PrivIM: Differentially Private Graph Neural Networks for Influence Maximization

Renxuan Hou, Qingqing Ye, Xun Ran, Sen Zhang, Haibo Hu
Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong SAR
renxuan.hou@connect.polyu.hk *qqing.ye@polyu.edu.hk* *qi-xun.ran@connect.polyu.hk*
senzhang@polyu.edu.hk *haibo.hu@polyu.edu.hk*

***Abstract*—Influence Maximization (IM), aiming to identify a small set of highly influential nodes in social networks, is a critical problem in graph analysis. Recently, Graph Neural Networks (GNNs) have demonstrated superior effectiveness in addressing IM. However, a trained GNN still raises significant privacy concerns, as it may expose sensitive node features and structural information. While Differential Privacy (DP) techniques have been widely applied to GNNs for node-level tasks, they cannot be directly extended to IM problems. This is because IM requires more complex structural information for training, resulting in an extremely larger DP noise scale than node-level tasks. To tackle these issues, we propose PrivIM, a novel differentially private subgraph-based GNNs framework for IM tasks, which ensures node-level DP guarantees. Within PrivIM, we design a unique dual-stage adaptive frequency sampling scheme to optimize the model utility. First, it reduces the correlation between nodes by dynamically adjusting each node's sampling probability. Then additional subgraphs are incorporated to supplement boundary structural information, enhancing utility without increasing privacy budget. Extensive experiments on six real-world datasets demonstrate that PrivIM maintains high utility in IM compared to baseline methods.**

***Index Terms*—Influence maximization, Graph neural networks, Differential privacy**

I. Introduction

In social networks, Influence Maximization (IM) aims to identify a small group of users who are likely to impact as many individuals as possible. With the prevalence of graph data, IM has been widely employed in various applications, such as social recommendation [1], [2], network monitoring [3], virus marketing [4], [5], and rumor blocking [6], [7]. Many companies, such as Amazon, now leverage IM algorithms as primary tools for promoting their products, capitalizing on "word of mouth" effects to amplify their influence. Recently, due to the powerful graph representation capability, Graph Neural Networks (GNNs) have exhibited superior effectiveness in addressing IM [8], [9].

However, training GNNs on graph data for IM tasks raises significant privacy concerns. This is particularly critical when the data is collected from individual users, as adversaries may infer sensitive node attributes or structural properties of users' graph data from the trained models. To address such concerns, the idea of Differential Privacy (DP) has been adopted for ensuring individual-level privacy in GNN-based methods. A prime technique is DP Stochastic Gradient Descent (DP-SGD) [10]–[12], which introduces calibrated noise into

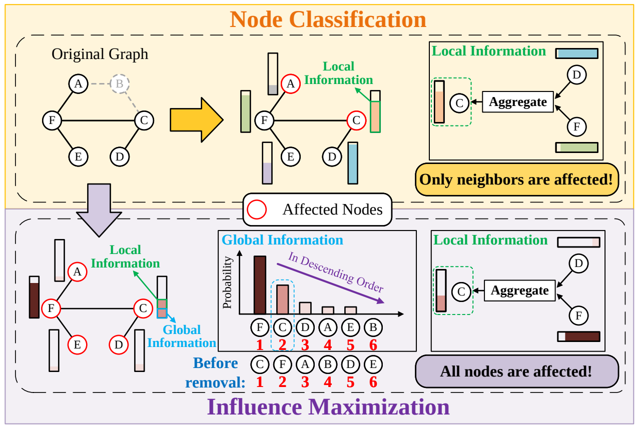


Fig. 1: An illustrative example of how removing a single node affects influence scores and alters the seed set selection process

gradients during training to preserve privacy. While most existing works focus on developing differentially private GNNs for node classification [13], [14], there remains a gap in research on IM, which inherently involves a more complex interplay of node interactions and influence propagation.

Addressing this gap is non-trivial, as directly extending approaches designed for node classification to IM faces unique challenges. Within GNNs, a node's representation is learned not only from its own features but also from the aggregated information of its neighbors. This introduces a complex data dependency issue, which poses additional challenges for IM compared to node-level tasks. To illustrate these challenges, Figure 1 shows an example using 1-layer GNN.

Example 1. *In each GNN layer, nodes aggregate the embedding vectors of their neighbors. For node-level tasks, the impact of removing a node (e.g., node Ⓑ) on other node embeddings can be assessed by simply counting its neighbors (e.g., nodes Ⓐ and Ⓒ). However, in IM tasks, removing node Ⓑ affects not only its neighbors' embeddings but also all nodes' embeddings in the graph. As illustrated in Figure 1, each node's embedding aggregates both its local neighborhood information and global information. This global information includes the rank of each node based on its influence score in descending order, which determines the probability of a node being selected in the seed set.*

Existing studies have proposed two strategies to address the data dependency issue and bound the DP noise scale during training. The first strategy aims to limit correlations between nodes by restricting the number of each node's neighbors [15], [16]. The second strategy reduces the reliance of GNNs on graph data. This is achieved by limiting the depth of GNNs or using a shallow Multi-Layer Perceptron to mimic the message-passing mechanism in GNNs [17]. However, both strategies can only be applied to node-level or edge-level tasks. Since IM is a graph-level task operating on the entire graph, it introduces the following two challenges:

- **Overwhelming DP Noise.** In IM tasks, a node's embedding is determined by the collective influence of all nodes within the graph, rather than solely by its immediate neighbors. This fundamentally differs from node-level tasks (e.g., node classification), where a node's representation primarily depends on its neighbors. Consequently, existing methods that limit neighbors to reduce noise scale are ineffective for IM tasks, as they cannot control the broader dependencies, still resulting in overwhelming DP noise.
- **Complex Graph Structure.** IM relies on comprehensive connections and rich structural information to capture the global influence of nodes across the entire network. Unlike node classification, which typically uses local neighborhood information, IM requires an understanding of the entire graph. However, methods that limit neighbors to reduce DP noise can disrupt the intricate structure, leading to overly simplified connectivity. This can weaken the model's generalization capability and hamper its performance on IM. While effective for node classification, these neighbor-constrained methods don't meet the needs of IM tasks.

To address these challenges, we propose PrivIM, a novel framework for training differentially private GNNs for IM tasks. PrivIM not only ensures node-level DP guarantees, but also preserves high model utility and is adaptable to various GNN models. Within PrivIM, we design a dual-stage adaptive frequency sampling scheme. In the first stage, we dynamically adjust sampling probabilities to reduce inter-node dependency, which minimizes the influence of individual nodes on each other while preserving the overall graph structure. In the second stage, we supplement the sampling process with additional subgraphs that capture boundary structural information, enhancing the model's utility without consuming privacy budget. We provide a comprehensive theoretical analysis of privacy and utility, and conduct extensive experiments on various public datasets, demonstrating the effectiveness of our PrivIM framework. Our main contributions are threefold.

- We propose PrivIM, a novel differentially private GNNs framework for IM tasks. To the best of our knowledge, this is the first study to solve the IM problem while satisfying rigorous node-level DP.
- We design a dual-stage adaptive frequency sampling scheme within the PrivIM framework. The first stage reduces the impact of individual nodes while maintaining the overall structure of the original graph. The second stage further improves the utility, by incorporating supplementary subgraphs from boundary areas to enrich the structural information.
- Extensive experiments conducted on six real-world datasets provide compelling evidence of the exceptional ability of our proposed methods to achieve a better trade-off between privacy and utility. These results highlight both the effectiveness and efficiency of our framework.

The rest of the paper is structured as follows. Section II describes the preliminaries. Section III presents our PrivIM framework. Section IV proposes our dual-stage frequency sampling method. Section V shows the experimental results. Section VI reviews the related work, followed by a conclusion in Section VII.

II. Preliminaries

In this section, we introduce some preliminaries on Graph Neural Networks (GNNs), Differential Privacy (DP), and Influence Maximization (IM).

A. Graph Neural Networks (GNNs)

Let $G = (V, E)$ be a graph with node set V and edge set E. In practice, a graph G is represented by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ and its node feature matrix $\mathbf{X} \in \mathbb{R}^{|V|\times d}$, where each node has a d-dimensional embedding [18]. Note that in this paper, we focus on directed graphs because they offer a more general setting, while undirected graphs can be treated as directed ones [19].

Definition 1 (GNN). *GNN follows the message-passing mechanism, where the embedding of a node is updated by recursively aggregating information from its neighbors. Formally, node v's embedding at the $(i+1)$-th layer $\mathbf{h}_v^{(i+1)}$ is updated by:*

$$\mathbf{m}_{\mathcal{N}(v)}^{(i)} = \mathrm{AGG}^{(i)}\left(\left\{\mathbf{h}_u^{(i)} | u \in \mathcal{N}(v)\right\}\right),$$
$$\mathbf{h}_v^{(i+1)} = \mathrm{UPD}^{(i)}\left(\mathbf{h}_v^{(i)}, \mathbf{m}_{\mathcal{N}(v)}^{(i)}\right),$$

where $\mathrm{AGG}(\cdot)$ *and* $\mathrm{UPD}(\cdot)$ *are the aggregation and update functions, respectively, and* $\mathcal{N}(v)$ *is node* v*'s neighborhood.*

B. Differential Privacy (DP)

Definition 2 (Node (Edge)-Level Differential Privacy [20]). *Let G and G' be two adjacent graphs that differ by only one node (edge). A randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if for all adjacent graphs G and G', and for all $\mathcal{S}$ in the output space of $\mathcal{A}$, we have:*

$$\Pr(\mathcal{A}(G) \in \mathcal{S}) \leq e^{\epsilon} \Pr(\mathcal{A}(G') \in \mathcal{S}) + \delta.$$

Note that achieving node-level DP is more challenging than edge-level DP as modifying a single node may result in the removal of $(|V|-1)$ edges in the worst case. Therefore, node-level DP generally provides stronger privacy guarantees than edge-level DP. In this paper we primarily focus on node-level DP. However, our method is flexible and can be extended to edge-level DP, allowing for adaptation based on different privacy requirements.

Technically, given two adjacent graphs G and G', if they differ by one node being replaced, then it is called bounded node-level DP [21]; if they differ by adding or removing one node, it is called unbounded node-level DP [22]. As with the previous studies on graph data [23], [24], in this work we leverage the unbounded node-level DP definition. Next, we introduce Rényi Differential Privacy (RDP) to analyze the composition of private mechanisms while giving tight results.

Definition 3 ((α, γ)-Rényi Differential Privacy [25])**.** *For two probability distributions P and Q defined over $\mathcal{R}$, the Rényi divergence [26] is defined as $\mathcal{D}_\alpha(P\|Q) = \frac{1}{\alpha-1}\log \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}\right]^\alpha$ with $\alpha > 1$. A randomized mechanism $\mathcal{A}$ is said to be (α, γ)-RDP, if*

$$\mathcal{D}_\alpha(\mathcal{A}(G)\|\mathcal{A}(G')) \leq \gamma$$

holds for all adjacent graphs G and G'.

To ensure the differential privacy of a mechanism, it is generally necessary to introduce random noise to its output. The scale of the noise should be calibrated according to the sensitivity of a specific query, which is defined below.

