-preserving graph generation. By applying differential privacy during the summarization phase, DPGS significantly reduces sensitivity and avoids noise injection in downstream tasks. Our framework supports structure-feature fusion and uses a one-shot perturbation strategy to achieve strong utility under strict privacy budgets. Extensive experiments on node classification, link prediction, and membership inference demonstrate that DPGS consistently outperforms state-of-the-art baselines in both performance and privacy. These results validate the effectiveness of summarization-based synthesis as a promising paradigm for private graph publishing.

APPENDIX

Proof of Theorem 1

Proof: We analyze the privacy of the DPGS mechanism by decomposing the GraphSummarization procedure into its core components and applying the composition and post-processing properties of differential privacy.

(1) Structure Summarization via DP-SCGNN. The spectral clustering component injects Gaussian noise into each layer of the GNN's message passing (Eq. (2)) to achieve differential privacy:

$$\boldsymbol{\eta}^{(k)} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}).$$

This corresponds to DP-SGD over a sequence of gradient updates, and its privacy guarantee is formally captured using Rényi differential privacy. As a result, the output cluster assignment matrix $\boldsymbol{P}$ satisfies $(\varepsilon_{11}, \delta_1)$-Edge-DP.

(2) Cluster Refinement via Exponential Mechanism. For each uncertain node $i \in \mathcal{R}$, DPGS selects a refined cluster assignment using the exponential mechanism:

$$\Pr[c_i = c] \propto \exp\left(\frac{\varepsilon_2 \cdot s_i(c)}{2}\right),$$

where $s_i(c)$ counts how many neighbors of node i are confidently assigned to cluster c, and the sensitivity $\Delta_q = 1$ due to the addition/removal of a single edge. Therefore, this step satisfies ε_{12}-Edge-DP.

(3) Weight Assignment via Gaussian Mechanism. The summary weight matrix $\bar{\boldsymbol{W}}$ is computed by aggregating intra- and inter-cluster edges, followed by the addition of calibrated Gaussian noise:

$$\bar{\boldsymbol{W}}_{u,v} = \frac{\sum_{i \in \bar{V}^u, j \in \bar{V}^v} \boldsymbol{W}_{i,j} + \mathcal{N}(0, \sigma^2)}{|\bar{V}^u| \cdot |\bar{V}^v|}.$$

Given that each edge affects at most one count, the sensitivity is 1, and the Gaussian mechanism ensures $(\varepsilon_{13}, \delta_2)$-Edge-DP.

(4) Feature Summarization via Objective Perturbation. To protect the node features $\boldsymbol{X}$, DPGS solves a convex optimization problem with a perturbed objective:

$$\min_{\bar{\boldsymbol{X}}} \operatorname{tr}(\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{L}}_s \bar{\boldsymbol{X}}) + \frac{\alpha}{2} \|\bar{\boldsymbol{Z}}\bar{\boldsymbol{X}} - \boldsymbol{X}\|_F^2 + \operatorname{tr}(\boldsymbol{B}^\top \bar{\boldsymbol{X}}),$$

where $\boldsymbol{B} \sim \mathcal{N}(0, \Delta_B^2 \sigma^2 \mathbb{I})$. Since the objective is strongly convex and $\bar{\boldsymbol{L}}_s$, $\bar{\boldsymbol{Z}}$ are derived from DP-compliant summaries, the entire mechanism satisfies $(\varepsilon_2, \delta_3)$-DP with respect to $\boldsymbol{X}$ under row-level changes.

(5) Post-processing via GraphReconstruction. The final synthetic graph $\hat{G}$ is generated solely using $\bar{\boldsymbol{W}}, \bar{\boldsymbol{X}}, \bar{\boldsymbol{Z}}$, which are outputs of DP mechanisms. According to the post-processing property of DP, any function applied to these outputs does not incur additional privacy loss.