Definition 4 (Node-level Sensitivity [21])**.** *Given a query $g : \mathcal{G} \rightarrow \mathbb{R}^d$, g's l_2-sensitivity on any two adjacent graphs $G, G' \subset \mathcal{G}$ is defined as*

$$\Delta_g = \max_{G\sim G'} \|g(G) - g(G')\|_2.$$

The following definition presents the sequential composition property of RDP, which is especially useful for iterative mechanism design (e.g., in DP-SGD), where privacy cost accumulates over iterations.

Definition 5 (Sequential Composition of RDP [25])**.** *Let $f : \mathcal{G} \rightarrow \mathcal{R}_1$ be (α, γ_1)-RDP and $g : \mathcal{R}_1 \times \mathcal{G} \rightarrow \mathcal{R}_2$ be (α, γ_2)-RDP, then the combination of these two mechanisms, defined as $h := (f(G), g(f(G), G))$, satisfies $(\alpha, \gamma_1 + \gamma_2)$-RDP.*

Finally, we can covert an (α, γ)-RDP guarantee to the (ϵ, δ)-DP formulation using the following conversion rule.

Theorem 1 (Conversion Rule for (α, γ)-RDP [27])**.** *For $\alpha > 1$ and $\delta > 0$, and for a mechanism $\mathcal{A}$ which is (α, γ)-RDP, then $\mathcal{A}$ satisfies (ϵ, δ)-DP where*

$$\epsilon = \gamma + \log\frac{\alpha - 1}{\alpha} - \frac{\log\delta + \log\alpha}{\alpha - 1}.$$

C. Influence Maximization (IM)

The IM problem primarily revolves around modeling the influence diffusion process under a specific diffusion model within a network. As with the previous studies [4], [28], our focus is on the Independent Cascade (IC) model, which is widely recognized as the most popular influence diffusion model in the literature.

Definition 6 (Independent Cascade (IC) Model [4])**.** *Given a graph $G = (V, E)$, we assume that the edge (u, v) has a weight w_{uv} which represents its influence probability, i.e., $0 \leq w_{uv} \leq 1$. The diffusion begins from a seed set $S \subseteq V$, and S_j denotes the set of nodes that are active in time-step j, $j \in \mathbb{N}$, with $S_0 = S$. Each newly-activated vertex u in the previous step $(j - 1)$ has a single chance at the current step to influence its inactive neighbor v independently with the probability w_{uv}. The influence diffusion terminates when no more inactive nodes can be activated.*

As IC model is the most popular diffusion model, we employ it to formally define the IM problem as follows.

Definition 7 (Influence Maximization (IM) [4])**.** *Given a graph $G = (V, E)$, a positive integer k, the generic IM problem aims to select a set $S \subseteq V$ with $|S| = k$ as the seed set to maximize the influence spread $I(S, G)$ under IC model:*

$$\tilde{S} = \arg\max_{|S|=k} I(S, G),$$

where $\tilde{S}$ is the optimal seed set which produces a maximal influence spread in G.

There are many real-world applications of the IM problem [2], [3], such as social recommendations, election campaigns, and viral marketing in social networks. Our objective is to train a private GNN model, satisfying node-level DP, to solve the IM problem on large-scale graphs.

III. PrivIM Framework

In this section, we study the problem of influence maximization in the context of DP. In Section III-A, we begin by introducing our design rationale for integrating GNNs, which directly leads to the framework **PrivIM**, short for differentially **Priv**ate GNNs for **I**nfluence **M**aximization. We then elaborate on the implementations of PrivIM in Sections III-B to III-D, and finally summarize our observations in Section III-E.

A. Design Rationale

Infeasibility of traditional IM methods with DP. Traditional IM methods compute marginal gains for all nodes and select the highest one at each step. However, ensuring DP requires adding noise based on the sensitivity (i.e., the maximum gain change from adding or removing a node), which scales with the network size. As a result, the noise overwhelms the gain value, rendering the method ineffective [29]. The following example clearly illustrates this limitation.

Example 2. *For Gowalla [30], with $|V| \approx 2 \times 10^5$ nodes, the potential influence range of a node u could span the entire graph. Under the node-level DP setting with the Laplace mechanism, the noise scale is $\frac{\Delta f}{\epsilon}$, where sensitivity $\Delta f \approx 2 \times 10^5$ for the greedy algorithm. With privacy budget $\epsilon = 1$, the noise scale is about 2×10^5, while the actual gain value typically ranges from 10^0 to 10^3. As a result, the noise overwhelms the gain, making the seed selection completely ineffective.*

GNNs, with their strong graph representation capabilities, are emerging as efficient solvers for IM problems. However, ensuring node-level DP in IM tasks is challenging due to intricate node dependencies. Fortunately, GNNs can be trained on homogeneous subgraphs and applied to large networks. By leveraging this property, we can extract homogeneous subgraphs from the entire graph to reduce inter-node dependencies (Section III-B). However, this subgraph-based approach raises a critical question: *How can we compute node sensitivity, i.e.,*

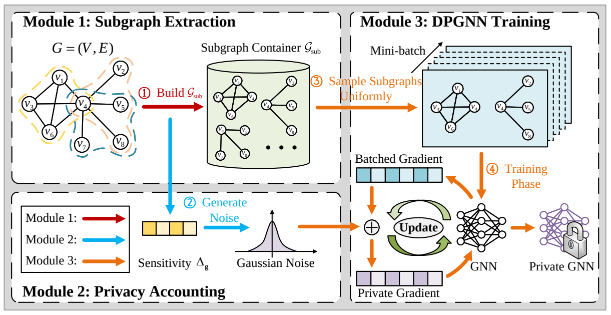


Fig. 2: Workflow of PrivIM framework

the impact of individual node changes on the GNN output across multiple subgraphs? To tackle this issue, we establish the upper bound on node sensitivity and calculate the DP noise required for privacy protection (Section III-D). Finally, inspired by Erdős' method [31], we design a probabilistic penalty loss function for seed selection and apply DP-SGD [10] to ensure rigorous node-level DP during training (Section III-C).

Overall, we design PrivIM, an integrated framework consisting of three modules, to train GNN models that satisfy node-level DP for IM problems, as shown in Figure 2. In **Subgraph Extraction Module**, we construct the subgraph container $\mathcal{G}_{\text{sub}}$ by extracting homogeneous subgraphs from original graph G. This container serves as a pool for mini-batch sampling in the training phase, enabling us to apply DP-SGD techniques. For **Privacy Accounting Module**, we analyze the upper bound of node sensitivity $\Delta_{\mathbf{g}}$ based on the sampling process in Module 1. Additionally, Gaussian noise is generated using this sensitivity $\Delta_{\mathbf{g}}$, which is subsequently leveraged in Module 3 for GNN training. Finally, in **DPGNN Training Module**, we perform mini-batch sampling from the subgraph container $\mathcal{G}_{\text{sub}}$ built in Module 1, and inject the calibrated Gaussian noise from Module 2 into the gradients. This framework enables us to achieve the private GNN model with rigorous node-level DP for IM tasks.

B. Subgraph Extraction

An intuitive way to extract subgraphs is to perform Breadth-First-Search (BFS) starting from a node v_0 randomly. However, if the starting node v_0 has a high clustering coefficient, the extracted subgraph may become excessively large due to the small-world property in social networks [32]. Training on such large subgraphs cannot achieve our goal of reducing the inter-node dependencies. To address this, we sample fixed-size subgraphs from v_0's r-hop neighbors.

A natural choice for subgraph extraction is to perform Random Walk with Restart (RWR) [33]. To further reduce the dependencies, we first limit nodes' in-degree θ and the hop number r between the starting node v_0 and any sampled node. Then, we start the random walk from v_0 to one of its neighbors randomly. We repeat this process until a fixed number of n vertices is collected. Specifically, we provide the pseudo-code of this process in Algorithm 1.

Given a graph $G = (V, E)$ and a maximum node in-degree of θ, we first project the original graph G into θ-bounded graph

Algorithm 1: Subgraph Extraction Process

```
Input: $\theta$-bounded graph $G^\theta = (V^\theta, E^\theta)$, subgraph size $n$,
       return probability of RWR $\tau$, sampling rate $q$,
       random walk length $L$, the hop number $r$
Output: Set of subgraphs $\mathcal{G}_{\text{sub}}$
1  for $v \in V^\theta$ do
       // Sample the starting node $v_0$ randomly
2      $p \leftarrow \text{Uniform}(0, 1)$
3      if $p < q$ then
4          $V_{\text{sub}} \leftarrow \emptyset, v_0 \leftarrow v, v_{\text{cur}} \leftarrow v$
5          $V_{\text{sub}}.add(v)$
6          for $l = 1, \ldots, L$ do
               // Collect nodes with RWR
7              $p \leftarrow \text{Uniform}(0, 1)$
8              if $p < \tau$ then
9                  $v_{\text{cur}} \leftarrow v_0$
10             $v_{\text{next}} \leftarrow$ Sample one of the neighbors randomly
                 from $\mathcal{N}(v_{\text{cur}}) \cap \mathcal{N}_r(v_0)$.
11             $v_{\text{cur}} \leftarrow v_{\text{next}}$
12             if $v_{\text{next}}$ is not in $V_{\text{sub}}$ then
13                 $V_{\text{sub}}.add(v_{\text{next}})$
14             if $|V_{\text{sub}}| = n$ then
15                 Extract $G_{\text{sub}}$ from $G^\theta$ with nodes in $V_{\text{sub}}$.
16                 $\mathcal{G}_{\text{sub}}.add(G_{\text{sub}})$
17                 break
18 return Set of subgraphs $\mathcal{G}_{\text{sub}}$
```

$G^\theta = (V^\theta, E^\theta)$ by randomly removing edges from nodes whose in-degree exceeds θ. This ensures that all nodes in V^θ have an in-degree of at most θ, effectively bounding the impact of individual nodes and reducing the noise scale required during training. For random walk sampling, we first select a starting node v_0 with the sampling rate q, initializing it as the current node v_{cur} (Lines 2-5). At each step l, with a probability τ, we reset v_{cur} to the starting node v_0 (Lines 6-9). The next node v_{next} is then sampled from $\mathcal{N}(v_{\text{cur}}) \cap \mathcal{N}_r(v_0)$ (Line 10), where $\mathcal{N}(v_{\text{cur}})$ represents neighbors of v_{cur}, and $\mathcal{N}_r(v_0)$ represents r-hop neighbors of v_0. This constraint ensures that the random walk remains within the r-hop neighborhood of the starting node, thereby reducing inter-node dependencies. Any newly sampled node v_{next} that is not already in the node set V_{sub} will be added to it (Lines 12-13). Once we collect n unique nodes in V_{sub} within L steps, we extract the subgraph G_{sub} from G^θ with V_{sub}. The resulting subgraph G_{sub} is subsequently added to the subgraph container $\mathcal{G}_{\text{sub}}$ (Lines 14-16). After completing the extraction of G_{sub}, we select a new starting node and initiate another random walk. This iterative procedure continues until all potential starting nodes, sampled with rate q, have been processed. This module finally outputs the subgraph container $\mathcal{G}_{\text{sub}}$, which serves as a pool for the mini-batch sampling in the training phase.

C. DPGNN Training

With the subgraph container constructed from Module 1, we will now introduce the process of training a GNN model under node-level DP. The details of Module 2, which involves privacy accounting process, will be discussed in Section III-D.

Our objective is to design a probabilistic penalty loss function to optimize GNN parameters, while using DP-SGD to inject noise into gradients for node-level DP. The final GNN model outputs a probability vector for each node, representing its likelihood of being selected into seed set S. We first present the general GNN training formula. Let $G = (V, E)$ be a graph with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ and node feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$. For an r-layer GNN, node u's embedding at the $(i+1)$-th layer $\mathbf{h}_u^{(i+1)}$ is updated by:

$$\mathbf{h}_u^{(i+1)} = \phi\left(\sum\nolimits_{v \in \mathscr{N}(u)} \mathbf{h}_v^{(i)} \mathbf{W}^{(i)}\right), \tag{1}$$

where $\mathbf{W}^{(i)}$ is the learnable weight matrix at the i-th layer, $\mathscr{N}(u)$ is node u's neighbor set, and $\phi(\cdot)$ is a non-linear activation function (e.g., ReLU). This applies to mainstream GNNs like GIN and GAT. For IC model in IM tasks, each edge (u, v) has a weight w_{uv} that represents the influence probability, which is stored in adjacency matrix $\mathbf{A}$. The u-th row, $\mathbf{A}_u$, contains the probabilities of node u being influenced by other nodes. Thus, the probability of node u getting influenced by node v is:

$$A_{uv} = \begin{cases} w_{vu}, & v \in \mathscr{N}(u), \\ 0, & v \notin \mathscr{N}(u). \end{cases} \tag{2}$$

The diffusion process begins with a seed set $S \subseteq V$ and progresses for j steps, where each newly-activated node v influences its neighbor u with probability w_{vu}. Next, using the message-passing mechanism of GNNs, we approximate the diffusion range. We restrict the diffusion step $j \leq r$, since an r-layer GNN captures information up to r-hop neighborhoods. Then we analyze the probability of node u being influenced after j steps by S. Let $\mathbf{H}^{(j-1)} \in \mathbb{R}^{|V| \times d}$ be the node feature matrix after $(j-1)$ steps, with $\mathbf{H}^{(0)} = \mathbf{X}$. The following theorem applies to any j-step diffusion process, providing an upper bound on diffusion probability by leveraging well-known parameters.

Theorem 2. *For a graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$, node feature matrix $\mathbf{H}^{(j-1)}$ and seed set S_{j-1} after $(j-1)$-step diffusion, let $p_j(u|S_{j-1})$ denote the probability that node u is influenced by S_{j-1} under the j-th step in IC model. Using message passing, the upper bound $\hat{p}_j(u|S_{j-1})$ is:*

$$\begin{aligned}\hat{p}_j(u|S_{j-1}) &= \phi(\mathbf{A}_u \cdot \mathbf{H}^{(j-1)}) = \phi\left(\sum\nolimits_{v \in \mathscr{N}(u)} w_{vu} \mathbf{h}_v^{(j-1)}\right) \\ &\geq \phi\left(1 - \prod\nolimits_{v \in \mathscr{N}(u)} \left(1 - w_{vu} \mathbf{h}_v^{(j-1)}\right)\right) = p_j(u|S_{j-1}).\end{aligned} \tag{3}$$

Proof. Please refer to Appendix B for the proof. □

Now we can compute the upper bound of cumulative influence probability $\hat{P}_j(S)$ for all nodes after j steps, which reflects the upper bound of total influence spread. Thus we have:

$$\hat{P}_j(S) = \sum\nolimits_{u \in V} \left(1 - \prod\nolimits_{i=1}^{j} (1 - \hat{p}_i(u|S_{i-1}))\right). \tag{4}$$

Here $\prod_{i=1}^{j} (1 - \hat{p}_i(u|S_{i-1}))$ represents the probability that node u remains inactive within j steps, and minimizing this term for all nodes helps us to maximize the total influence probability $\hat{P}_j(S)$. Thus, we define our loss function $\mathcal{L}(G; \mathbf{W})$ for IM problem as follows:

Algorithm 2: Differentially Private GNNs

```
   Input: Set of subgraphs $\mathcal{G}_{\mathrm{sub}}$, batch size $B$, learning rate $\eta_t$,
          number of iteration $T$, noise scale $\sigma$, clip bound $C$
   Output: The trained model $\mathbf{W}_T$
1  Initialize $\mathbf{W}_0$ randomly
2  for $t = 1 \ldots, T$ do
3      Sample a set of $B$ subgraphs $\mathcal{B}_t \subseteq \mathcal{G}_{\mathrm{sub}}$ uniformly
4      for each subgraph $G_i \in \mathcal{B}_t$ do
5          $\mathbf{g}_t(G_i) \leftarrow \nabla_{\mathbf{W}_t} \mathcal{L}(G_i; \mathbf{W}_t)$
6          $\hat{\mathbf{g}}_t(G_i) \leftarrow \mathbf{g}_t(G_i) / \max(1, \frac{\|\mathbf{g}_t(G_i)\|_2}{C})$
7      $\overline{\mathbf{g}}_t \leftarrow \sum_i \hat{\mathbf{g}}_t(G_i)$
8      $\tilde{\mathbf{g}}_t \leftarrow \overline{\mathbf{g}}_t + \mathcal{N}(0, \sigma^2 \Delta_{\mathbf{g}}^2 \mathbb{I}^d)$   //Add noise
9      $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \frac{\eta_t}{B} \tilde{\mathbf{g}}_t$   //Update parameters
10 return The trained model $\mathbf{W}_T$
```

$$\mathcal{L}(G; \mathbf{W}) \triangleq \sum\nolimits_{u \in V} \prod\nolimits_{i=1}^{j} (1 - \hat{p}_i(u|S_{i-1})) + \lambda \sum\nolimits_{u \in V} \phi(\mathbf{h}_u), \tag{5}$$

where $\hat{p}_i(u|S_{i-1})$ represents the probability of node u being influenced at the i-th step, $\mathbf{W}$ is the learnable parameter in GNN model, and $\phi(\mathbf{h}_u)$ is the probability of node u being selected for seed set. In Eq. 5, the first term minimizes the total probability of nodes not being influenced, and the second term selects nodes with higher importance into the seed set. The parameter $\lambda > 0$ controls the trade-off between two terms. After training, the GNN model outputs the probability of each node being selected for the seed set, and the top-k nodes are chosen as seed nodes.

After defining the loss function, we leverage the gradient perturbation method to achieve node-level DP. Algorithm 2 shows the pseudo-code of GNN training process under node-level DP. Before training, the learnable matrix $\mathbf{W}_0$ is initialized randomly (Line 1). Then for each iteration t, a batch $\mathcal{B}_t$ of subgraphs is uniformly sampled from the subgraph container $\mathcal{G}_{\mathrm{sub}}$ (Lines 2-3). Subsequently, we treat a single subgraph as a per-sample, and clip the per-sample gradient with bounded l_2-norm threshold C (Lines 4-6). After that, Gaussian noise $\mathcal{N}(0, \sigma^2 \Delta_{\mathbf{g}}^2 \mathbb{I}^d)$ is added to the sum of clipped gradient vectors to form the private gradient $\tilde{\mathbf{g}}_t$ (Lines 7-8). Finally, the model is updated with the averaged private gradient (Line 9). This process continues for T iterations and then returns the trained GNN model $\mathbf{W}_T$ as the output.

Overall, with the loss function designed in Eq. 5 and the DPGNN training process outlined in Algorithm 2, we successfully achieve a private GNN model with rigorous node-level DP. In the following subsection, we present a theoretical analysis of DP noise required during GNN training process and provide the privacy guarantee for the entire framework.

D. Privacy Accounting

In this subsection, we establish the privacy guarantee of the PrivIM framework. As aforementioned, this requires deriving the node sensitivity $\Delta_{\mathbf{g}}$, which quantifies the maximum change in output caused by adding or removing a node from G. Before introducing the main privacy result, we first present Lemma 1 to calculate the impact of a single node on the subgraphs extracted by Algorithm 1.

Lemma 1. *Given an original graph $G = (V, E)$, a fixed subgraph size n, the maximum node in-degree θ, and an r-layer GNN model. Algorithm 1 ensures that the maximum occurrence of any node across all subgraphs is bounded by N_g, where:*

$$N_g = \sum\nolimits_{i=0}^{r} \theta^i = \frac{\theta^{r+1} - 1}{\theta - 1}. \tag{6}$$

Proof. Please refer to Appendix C for the proof. □

Lemma 1 provides the upper bound on the number of times any node can appear across the subgraphs extracted by Algorithm 1, which grows exponentially with the GNN layers r. This result is crucial for calculating the node-level sensitivity $\Delta_{\mathbf{g}}$, which determines the amount of noise to be added during the gradient updates. Subsequently, we analyze the node-level sensitivity $\Delta_{\mathbf{g}}$ in Lemma 2, and provide the formal privacy guarantee in Theorem 3.

Lemma 2. *Given two adjacent graphs $G, G' \subset \mathcal{G}$ that differ by at most one node, a fixed subgraph size n, the maximum node in-degree θ and the clipping bound C. Let the aggregated batched gradients for two adjacent graphs be $\overline{\mathbf{g}}_t$ and $\overline{\mathbf{g}'}_t$, respectively. Then the node-level sensitivity $\Delta_{\mathbf{g}}$ satisfies the following inequality:*

$$\Delta_{\mathbf{g}} = \|\overline{\mathbf{g}}_t - \overline{\mathbf{g}'}_t\|_2 \leq C \cdot N_g. \tag{7}$$

Proof. Please refer to Appendix D for the proof. □

With the bounded node-level sensitivity provided in Lemma 2, we can calibrate the amount of noise added into the gradients. We have the following privacy accounting result.

Theorem 3. *Let $\mathcal{G}_{\text{sub}}$ be the subgraph container with $|\mathcal{G}_{\text{sub}}| = m$, B be the batch size, N_g be the upper bound of node occurrences. Over T iterations, Algorithm 2 satisfies node-level $(\alpha, \gamma T)$-Rényi DP, where*

$$\gamma = \frac{1}{\alpha - 1} \log \left[\sum_{i=0}^{N_g} \binom{B}{i} \frac{N_g^i (m - N_g)^{B-i}}{m^B} \exp\left(\frac{(\alpha^2 - \alpha) i^2}{2 N_g^2 \sigma^2} \right) \right]. \tag{8}$$

Proof. Please refer to the Appendix E for the proof. Notably, applying Theorem 1 to the above result leads to the (ϵ, δ)-DP guarantee. □

E. Observations

Tailored for the IM task, the PrivIM framework effectively provides node-level DP protection for graph data during GNN training. Nevertheless, it still exhibits limitations that prevent it from achieving a better balance between privacy and utility.

- While reducing the noise sensitivity, projecting the original graph G to a θ-bounded graph G^θ disrupts essential structural information. In the IM task, as each node's embedding depends on both the neighborhood information and the global information, a sparse structure significantly weakens the model's generalization capability and hinders its performance.
- When training a GNN, the required noise scale increases exponentially with the number of GNN layers r. Consequently, the above implementation necessitates injecting substantial DP noise during training, resulting in a deterioration of model utility.