(6) Composition. By sequential composition of the above components, the total privacy budget over the structure is:

$$(\varepsilon_{11} + \varepsilon_{12} + \varepsilon_{13}, \delta_1 + \delta_2),$$

and for the feature-level protection, we have:

$$(\varepsilon_2, \delta_3)\text{-DP}.$$

Letting $\delta := \delta_1 + \delta_2 + \delta_3$ completes the proof. ∎

Proof of Theorem 2

We provide a detailed proof for the following theorem:

Theorem 2 [Privacy Guarantee for any r-Layer GNN] Let N be the number of training instances V_{tr}, N_k be the maximum number of cluster neighbors any node can have in the coarsened graph (i.e., cluster degree), r be the number of GNN layers, and m be the batch size during training. For any choice of the noise standard deviation $\sigma > 0$ and clipping threshold η, every iteration t of DP-SGD is (α_ρ, γ) node-level Rényi DP, where:

$$\gamma = \frac{1}{\alpha_\rho - 1} \ln \mathbb{E}_\rho \left[\exp\left(\alpha_\rho(\alpha_\rho - 1) \cdot \frac{2\rho^2\eta^2}{\sigma^2}\right)\right],$$

$$\rho \sim \text{Hypergeometric}\left(N, \frac{N_k^{r+1} - 1}{N_k - 1}, m\right).$$

Proof. Let G and G' be two node-adjacent graphs differing by a single node v. For a GNN with r layers, the output of each node in the training set V_{tr} is influenced by its r-hop neighborhood. Under the assumption that each node in the coarsened (clustered) graph has degree at most N_k, the number of subgraphs in which the removed node v could appear is bounded by $d = \frac{N_k^{r+1}-1}{N_k-1}$.

Let $\mathcal{B}_t$ be the mini-batch sampled at iteration t, and ρ be the number of affected subgraphs from the total d that fall into $\mathcal{B}_t$. Then, ρ follows a hypergeometric distribution Hypergeometric(N, d, m).

Assume each clipped per-node gradient has ℓ_2-norm at most η. Then the total sensitivity of the summed gradient $\mathbf{u}_t$ is upper bounded by $2\eta \cdot \rho$. Adding Gaussian noise of variance σ^2 implies by Corollary 3 of [36] that the iteration satisfies:

$$(\alpha_\rho, \gamma_\rho)\text{-RDP}, \quad \gamma_\rho = \frac{\alpha_\rho \cdot (2\rho\eta)^2}{2\sigma^2}.$$

Taking expectation over ρ:

$$\gamma = \frac{1}{\alpha_\rho - 1} \ln \mathbb{E}_\rho \left[\exp\left(\alpha_\rho(\alpha_\rho - 1) \cdot \frac{2\rho^2\eta^2}{\sigma^2}\right)\right].$$

Therefore, each iteration is (α_ρ, γ)-RDP, and over T iterations, by the composition property of Rényi DP, the full algorithm satisfies $(\alpha_\rho, \gamma T)$-RDP.

□

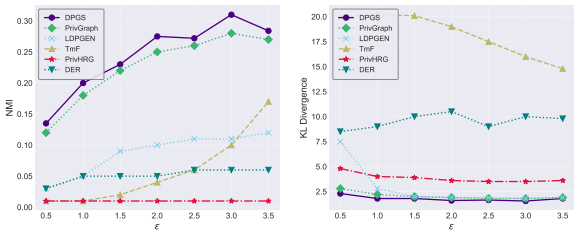
Additional Experimental results

We evaluate the quality of the synthetic graphs using two additional metrics: **Normalized Mutual Information (NMI)** and **Kullback–Leibler (KL) divergence**. NMI quantifies the alignment between community detection results on the original and synthetic graphs, while KL divergence measures the difference in degree distributions between the two graphs. Below are the formal definitions of both metrics.