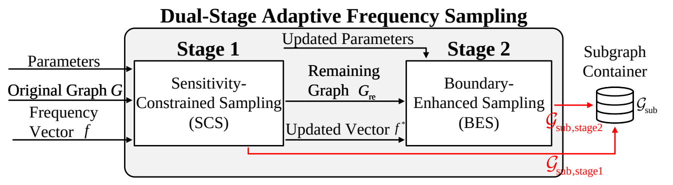

Dual-Stage Adaptive Frequency Sampling
Parameters
Original Graph G
Frequency Vector f
Stage 1
Sensitivity-Constrained Sampling (SCS)
Updated Parameters
Remaining Graph G_{re}
Updated Vector f^*
Stage 2
Boundary-Enhanced Sampling (BES)
Subgraph Container
$\mathcal{G}_{\text{sub}}$
$\mathcal{G}_{\text{sub,stage2}}$
$\mathcal{G}_{\text{sub,stage1}}$

Fig. 3: An overview of dual-stage frequency sampling scheme

Although sub-optimal, the above implementation offers two key insights that motivate us to design a more effective subgraph sampling approach. This approach should aim to preserve as much structural information as possible while constraining the upper bound of gradient sensitivity to minimize excessive noise. Following these two insights, we introduce a dual-stage adaptive frequency sampling scheme within our PrivIM framework to enhance performance.

IV. Dual-Stage Frequency Sampling Scheme

In this section, a dual-stage adaptive frequency sampling scheme is proposed to address the above limitations, optimizing the subgraph extraction process within our PrivIM framework. An overview of this scheme is shown in Figure 3. In particular, we first design a Sensitivity-Constrained Sampling (SCS) method to minimize the influence of individual nodes on each other while preserving the overall graph structure (Section IV-A). We then devise a Boundary-Enhanced Sampling (BES) approach to capture boundary structural information, enhancing the model's utility without further consuming any privacy budget (Section IV-B). An indicator is then developed for optimal parameter selection (Section IV-C). Finally, we establish theoretical analysis of our PrivIM framework with the proposed sampling method (Section IV-D).

A. Sensitivity-Constrained Sampling (SCS)

In this subsection, we present our Sensitivity-Constrained Sampling (SCS) algorithm, i.e., Stage 1 of our dual-stage adaptive frequency sampling. The high-level idea of SCS is that if one node has higher frequency value, its sampling probability will be reduced to balance its existing impact, leading to lower DP noise. We provide a detailed explanation of this sampling method, described as a function **FreqSampling** in Algorithm 3 (Lines 9-28).

Given an original graph $G = (V, E)$, we first initialize the frequency vector f with all zeros (Line 1). Note that compared to the naive sampling in Section III-B, there is no assumption of projection in this setting. Then, we proceed with subgraph extraction by RWR sampling method (Line 10-28). Unlike Algorithm 1, at each step l, we leverage the frequency vector f to dynamically adjust the sampling probability d_v for each node v as follows:

$$d_v = \frac{e_v}{\sum_{u \in \mathcal{N}(v_{\text{cur}})} e_u}, \qquad e_v = \begin{cases} \dfrac{1}{(f_v + 1)^\mu}, & \text{if } f_v < M, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

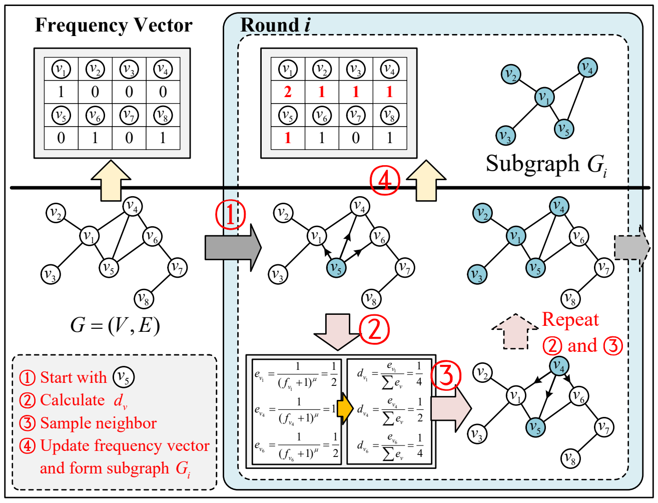
where the frequency value f_v represents the occurrences of node v being sampled, and μ represents the decay factor, which

Algorithm 3: Dual-Stage Frequency Sampling Scheme

```
Input: Graph $G = (V, E)$, subgraph size $n$, return
       probability of RWR $\tau$, decay factor $\mu$, sampling rate
       $q$, positive integer $s$, random walk length $L$,
       frequency threshold $M$
Output: Set of subgraphs $\mathcal{G}_{\text{sub}}$
   // Sensitivity-Constrained Sampling
1  Initialize the frequency vector $f \in \mathbb{R}^{|V| \times 1}$ with all zeros.
2  $\mathcal{G}_{\text{sub,stage1}} \leftarrow$ FreqSampling($f, G, n$)
   // Boundary-Enhanced Sampling
3  $V_{\text{re}} \leftarrow V \setminus \{\text{for } v \in V \text{ if } f_v = M\}$
4  Update remaining edges $E_{\text{re}}$ and frequency vector $f^*$
5  Build $G_{\text{re}} = (V_{\text{re}}, E_{\text{re}})$
6  $\mathcal{G}_{\text{sub,stage2}} \leftarrow$ FreqSampling($f^*, G_{\text{re}}, \frac{n}{s}$)
7  $\mathcal{G}_{\text{sub}} \leftarrow \mathcal{G}_{\text{sub,stage1}} + \mathcal{G}_{\text{sub,stage2}}$
8  return Set of subgraphs $\mathcal{G}_{\text{sub}}$
9  Function FreqSampling($f$, $G$, $n$):
10     for $v \in V$ do
11         $p \leftarrow$ Uniform(0, 1)
12         if $p < q$ and $f_v < M$ then
13             $V_{\text{sub}} \leftarrow \emptyset, v_0 \leftarrow v, v_{\text{cur}} \leftarrow v$
14             $V_{\text{sub}}.add(v)$
15             for $l = 1, \ldots, L$ do
16                 $p \leftarrow$ Uniform(0, 1)
17                 if $p < \tau$ then
18                     $v_{\text{cur}} \leftarrow v_0$
19                 $v_{\text{next}} \leftarrow$ Sample neighbors with Eq. 9.
20                 $v_{\text{cur}} \leftarrow v_{\text{next}}$
21                 if $v_{\text{next}}$ is not in $V_{\text{sub}}$ then
22                     $V_{\text{sub}}.add(v_{\text{next}})$
23                 if $|V_{\text{sub}}| = n$ then
24                     Form $G_{\text{sub}}$ with nodes in $V_{\text{sub}}$.
25                     $\mathcal{G}_{\text{sub}}.add(G_{\text{sub}})$
26                     Update $f \in \mathbb{R}^{|V| \times 1}$ with $V_{\text{sub}}$.
27                     break
28     return Set of subgraphs $\mathcal{G}_{\text{sub}}$
```

controls the impact of f_v on the sampling probability d_v. Eq. 9 ensures that the sampling probability of each neighbor is inverse-related to f_v, effectively reducing the likelihood of frequently occurring nodes being sampled repeatedly. Besides, this approach ensures that the sampling process captures information from diverse regions of the original graph G, preventing it from being confined to areas around nodes with high in-degrees. In Eq. 9, we set a threshold M for the frequency value f_v, ensuring that each node appears in subgraphs no more than M times. As a result, the maximum occurrence of any node is $N_g^* = M < N_g$. After obtaining each subgraph, we update f (Line 26) and repeat the above process to build the subgraph container $\mathcal{G}_{\text{sub,stage1}}$. This iterative procedure continues until all potential starting nodes, selected based on the sampling rate q, have been processed. In Figure 4, we present a toy example illustrating the critical steps of our adaptive frequency sampling method.

Example 3. *Before sampling, we construct the frequency vector for all nodes in graph $G = (V, E)$. For round i, we initiate our frequency sampling process from v_5 (Step 1). Then we calculate the sampling probabilities for neighbors of*



Frequency Vector
Round i
v_1 v_2 v_3 v_4
1 0 0 0
v_5 v_6 v_7 v_8
0 1 0 1
v_1 v_2 v_3 v_4
2 1 1 1
v_5 v_6 v_7 v_8
1 1 0 1
Subgraph G_i
$G = (V, E)$
① ② ③ ④
Repeat ② and ③
① Start with v_5
② Calculate d_v
③ Sample neighbor
④ Update frequency vector and form subgraph G_i
$e_{v_1} = \frac{1}{(f_{v_1}+1)^\mu} = \frac{1}{2}$
$e_{v_4} = \frac{1}{(f_{v_4}+1)^\mu} = 1$
$e_{v_6} = \frac{1}{(f_{v_6}+1)^\mu} = \frac{1}{2}$
$d_{v_1} = \frac{e_{v_1}}{\sum e_v} = \frac{1}{4}$
$d_{v_4} = \frac{e_{v_4}}{\sum e_v} = \frac{1}{2}$
$d_{v_6} = \frac{e_{v_6}}{\sum e_v} = \frac{1}{4}$

Fig. 4: An illustrative example of the key steps in the frequency sampling method

v_5 by Eq. 9 (Step 2). Here we find that node v_4's sampling probability $d_{v_4} = \frac{1}{2}$ is the highest, thereby we choose it as the next sampled node (Step 3). We iterate the process in Step 2 and 3 until we achieve n unique nodes to construct the subgraph G_i, here $n = 5$. Finally, the frequency vector f is updated (Step 4), and we move to next round $(i + 1)$ until all potential starting nodes have been processed.

Overall, our SCS algorithm leverages the adaptive frequency sampling method for constraining the impact of sampled nodes, preventing excessive noise.

B. Boundary-Enhanced Sampling (BES)

While our SCS algorithm benefits from the global threshold M and the frequency vector f to reduce the sensitivity of the gradient, we observe that after our first sampling stage, certain boundary regions of the original graph still contain nodes that do not reach the threshold. These boundary regions, typically consisting of clusters with fewer nodes than the sampling size n, still encapsulate significant structural information from the original graph. Hence, a natural question arises: *Is it possible to leverage these regions to improve model performance without consuming privacy budget?* To answer this question, we propose our Boundary-Enhanced Sampling (BES) algorithm, which mainly focuses on supplementing the structural information from the boundary areas.

Lines 3-6 of Algorithm 3 explain the procedure of BES. The core idea is that, after the first stage, we remove nodes whose frequency value reaches the threshold, i.e., $f_v = M$, and achieve the node set V_{re} (Line 3). Then we update the edge list E_{re} and set the vector f^* to focus solely on nodes that have not reached M (Line 4). We construct the new graph $G_{\text{re}} = (V_{\text{re}}, E_{\text{re}})$ (Line 5), and the subgraph size is reduced to $\frac{n}{s}$ for boundary areas. These steps allow us to better utilize the remaining regions, thereby enhancing the model's performance. Furthermore, to avoid increasing the privacy budget, we leverage the adaptive frequency sampling method again with different parameters to build the subgraph container $\mathcal{G}_{\text{sub,stage2}}$ (Line 6). Finally, we combine $\mathcal{G}_{\text{sub,stage1}}$ and $\mathcal{G}_{\text{sub,stage2}}$ together to construct the overall $\mathcal{G}_{\text{sub}}$ (Line 7). Note that the goals of using adaptive frequency sampling method in two stages are different. For the first stage, it is used to reduce the overall DP noise, while the second stage focuses on supplementing graph information without consuming additional privacy budget.