Normalized Mutual Information (NMI).. Let $A = \{A_1, A_2, \ldots, A_R\}$ and $B = \{B_1, B_2, \ldots, B_S\}$ be two partitions of a graph $G = (V, E)$. Their overlap can be represented using a contingency table H, where H_{ij} denotes the number of nodes shared by clusters A_i and B_j. Let x_i (resp. y_j) be the sum of the i-th row (resp. j-th column) in H. The NMI between A and B is computed as:

$$\text{NMI}(A, B) = \frac{-2 \sum_{i=1}^{R} \sum_{j=1}^{S} H_{ij} \log\left(\frac{H_{ij} n}{x_i y_j}\right)}{\sum_{i=1}^{R} x_i \log\left(\frac{x_i}{n}\right) + \sum_{j=1}^{S} y_j \log\left(\frac{y_j}{n}\right)}. \tag{9}$$

Kullback–Leibler (KL) Divergence. Given the degree distributions $P(x)$ and $\hat{P}(x)$ of the original and synthetic graphs respectively, the



NMI on Cora KL Divergence on Cora

Figure 9: Results on NMI and KL Divergence over dataset Cora

KL divergence is defined as:

$$D_{\mathrm{KL}}(P \parallel \hat{P}) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{\hat{P}(x)} \right). \tag{10}$$

The results are shown in Figure 9.

REFERENCES

[1] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed Networks in Social Media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 1361–1370.

[2] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph Evolution: Densification and Shrinking Diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 2–es, 2007.

[3] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–51, 2023.

[4] D. Sharma, R. Shukla, A. K. Giri, and S. Kumar, "A Brief Review on Search Engine Optimization," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering*. IEEE, 2019, pp. 687–692.

[5] F. Jiang, C. K. Leung, and A. G. Pazdor, "Big Data Mining of Social Networks for Friend Recommendation," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 921–922.

[6] Y. Wang, L. Xie, B. Zheng, and K. C. Lee, "High Utility K-Anonymization for Social Network Publishing," *Knowledge and Information Systems*, vol. 41, no. 3, pp. 697–725, 2014.

[7] J. Cheng, A. W.-c. Fu, and J. Liu, "K-Isomorphism: Privacy Preserving Network Publication against Structural Attacks," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, pp. 459–470.

[8] S. Ji, P. Mittal, and R. Beyah, "Graph Data Anonymization, De-Anonymization Attacks, and De-Anonymizability Quantification: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1305–1326, 2016.

[9] J. Qian, X.-Y. Li, C. Zhang, L. Chen, T. Jung, and J. Han, "Social Network De-Anonymization and Privacy Inference with Knowledge Graph Model," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 679–692, 2017.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[11] L. Du, Z. Zhang, S. Bai, C. Liu, S. Ji, P. Cheng, and J. Chen, "AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1266–1288.

[12] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 212–229.

[13] T. Wang, J. Q. Chen, Z. Zhang, D. Su, Y. Cheng, Z. Li, N. Li, and S. Jha, "Continuous Release of Data Streams under both Centralized and Local Differential Privacy," in *ACM CCS*, 2021.

[14] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, "Sharing Graphs Using Differentially Private Graph Models," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, 2011, pp. 81–98.

[15] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially Private Network Data Release via Structural Inference," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 911–920.

[16] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate Estimation of the Degree Distribution of Private Networks," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 169–178.

[17] H. Sun, X. Xiao, I. Khalil, Y. Yang, Z. Qin, H. Wang, and T. Yu, "Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[18] T. Ji, C. Luo, Y. Guo, Q. Wang, L. Yu, and P. Li, "Community Detection in Online Social Networks: A Differentially Private and Parsimonious Approach," *IEEE transactions on Computational Social Systems*, vol. 7, no. 1, pp. 151–163, 2020.

[19] H. H. Nguyen, A. Imine, and M. Rusinowitch, "Differentially Private Publication of Social Graphs at Linear Cost," in *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2015, pp. 596–599.

[20] R. Chen, B. Fung, P. S. Yu, and B. C. Desai, "Correlated Network Data Publication via Differential Privacy," *The VLDB Journal*, vol. 23, no. 4, pp. 653–676, 2014.

[21] A. Clauset, C. Moore, and M. E. Newman, "Hierarchical Structure and the Prediction of Missing Links in Networks," *Nature*, vol. 453, no. 7191, pp. 98–101, 2008.

[22] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating Synthetic Decentralized Social Graphs with Local Differential Privacy," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 425–438.

[23] Q. Yuan, Z. Zhang, L. Du, M. Chen, P. Cheng, and M. Sun, "{PrivGraph}: differentially private graph data publication by exploiting community information," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3241–3258.