Discussions. Our dual-stage adaptive frequency sampling method has the following improvements:

- For the first stage, our goal is to limit the occurrences of each node in the subgraphs. Therefore, we set the sampling ratio of each neighbor inverse-related to the current frequency value f_v during the sampling process. Besides, by introducing a global threshold M on f, we can constrain the impact of sampled nodes, preventing excessive noise.
- For the second stage, our goal is to supplement the structural information without increasing privacy budget. To this end, we fix the global threshold and reduce the subgraph size for additional structural information. The gain of model's utility from this stage is evaluated in Section V-B.
- Unlike the naive random sampling, our adaptive frequency sampling avoids projecting the original graph into a θ-bounded graph, which ensures that the topological features of the sampled training subgraphs remain as consistent as possible with the original graph, thereby enhancing the generalization capability of the GNN model.

C. Parameter Selection

For our dual-stage adaptive frequency sampling strategy, a remaining problem is to select optimal parameters to ensure the effectiveness of the trained model, including the subgraph size n and the global threshold M. Specifically, given a fixed n, a smaller M may lead to fewer subgraphs, which limits the model's generalization capability. Conversely, a larger M may introduce excessive noise, resulting in worse utility. Similarly, for a fixed M, a smaller n means sparser subgraph structures, while a larger n reduces the number of generated subgraphs, both of which negatively affect the model's utility.

Based on the above analysis, it is intuitive that the model's utility initially increases and then declines as n and M grow. To formalize and capture this trend, we design an indicator for parameter selection, which serves two key purposes: *i)* modeling the trends of performance w.r.t. n and M; *ii)* identifying the optimal values of n and M for different datasets. For the first purpose, we leverage the probability density function of Gamma distribution. Specifically, given any n, M, the indicator $I(n, M)$ is defined as:

$$I(n,M) = \frac{\xi(n;\beta_n,\psi_n) + \xi(M;\beta_M,\psi_M)}{\max\limits_{n,M}\left(\xi(n;\beta_n,\psi_n) + \xi(M;\beta_M,\psi_M)\right)}, \quad (10)$$

$$\xi(n;\beta_n,\psi_n) = \frac{n^{\beta_n - 1} e^{-\frac{n}{\psi_n}}}{\psi_n^{\beta_n}\Gamma(\beta_n)}, \quad \xi(M;\beta_M,\psi_M) = \frac{M^{\beta_M - 1} e^{-\frac{M}{\psi_M}}}{\psi_M^{\beta_M}\Gamma(\beta_M)}, \quad (11)$$

where ψ_n, ψ_M represent the scale parameters and β_n, β_M represent the shape parameters. To achieve the second purpose, we need to associate these parameters with the size of different datasets. Intuitively, for larger datasets, the number of sampled subgraphs increases, allowing for larger n and smaller M. Conversely, for smaller datasets, fewer subgraphs are sampled, leading to smaller n and larger M. Notably, the variation in dataset size $|V|$ is significantly larger than that of n and M. This trend is further supported by the prior experiments in Section V-C. Therefore, we establish the relationship between β_n, β_M and the dataset size $|V|$ by:

$$\beta_n = k_n \ln(|V|) + b_n, \quad \beta_M = \frac{k_M}{\ln(|V|)} + b_M. \quad (12)$$

We provide the method for determining the parameters $\psi_n, \psi_M, k_n, k_M, b_n, b_M$ of Eq. 10, 11, 12 in the Appendix H. Due to the relatively small value space for both parameters, we combine grid search with our indicator to adaptively identify the optimal values for different datasets, without the need to run the entire algorithm, which would be computationally expensive. The effectiveness of this indicator is further evaluated in Section V-D.

D. Theoretical Analysis

In this subsection, we further analyze the privacy guarantee and time complexity of our PrivIM framework, particularly when integrating the enhanced dual-stage sampling scheme into PrivIM (denoted as PrivIM*).

Privacy Analysis. As our dual-stage adaptive frequency sampling scheme reduces the maximum occurrences of nodes across all subgraphs from N_g to N_g^*, we can apply Theorem 3 to show that PrivIM* satisfies node-level $(\alpha, \gamma^* T)$-Rényi DP, where γ^* is computed by substituting N_g with N_g^* in Eq. 23.

Time Complexity Analysis. We divide the training process into two phases, namely preprocessing and per-epoch training, and analyze the time complexity of each phase in PrivIM and PrivIM*.

- **Preprocessing.** For PrivIM, this phase includes both projection and subgraph extraction, whereas PrivIM* only involves subgraph extraction. We first analyze the projection part in PrivIM: for each node v in V, up to θ neighbors are sampled in $O(\theta)$ time. Hence, the total loop's time complexity is $O(|V|\theta)$. As for the subgraph extraction part in PrivIM, according to Algorithm 1, the expected running time for the outer loop is $O(q|V^\theta|)$, and that for the inner loop is essentially the length of random walk L. Therefore, the overall complexity for Algorithm 1 is $O(q|V^\theta|L)$. For the first stage in Algorithm 3, the time complexity is $O(q|V|L)$. For the second stage in Algorithm 3, the time complexity is $O(q(|V| - |V_M|)L)$, where V_M represents the nodes that have reached the frequency threshold M after the first stage.
- **Training iterations.** We assume the GNN model's complexity for processing a single subgraph is $O(h)$, where h represents the computational cost of the GNN model. Thus the main training loop contributes $O(Tmh)$, where T is the number of iterations, and m represents the number of subgraphs.

Therefore, the overall time complexity of PrivIM and PrivIM* can be approximated as $O(|V|\theta + q|V^\theta|L + Tmh)$ and $O(q(2|V| - |V_M|)L + Tmh)$, respectively. Compared with

TABLE I: Statistics of the experimented datasets (K = 10^3, M = 10^6, B = 10^9)

Dataset	$\|V\|$	$\|E\|$	Type	Avg. Degree
Email	1K	25.6K	Directed	25.44
Bitcoin	5.9K	35.6K	Directed	6.05
LastFM	7.6K	27.8K	Undirected	7.29
HepPh	12K	118.5K	Undirected	19.74
Facebook	22.5K	171K	Undirected	15.22
Gowalla	196K	950.3K	Undirected	9.67
Friendster	65.6M	1.8B	Undirected	55.06

PrivIM, PrivIM* reduces the cost in projection phase, which is typically high on extremely large and dense graphs. We will validate these analytical results in Section V-F.

V. Evaluation

In this section, we conduct extensive experiments to empirically evaluate the effectiveness and efficiency of our proposed methods.

A. Experimental Setting

Datasets. We evaluate our methods on six widely-used real-world directed or undirected graph datasets with varying sizes and edge density, including the email network (Email [34]), the social networks (Bitcoin [35], LastFM [36], Facebook [37], Gowalla [30]), and the citation network (HepPh [38]). We also evaluate our methods on large-scale Friendster [39] dataset, which contains billions of edges, to demonstrate the scalability of our methods on very large networks. Due to the hardware memory limitations, we partition Friendster into multiple graphs during both training and evaluation phases, which facilitates efficient processing while mitigating the risk of memory overflow. Table I summarizes their statistics. For more comprehensive data descriptions, please refer to Appendix L.

Competitors. For comprehensive evaluation of model effectiveness, we compare our framework PrivIM with five baselines as follows.

- **PrivIM**: It is our framework with the naive implementation presented in Section III.
- **PrivIM***: It is our framework with the dual-stage adaptive frequency sampling scheme presented in Section IV.
- **EGN** [40]: It is the foundational unsupervised learning framework that offers a principled way to solve IM problems. To ensure the same level of DP guarantee, we employ DP-SGD during its model training phase.
- **HP** [16]: It offers a solution that specifically addresses node-level privacy. In this setting, we apply its HeterPoisson algorithm and Symmetric Multivariate Laplace (SML) noise to IM tasks and maintain the same level of DP guarantee as our framework.
- **HP-GRAT**: It is a variant of HP [16], while training with the same GNN model (i.e., GRAT [41]) as our framework.
- **Non-Private**: It is our non-private algorithm, i.e., PrivIM* with $\epsilon = \infty$.
- **CELF** [3]: It is the greedy algorithm leveraging the submodularity of coverage functions, which guarantees the worst-case approximation ratio of $(1 - \frac{1}{e})$. Hence, it is regarded as the ground truth.

Note that another existing work GAP [17], which combines GNN and DP, relies on DP-SGD for training the classification module and leverages aggregation perturbation method to train the aggregation module. Given its inherent designed for node classification tasks, it is not suitable for comparison in our experiments on IM problems.

Evaluation Metrics. To evaluate our framework quantitatively, we leverage the following performance metrics.

- **Effectiveness Metrics**: Consistent with prior works [8], [40], [42], we evaluate the performance of our PrivIM and PrivIM* by quantifying *Influence Spread*, i.e., the number of final infected nodes. Note that since CELF provides a theoretical $(1 - \frac{1}{e})$ lower bound guarantee, we introduce an additional metric *Coverage Ratio*, which is computed by $\frac{|V_{\text{method}}|}{|V_{\text{CELF}}|}$, where $|V_{\text{method}}|$ and $|V_{\text{CELF}}|$ represent the influence spread of the compared methods and CELF, respectively.
- **Efficiency Metrics**: We evaluate the efficiency of our methods by computing the time cost of two phases, namely preprocessing and per-epoch training.

Parameter Settings. Given an original graph, we split the training and testing nodes randomly by $(50\%, 50\%)$. We leverage IC diffusion model for the evaluation and set influence probability $w_{vu} = 1$, step $j = 1$, and seed size $k = 50$. For privacy parameters, we set $\delta < \frac{1}{|V_{\text{train}}|}$ and vary ϵ from 1 to 6. We set the sampling rate $q = \frac{256}{|V_{\text{train}}|}$, random walk length $L = 200$, maximum node in-degree $\theta = 10$ and return probability $\tau = 0.3$. All the parameters are trained with learning rate $\eta_t = 0.005$. Since the model training is a randomized procedure, we repeat all the experiments 5 times and report the mean value. We leverage a three-layer GRAT structure, an advanced variant of traditional GAT structure, for our methods PrivIM and PrivIM*, with 32 hidden units per layer. The same structure also applies to HP-GRAT. For HP and EGN methods, we leverage a three-layer GCN structure with 32 hidden units per layer. We also evaluate our proposed PrivIM* with four classical GNN models apart from the default GRAT model. We provide all the GNN models we used in Appendix G.

All the models are implemented in PyTorch using PyTorch Geometric (PyG). Experiments are conducted on a device with one NVIDIA GeForce RTX 4070 Ti GPU, an Intel Core i7-14700K CPU, and 64GB RAM.

B. Overall Results

We first validate the effectiveness of our framework PrivIM by conducting the performance evaluation over six datasets and comparing them with the other competitors. The overall results of our evaluation are presented in Figure 5, while the results of HepPh can be found in Appendix J due to space constraints. Additionally, we also evaluate the performance of all competitors on Friendster in Figure 5.

The non-private version of our algorithm achieves very similar influence spread to the ground truth results obtained by

TABLE II: Coverage ratio of the methods over six datasets. The best and second best performance are highlighted in bold and underlined separately.

Methods	ϵ	Email	Bitcoin	LastFM	HepPh	Facebook	Gowalla
Non-Private	∞	98.09 ± 1.69	98.99 ± 0.46	97.93 ± 0.97	98.42 ± 0.31	97.50 ± 0.34	99.78 ± 0.03
PrivIM	4	76.08 ± 0.16	73.48 ± 0.45	79.39 ± 1.62	73.54 ± 2.01	82.60 ± 3.88	82.26 ± 0.34
PrivIM+SCS	4	**95.08 ± 0.75**	<u>92.24 ± 1.15</u>	<u>89.56 ± 1.78</u>	<u>93.91 ± 1.67</u>	<u>90.92 ± 1.52</u>	<u>95.47 ± 0.88</u>
PrivIM+SCS+BES (i.e., PrivIM*)	4	<u>94.44 ± 1.32</u>	**92.84 ± 1.40**	**93.76 ± 0.73**	**94.50 ± 0.93**	**95.55 ± 0.23**	**98.02 ± 0.05**
PrivIM	1	62.01 ± 0.37	60.13 ± 1.07	31.72 ± 0.88	32.24 ± 2.32	70.20 ± 3.22	68.05 ± 0.98
PrivIM+SCS	1	<u>83.05 ± 1.83</u>	<u>85.76 ± 3.61</u>	<u>80.72 ± 3.84</u>	<u>80.98 ± 1.78</u>	<u>82.68 ± 4.37</u>	<u>81.28 ± 1.40</u>
PrivIM+SCS+BES (i.e., PrivIM*)	1	**83.87 ± 2.25**	**86.42 ± 0.88**	**83.50 ± 1.31**	**85.49 ± 0.30**	**85.10 ± 1.15**	**86.57 ± 0.34**

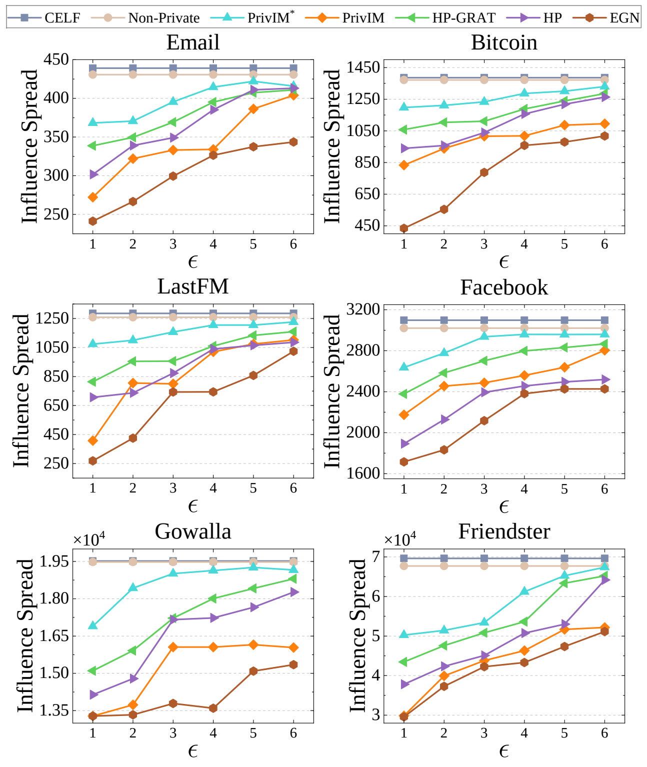


Fig. 5: Influence spread of all the methods over six datasets, by varying privacy budget ϵ

CELF. Under DP guarantee, our method PrivIM* consistently outperforms the other methods in all cases, especially when given a smaller privacy budget. The gap between PrivIM* and PrivIM indicates the effectiveness of our dual-stage adaptive frequency sampling scheme. On the other hand, HP-GRAT performs better than HP. This can be attributed to the superiority of GRAT model, which has also been deployed in our framework. While the methods HP and HP-GRAT attempt to bound the maximum in-degree θ and number of hops r to reduce the DP noise, they are specifically for node-level tasks and focus solely on single node for each subgraph. Therefore, these methods disrupt the intricate structural information and lead to overly simplified connectivity, resulting in poorer utility. EGN achieves the lowest influence spread in all cases, and this is because it randomly samples the subgraphs for training, which introduces excessive DP noise, leading to the worst performance. For large-scale network like Friendster, as shown in Figure 5, our method PrivIM* consistently outperforms the other methods, especially when given a smaller privacy budget, which also shows our method's scalability on large-scale datasets.

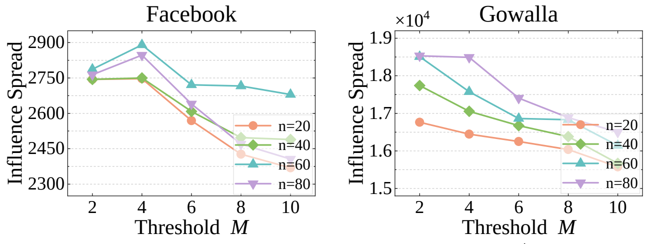


Fig. 6: Impact of threshold M on PrivIM* ($\epsilon = 3$)

To further study the impact of each stage (i.e., SCS and BES) of our dual-stage adaptive sampling scheme, Table II shows the coverage ratio of our different implementations for PrivIM framework, including "PrivIM+SCS" which means only Stage 1 (SCS) is integrated. Note that "PrivIM+SCS+BES" means both stages are integrated (i.e., PrivIM*). The gap between PrivIM and PrivIM+SCS shows the advantage of SCS, while the gap between PrivIM+SCS and PrivIM* highlights the contribution of BES. We find that both algorithms improve the utility of the GNN model, with SCS providing a larger boost. This is because SCS reduces overall DP noise, while BES focuses on supplementing graph information.

C. Parameter Studies

This subsection investigates the impact of different parameters on the performance of our proposed PrivIM*. We summarize the results with the following observations:

- **Threshold** M: As the threshold M relates to the number of nodes $|V|$ in a graph, in this study, it varies from $\{4, 6, 8, 10, 12\}$ for Email dataset that has only 1K nodes, while for the other five datasets with more nodes, M is selected from $\{2, 4, 6, 8, 10\}$. The privacy budget is fixed at $\epsilon = 3$, and the subgraph size n varies from $\{20, 40, 60, 80\}$. Figure 6 shows the results on Facebook and Gowalla, while other results are in Appendix F. Overall, we can observe that PrivIM* achieves its peak performance at smaller M. This is due to the trade-off between having more subgraphs for generalization and the amount of injected noise: the larger M is, the more subgraphs are sampled for training, but on the other hand, it will lead to more noise injection.
- **Subgraph Size** n: We plot the performance curve of our method PrivIM* by varying n from 10 to 80. Figure 7

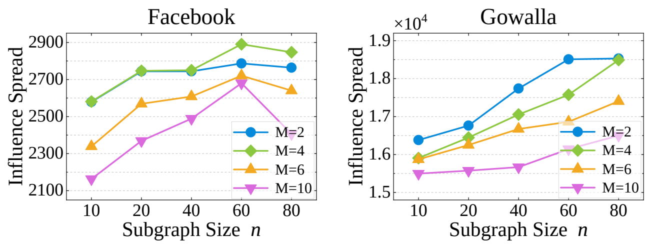


Fig. 7: Impact of subgraph size n on PrivIM* ($\epsilon = 3$)

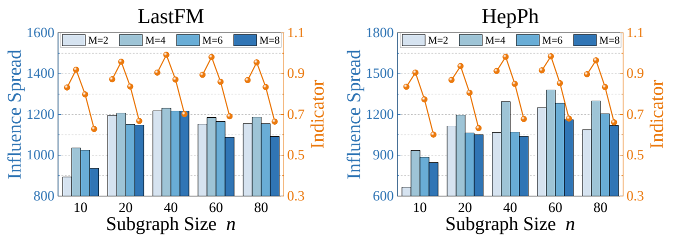


Fig. 8: Theoretical values of the indicator and empirical results of PrivIM* ($\epsilon = 3$)

shows that the influence spread keeps growing with n on Gowalla, while on other datasets it increases with n up to a peak point, and drops afterwards. This is attributed to the fact that on the one hand, n represents the amount of information contained by a single subgraph. A smaller n may result in less information per subgraph, which is insufficient for better performance. On the other hand, a higher n may reduce the number of subgraphs, adversely affecting the model's generalization ability. More results can be found in Appendix F.

D. *Indicator Effectiveness*

As shown in our parameter studies, the performance of PrivIM* exhibits an initially increasing and then decreasing trend as M or n increases. We also observe that the optimal values for M and n depend on the dataset size $|V|$. Leveraging these observations, we design an indicator based on the probability density function of Gamma distribution to capture the optimal M and n for each dataset in Section IV-C. We set all the parameters based on the prior experiments in the Appendix H. To evaluate the effectiveness of our designed indicator, we set the scale parameters $\psi_M = 5$ and $\psi_n = 25$. For all the datasets, we set $k_M = 4.02$, $b_M = 1.22$ and $k_n = 0.47$, $b_n = -1.03$, respectively.

We report the values of our designed indicator and the experimental results in Figure 8, with a privacy budget $\epsilon = 3$. More results could be found in Appendix F. Bars show the empirical results of influence spread by PrivIM*, while curves present the theoretical values returned by our indicator. We observe that, for a fixed n, our indicator accurately captures the trend of influence spread as M varies, and the peak of the indicator corresponds to the optimal value of M. For example, on LastFM dataset when $n = 60$, the influence spread keeps growing with M up to a peak point, and drops afterwards. Notably, our indicator follows the same trend. Furthermore, both the influence spread and our indicator share the same peak point, occurring at $M = 4$. Similarly, when M is fixed,

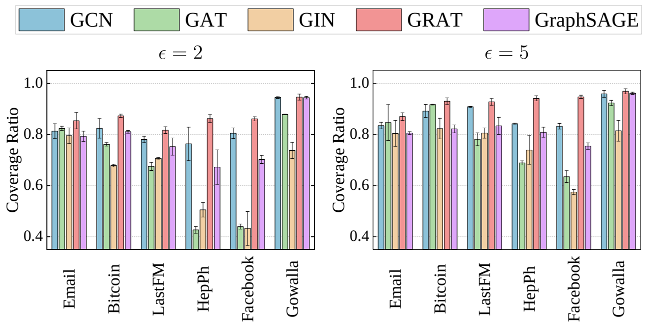


Fig. 9: Impact of different GNN models on PrivIM*

our indicator also reflects how the performance changes with varying n, and the peak of our indicator aligns with the optimal value of n. We also evaluate our designed indicator with different privacy budgets ($\epsilon = 1$ and $\epsilon = 6$) on LastFM, and report the values in the Appendix K due to space limitations.

The above observations provide compelling evidence for the effectiveness of our designed indicator. We attribute the correctness to the following two key factors. First of all, the trend of Gamma distribution's probability density function closely aligns with the correlation between influence spread and the parameters M and n, intuitively suggesting a fit between them. On the other hand, the shape parameters β_M, β_n in Gamma distribution determine the position of the peak. Through our analysis in Section IV-C, we establish a connection between β_M, β_n and $\ln(|V|)$, enabling our indicator to adapt to varying dataset sizes. Overall, our designed indicator can be leveraged to determine the parameters for different datasets and save the privacy budget for expensive parameter searching.