[24] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM computing surveys (CSUR)*, vol. 51, no. 3, pp. 1–34, 2018.

[25] X. Xian, T. Wu, Y. Liu, W. Wang, C. Wang, G. Xu, and Y. Xiao, "Towards Link Inference Attack against Network Structure Perturbation," *Knowledge-Based Systems*, vol. 218, p. 106674, 2021.

[26] M. Riondato, D. García-Soriano, and F. Bonchi, "Graph summarization with quality guarantees," *DMKD*, vol. 31, no. 2, pp. 314–349, 2017.

[27] K. LeFevre and E. Terzi, "Grass: Graph structure summarization," in *SDM*, 2010.

[28] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007, pp. 94–103.

[29] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[30] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "{GAP}: Differentially private graph neural networks with aggregation perturbation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3223–3240.

[31] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 551–556.

[32] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[33] M. Kumar, A. Sharma, S. Saxena, and S. Kumar, "Featured graph coarsening with similarity guarantees," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 953–17 975.

[34] X. Ran, Q. Ye, J. Lou, and H. Hu, "Differentially private implicit matrix factorization," https://github.com/marlowe518/CCS-2025/blob/main/dpgs_full.pdf, 2025.

[35] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, "Node-level differentially private graph neural networks," *arXiv preprint arXiv:2111.15521*, 2021.

[36] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[37] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions*. Wiley, 2011. [Online]. Available: https://books.google.co.in/books?id=YhF1osrQ4psC

[38] A. Loukas and P. Vandergheynst, "Spectrally approximating large graphs with smaller graphs," in *International conference on machine learning*. PMLR, 2018, pp. 3237–3246.

[39] K. Lee, H. Jo, J. Ko, S. Lim, and K. Shin, "Ssumm: Sparse summarization of massive graphs," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 144–154.

[40] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 150–163, 2012.

[41] J. Chen and I. Safro, "Algebraic distance on graphs," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3468–3490, 2011.

[42] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," in *2022 ieee symposium on security and privacy (sp)*. IEEE, 2022, pp. 2005–2024.

[43] A. Kolluri, T. Baluta, B. Hooi, and P. Saxena, "Lpgnet: Link private graph networks for node classification," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1813–1827.

[44] T. Tang, Y. Niu, S. Avestimehr, and M. Annavaram, "Edge private graph neural networks with singular value perturbation," *arXiv preprint arXiv:2403.10995*, 2024.

[45] I. E. Olatunji, W. Nejdl, and M. Khosla, "Membership inference attack on graph neural networks," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2021, pp. 11–20.

[46] E. Chien, W.-N. Chen, C. Pan, P. Li, A. Ozgur, and O. Milenkovic, "Differentially private decoupled graph convolutions for multigranular topology protection," *Advances in Neural Information Processing Systems*, vol. 36, pp. 45 381–45 401, 2023.

[47] S. Sajadmanesh and D. Gatica-Perez, "Progap: Progressive graph neural networks with differential privacy guarantees," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 596–605.

[48] Y. Zhang, M. Humbert, B. Surma, P. Manoharan, J. Vreeken, and M. Backes, "Towards Plausible Graph Anonymization," in *Proceedings of the 27th Annual Network and Distributed System Security Symposium*, 2020.

[49] C. Yang, L. Zhong, L.-J. Li, and L. Jie, "Bi-Directional Joint Inference for User Links and Attributes on Large Social Graphs," in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 564–573.

[50] N. Eagle, A. Pentland, and D. Lazer, "Inferring Friendship Network Structure by Using Mobile Phone Data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15 274–15 278, 2009.

[51] W.-H. Lee, C. Liu, S. Ji, P. Mittal, and R. B. Lee, "How to Quantify Graph De-Anonymization Risks," in *Information Systems Security and Privacy*, 2017, pp. 84–104.

[52] Y. Zhao and I. Wagner, "Using Metrics Suites to Improve the Measurement of Privacy in Graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 259–274, 2020.

[53] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka, "Compression of weighted graphs," in *KDD*, 2011.

[54] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, "Vog: Summarizing and understanding large graphs," in *SDM*, 2014.