E. *GNN Models*

In this subsection, we investigate the performance of our proposed PrivIM* with four classical GNN models apart from the default GRAT model, namely GraphSAGE [43], GCN [13], GAT [44], and GIN [14]. Specifically, we leverage the coverage ratio as the effectiveness metric and validate the performance of four GNN models over six datasets under the privacy budget $\epsilon = 2$ and $\epsilon = 5$, respectively. The results are shown in Figure 9, and we have the following major observations: *i)* Overall, the results show a marginally better performance of GRAT compared to others. We attribute this performance improvement to its attention mechanism designed at the source nodes instead of the target nodes, leading to a reduction in the reward for overlapping coverage. *ii)* Compared to GCN, GIN obtains less stable performance across different datasets. GIN leverages a Multi-Layer Perceptron (MLP) to enhance the non-linearity of aggregation. However, this complex aggregation mechanism may lead to overfitting of the graph structures in IM tasks. *iii)* GCN shows a better performance than GraphSAGE. This is because the random sampling strategy in GraphSAGE cannot effectively capture the underlying dependencies between nodes, leading to sub-optimal performance. In summary, our PrivIM* can be implemented with different GNN models. However, for IM tasks, we recommend leveraging GNN models that can identify nodes with highly overlapping coverage for better performance.

TABLE III: Computational time cost (in second)

Method	Phase	Email	Bitcoin	LastFM	HepPh	Facebook	Gowalla
PrivIM*	Preprocessing	0.77s	7.35s	15.09s	22.97s	6.02s	8.12s
	Per-epoch Training	1.57s	2.18s	4.28s	6.13s	10.78s	9.72s
PrivIM	Preprocessing	0.77s	1.29s	2.03s	2.91s	4.46s	18.48s
	Per-epoch Training	4.86s	5.29s	5.92s	8.32s	6.79s	7.18s
HP-GRAT	Preprocessing	1.47s	5.71s	4.65s	7.71s	5.25s	5.14s
	Per-epoch Training	5.06s	6.42s	8.61s	9.24s	9.56s	7.07s
ECN	Preprocessing	0.94s	2.49s	3.22s	5.98s	3.99s	3.56s
	Per-epoch Training	5.91s	7.96s	7.69s	10.99s	10.31s	8.29s

F. Time Consumption

Besides the effectiveness evaluation, another key evaluation metric of our methods is the time consumption. As shown in Table III, we validate the time cost of PrivIM*, PrivIM, HP-GRAT and ECN, respectively. Note that we divide the time consumption into two phases, namely preprocessing and per-epoch training. For PrivIM, the preprocessing phase includes both projection and subgraph extraction, whereas PrivIM* only involves subgraph extraction. Overall, PrivIM exhibits a lower time cost for preprocessing compared to PrivIM*, while the training time remains consistently a bit higher. This is because PrivIM imposes no constraints on the sampling process and obtains more subgraphs for training. In contrast, PrivIM* performs continuous updates to the frequency vector f and applies constraints to the original graph during sampling, leading to higher preprocessing time. As the size of the original graph increases, the number of training subgraphs also increases, resulting in higher training time for PrivIM*. The training time of HP-GRAT and ECN is longer than our proposed methods, which is due to their lack of restrictions on the sampling process, leading to more subgraphs. Overall, our methods are computationally competitive, balancing performance and time efficiency effectively.

VI. Related Work

In this section, we discuss the research work relevant to our study from three aspects: *i)* Influence Maximization; *ii)* Learning-based Methods for Combinatorial Optimization; *iii)* Differentially Private Graph Neural Networks.

A. Influence Maximization

Influence Maximization (IM) was first formulated as a classical Combinatorial Optimization (CO) problem in 2003 [4]. Traditional IM methods fall into three categories: simulation-based [3], proxy-based [5], and sampling-based [28]. Among these, sampling-based methods achieve a balance between effectiveness and efficiency [9], offering strict approximation bounds with low time complexity. Despite the strengths, traditional methods struggle with scalability on large networks.

B. Learning-based Methods for Combinatorial Optimization

Learning-based methods, including supervised learning [45], reinforcement learning [46], and unsupervised learning [47], address the limitations of traditional methods. Supervised learning is hindered by high labeling costs and poor generalization, while reinforcement learning suffers from instability and limited exploration. In contrast, unsupervised learning has emerged as the most promising approach for CO problems. Karaias et al. [40] pioneered a GNNs framework via a probabilistic penalty loss function, and Wang et al. [42] proposed an end-to-end model with cardinality constraints into its architecture. This paper designs a differentiable loss function to enable our privacy-preserving mechanism.

C. Differentially Private Graph Neural Networks

Existing efforts on DP in GNNs focus on three noise injection methods: *i) input perturbation* [48]; *ii) aggregation function perturbation* [17]; *iii) gradient perturbation* [49]. This work mainly focuses on the third category. Existing research on GNNs with DP primarily considers two DP definitions: edge-level DP and node-level DP. Yang et al. [50] employed DP-SGD for link privacy, while Sajadmanesh et al. [17] combined aggregation and gradient perturbation for node-level privacy. In our work, we implement node-level privacy protection for IM tasks with DP-SGD. Existing studies primarily focus on two strategies to address the data dependency issue and bound the DP noise scale during training: *i) restricting the number of each node's neighbors [15], [16]; ii) limiting the depth of GNNs [17].* However, both approaches are typically applicable to node-level or edge-level tasks. As a graph-level task, IM necessitates new strategies to safeguard structural information while mitigating DP noise. Note that since IM problem is mathematically a classical combinatorial optimization problem [4], our framework can be easily extended to other problems like maximum coverage and maximum cut, making it a general framework. For classical GNN tasks like node classification, our training phase remains effective. By designing the sampling process to extract specific subgraphs, it can also be adapted to these tasks, whereas existing solutions cannot solve IM problems.

VII. Conclusion

In this work, we propose PrivIM, a novel node-level differentially private GNNs framework for IM tasks. To perform a better trade-off between privacy and utility, we design a unique dual-stage adaptive frequency sampling scheme within this framework. In particular, we first reduce the impact between individual nodes by dynamically adjusting the sampling probability for each node. Then, we incorporate supplementary subgraphs from boundary areas of the original graph, which boosts the utility without increasing privacy budget. Extensive empirical results on real-world graph datasets demonstrate both the effectiveness and efficiency of PrivIM framework.

As for future work, we plan to further explore the scalability of PrivIM, including its extension to different diffusion models, such as Linear Threshold (LT) model [4] and Susceptible-Infectious-Susceptible (SIS) model [51], as well as the adaptation to various DP settings, such as Local Differential Privacy (LDP). We also plan to investigate applications like targeted advertising in social networks, where PrivIM can identify key users to maximize advertisement reach while considering budget, fairness and ethical considerations.

References

[1] M. Habibi and A. Popescu-Belis, "Keyword Extraction and Clustering for Document Recommendation in Conversations," *IEEE/ACM TASLP*, pp. 746–759, 2015.

[2] X. Chen, Y. Zeng, G. Cong, S. Qin, Y. Xiang, and Y. Dai, "On Information Coverage for Location Category Based Point-of-Interest Recommendation," in *AAAI*, 2015, pp. 37–43.

[3] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance, "Cost-Effective Outbreak Detection in Networks," in *ACM SIGKDD*, 2007, pp. 420–429.

[4] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the Spread of Influence Through a Social Network," in *ACM SIGKDD*, 2003, pp. 137–146.

[5] W. Chen, C. Wang, and Y. Wang, "Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks," in *ACM SIGKDD*, 2010, pp. 1029–1038.

[6] T. Chen, W. Liu, Q. Fang, J. Guo, and D. Du, "Minimizing Misinformation Profit in Social Networks," *IEEE TCSS*, pp. 1206–1218, 2019.

[7] J. Guo, T. Chen, and W. Wu, "A Multi-Feature Diffusion Model: Rumor Blocking in Social Networks," *IEEE/ACM ToN*, pp. 386–397, 2021.

[8] C. Ling, J. Jiang, J. Wang, M. T. Thai, R. Xue, J. Song, M. Qiu, and L. Zhao, "Deep Graph Representation Learning and Optimization for Influence Maximization," in *ICML*, 2023, pp. 21 350–21 361.

[9] Y. Li, H. Gao, Y. Gao, J. Guo, and W. Wu, "A Survey on Influence Maximization: From an ML-Based Combinatorial Optimization," *ACM TKDD*, pp. 133:1–50, 2023.

[10] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," in *ACM CCS*, 2016, pp. 308–318.

[11] J. Fu, Q. Ye, H. Hu, Z. Chen, L. Wang, K. Wang, and X. Ran, "DPSUR: Accelerating Differentially Private Stochastic Gradient Descent Using Selective Update and Release," *PVLDB*, pp. 1200–1213, 2024.

[12] J. Liu, J. Lou, L. Xiong, J. Liu, and X. Meng, "Projected Federated Averaging with Heterogeneous Differential Privacy," *PVLDB*, pp. 828–840, 2021.

[13] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *ICLR*, 2017.

[14] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *ICLR*, 2019.

[15] 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).

[16] Z. Xiang, T. Wang, and D. Wang, "Preserving Node-level Privacy in Graph Neural Networks," in *IEEE S&P*, 2024, pp. 4714–4732.

[17] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation," in *USENIX Security*, 2023, pp. 3223–3240.

[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE TNNLS*, pp. 4–24, 2021.

[19] W.-K. Chen, *Applied Graph Theory*, 2012.

[20] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate Estimation of the Degree Distribution of Private Networks," in *IEEE ICDM*, 2009, pp. 169–178.

[21] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *TCC*, 2006, pp. 265–284.

[22] C. Dwork, "Differential Privacy," in *ICALP*, 2006, pp. 1–12.

[23] D. Kifer and A. Machanavajjhala, "No Free Lunch in Data Privacy," in *ACM SIGMOD*, 2011, pp. 193–204.

[24] Y. Jiang, X. Luo, Y. Yang, and X. Xiao, "Calibrating Noise for Group Privacy in Subsampled Mechanisms," *arXiv:2408.09943*, (2024).

[25] I. Mironov, "Rényi Differential Privacy," in *IEEE CSF*, 2017, pp. 263–275.

[26] T. V. Erven and P. Harremoës, "Rényi Divergence and Kullback-Leibler Divergence," *IEEE TIT*, pp. 3797–3820, 2014.

[27] C. L. Canonne, G. Kamath, and T. Steinke, "The Discrete Gaussian for Differential Privacy," in *NeurIPS*, 2020.

[28] Y. Tang, Y. Shi, and X. Xiao, "Influence Maximization in Near-Linear Time: A Martingale Approach," in *ACM SIGMOD*, 2015, pp. 1539–1554.

[29] M. A. Rahimian, F.-Y. Yu, Y. Liu, and C. Hurtado, "Seeding with Differentially Private Network Information," *arXiv:2305.16590*, (2023).

[30] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and Mobility: User Movement in Location-Based Social Networks," in *ACM SIGKDD*, 2011, pp. 1082–1090.

[31] P. Erdös, "Graph Theory and Probability," *Canadian Journal of Mathematics*, pp. 34–38, 1959.

[32] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small-World' Networks," *Nature*, pp. 440–442, 1998.

[33] H. Tong, C. Faloutsos, and J. Pan, "Fast Random Walk with Restart and its Applications," in *IEEE ICDM*, 2006, pp. 613–622.

[34] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local Higher-Order Graph Clustering," in *ACM SIGKDD*, 2017, pp. 555–564.

[35] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge Weight Prediction in Weighted Signed Networks," in *IEEE ICDM*, 2016, pp. 221–230.

[36] B. Rozemberczki and R. Sarkar, "Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models," in *ACM CIKM*, 2020, pp. 1325–1334.

[37] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-Scale Attributed Node Embedding," *J. Complex Networks*, 2021.

[38] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, "Graph Evolution: Densification and Shrinking Diameters," *ACM TKDD*, p. 2, 2007.

[39] J. Yang and J. Leskovec, "Defining and Evaluating Network Communities Based on Ground-Truth," in *IEEE ICDM*, 2012, pp. 1–8.

[40] N. Karalias and A. Loukas, "Erdos Goes Neural: An Unsupervised Learning Framework for Combinatorial Optimization on Graphs," in *NeurIPS*, 2020.

[41] R. Ni, X. Li, F. Li, X. Gao, and G. Chen, "Fastcover: An Unsupervised Learning Framework for Multi-Hop Influence Maximization in Social Networks," *arXiv:2111.00463*, (2021).

[42] R. Wang, L. Shen, Y. Chen, X. Yang, D. Tao, and J. Yan, "Towards One-Shot Neural Combinatorial Solvers: Theoretical and Empirical Notes on the Cardinality-Constrained Case," in *ICLR*, 2023.

[43] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," *NeurIPS*, 2017.

[44] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," *arXiv:1710.10903*, (2017).

[45] Z. Li, Q. Chen, and V. Koltun, "Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search," in *NeurIPS*, 2018, pp. 537–546.

[46] X. Chen and Y. Tian, "Learning to Perform Local Rewriting for Combinatorial Optimization," in *NeurIPS*, 2019, pp. 6278–6289.

[47] A. Paulus, M. Rolínek, V. Musil, B. Amos, and G. Martius, "CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints," in *ICML*, 2021, pp. 8443–8453.

[48] Y. Yang, H. Yuan, B. Hui, N. Z. Gong, N. Fendley, P. Burlina, and Y. Cao, "Fortifying Federated Learning against Membership Inference Attacks via Client-Level Input Perturbation," in *IEEE/IFIP DSN*, 2023, pp. 288–301.

[49] X. Ran, Q. Ye, H. Hu, X. Huang, J. Xu, and J. Fu, "Differentially Private Graph Neural Networks for Link Prediction," in *IEEE ICDE*, 2024, pp. 1632–1644.

[50] C. Yang, H. Wang, L. Sun, and B. Li, "Secure Network Release with Link Privacy," *arXiv:2005.00455*, (2020).

[51] W. O. Kermack and A. G. McKendrick, "A Contribution to the Mathematical Theory of Epidemics," *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, pp. 700–721, 1927.

Appendix

A. Useful Lemmas

The following lemmas are used in Section III and IV for further proofs.

Lemma 3. *For any finite collection of events $A_1, A_2, \ldots, A_n$, the probability of their union satisfies:*

$$\Pr\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} \Pr(A_i). \tag{13}$$

Lemma 4. *For any $d \in \mathbb{N}$ and arbitrary distributions $P_1, P_2, \ldots, P_d$ and $Q_1, Q_2, \ldots, Q_d$, let $P^d := P_1 \times \cdots \times P_d$ and $Q^d := Q_1 \times \cdots \times Q_d$. Then we have:*

$$\mathcal{D}_\alpha(P^d \| Q^d) = \sum\nolimits_{i=1}^{d} \mathcal{D}_\alpha(P_i \| Q_i). \tag{14}$$

Lemma 5. *Let $\mathcal{N}(\mu, \sigma^2)$ denote the one-dimensional Gaussian distribution with mean μ and variance σ^2, it holds that:*

$$\mathcal{D}_\alpha\left(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0, \sigma^2)\right) = \mathcal{D}_\alpha\left(\mathcal{N}(0, \sigma^2) \| \mathcal{N}(\mu, \sigma^2)\right) = \frac{\alpha\mu^2}{2\sigma^2} \tag{15}$$

Lemma 6. *Let $\mu_0, \ldots, \mu_n$ and $\upsilon_0, \ldots, \upsilon_n$ be probability distributions such that: $\mathcal{D}_\alpha(\mu_0 \| \upsilon_0) \leq \gamma_0, \ldots, \mathcal{D}_\alpha(\mu_n \| \upsilon_n) \leq \gamma_n$, for some given $\gamma_0, \ldots, \gamma_n$. Let ρ be a probability distribution over $\{0, \ldots, n\}$. μ_ρ and υ_ρ are the probability distributions obtained by first sampling i from ρ and then randomly sampling from μ_i and υ_i. Therefore, we have:*

$$\begin{aligned}\mathcal{D}_\alpha(\mu_\rho \| \upsilon_\rho) &\leq \log \mathbb{E}_{i \sim \rho}\left[\exp\left((\alpha - 1)\gamma_i\right)\right] \\ &= \frac{1}{\alpha - 1} \log\left(\sum\nolimits_{i=0}^{n} \rho_i \exp\left((\alpha - 1)\gamma_i\right)\right).\end{aligned} \tag{16}$$

B. Proof of Theorem 2

Theorem 2. *For a graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$, node feature matrix $\mathbf{H}^{(j-1)}$ and seed set S_{j-1} after $(j-1)$-step diffusion, let $p_j(u|S_{j-1})$ denote the probability that node u is influenced by S_{j-1} under the j-th step in IC model. Using message passing, the upper bound $\hat{p}_j(u|S_{j-1})$ is:*

$$\begin{aligned}\hat{p}_j(u|S_{j-1}) &= \phi(\mathbf{A}_u \cdot \mathbf{H}^{(j-1)}) = \phi\left(\sum\nolimits_{v \in \mathcal{N}(u)} w_{vu} \mathbf{h}_v^{(j-1)}\right) \\ &\geq \phi\left(1 - \prod\nolimits_{v \in \mathcal{N}(u)} \left(1 - w_{vu} \mathbf{h}_v^{(j-1)}\right)\right) \\ &= p_j(u|S_{j-1}),\end{aligned} \tag{17}$$

Proof. We start with the 1-step diffusion case and introduce the following lemma.

Lemma 7. *Given a graph $G = (V, E)$, adjacency matrix $\mathbf{A}$, node feature matrix $\mathbf{X}$, and seed set S. Let $p(u|S)$ denote the probability that node u is influenced by S under 1-step diffusion process in IC model. Using message passing, we compute the upper bound $\hat{p}(u|S)$ for $p(u|S)$ as follows:*

$$\begin{aligned}\hat{p}(u|S) &= \phi\left(\mathbf{A}_u \cdot \mathbf{X}\right) = \phi\left(\sum\nolimits_{v \in \mathcal{N}(u)} w_{vu} \mathbf{h}_v^{(0)}\right) \\ &\geq \phi\left(1 - \prod\nolimits_{v \in \mathcal{N}(u)} \left(1 - w_{vu} \mathbf{h}_v^{(0)}\right)\right) = p(u|S),\end{aligned} \tag{18}$$

where the inequality is due to Boole's inequality (See Lemma 3), and $\phi(\cdot)$ is an activation function that maps the result to range $[0, 1]$.

Therefore, we requires estimating recursively the probability $p(u|S)$ to achieve the j-step diffusion probability $p_j(u|S_{j-1})$. □

C. Proof of Lemma 1

Lemma 1. *Given an original graph $G = (V, E)$, a fixed subgraph size n, the maximum node in-degree θ, and an r-layer GNN model. Algorithm 1 ensures that the maximum occurrence of any node across all subgraphs is bounded by N_g, where:*

$$N_g = \sum\nolimits_{i=0}^{r} \theta^i = \frac{\theta^{r+1} - 1}{\theta - 1}. \tag{19}$$

Proof. Given an original graph $G = (V, E)$, we first project it into a θ-bounded graph $G^\theta = (V^\theta, E^\theta)$. When considering the aggregation of one-hop neighbors, if one of θ neighbors of node v is selected as the starting node v_0, in the worst case, node v will be included in the same subgraph as v_0 after sampling. Therefore, the maximum number of times node v can appear in the same subgraph as any of its one-hop neighbors is θ.

Then, for an r-layer GNN model, we extend this to the r-hop neighborhood of node v. If any of the r-hop neighbors of node v is chosen as the starting node v_0, in the worst case, node v will be included in the same subgraph as well. Hence, for our Algorithm 1, the maximum number of occurrences of any node is bounded by $N_g = \sum_{i=0}^{r} \theta^i = \frac{\theta^{r+1}-1}{\theta-1}$. □

D. Proof of Lemma 2

Lemma 2. *Given two adjacent graphs $G, G' \subset \mathcal{G}$ that differ by at most one node, a fixed subgraph size n, the maximum node in-degree θ and the clipping bound C. Let the aggregated batched gradients for two adjacent graphs be $\overline{\mathbf{g}}_t$ and $\overline{\mathbf{g}'}_t$, respectively. Then the node-level sensitivity $\Delta_\mathbf{g}$ satisfies the following inequality:*

$$\Delta_\mathbf{g} = \|\overline{\mathbf{g}}_t - \overline{\mathbf{g}'}_t\|_2 \leq C \cdot N_g. \tag{20}$$

Proof. Given two adjacent datasets G, G' differ by only one node v. In Algorithm 2, we have the clipped gradients $\hat{\mathbf{g}}_t(G)$, $\hat{\mathbf{g}}_t(G')$ computed on G, G', respectively. Our goal is to compute the maximum l_2-norm difference between the sum of clipped gradients in a batch $\mathcal{B}_t$ at iteration t. Since the maximum number of occurrences of any node is denoted as N_g, we have the following inequality:

$$\begin{aligned}\Delta_\mathbf{g} &= \|\overline{\mathbf{g}}_t - \overline{\mathbf{g}'}_t\|_2 \\ &= \sum\nolimits_{S_v \in (N_g \cap \mathcal{B}_t)} \|\hat{\mathbf{g}}_t(G) - \hat{\mathbf{g}}_t(G')\|_2 \\ &= \sum\nolimits_{S_v \in (N_g \cap \mathcal{B}_t)} \|\mathrm{Clip}_C(\mathbf{g}_t(G)) - \mathrm{Clip}_C(\mathbf{g}_t(G'))\|_2.\end{aligned} \tag{21}$$

Since the values of both $\mathrm{Clip}_C(\mathbf{g}_t(G))$ and $\mathrm{Clip}_C(\mathbf{g}_t(G'))$ are within the range $[0, C]$, the following inequality holds:

$$\begin{aligned}\Delta_\mathbf{g} &= \|\overline{\mathbf{g}}_t - \overline{\mathbf{g}'}_t\|_2 \\ &= \sum\nolimits_{S_v \in (N_g \cap \mathcal{B}_t)} \|\mathrm{Clip}_C(\mathbf{g}_t(G)) - \mathrm{Clip}_C(\mathbf{g}_t(G'))\|_2 \\ &\leq C \cdot N_g.\end{aligned} \tag{22}$$

□

E. Proof of Theorem 3

Theorem 3. *Let $\mathcal{G}_{\text{sub}}$ be the subgraph container with $|\mathcal{G}_{\text{sub}}| = m$, B be the batch size, N_g be the upper bound of node occurrences. Over T iterations, Algorithm 2 satisfies node-level $(\alpha, \gamma T)$-Rényi DP, where*

$$\gamma = \frac{1}{\alpha - 1} \log \left(\sum\nolimits_{i=0}^{N_g} \rho_i \exp \left(\frac{\alpha(\alpha-1)i^2}{2N_g^2\sigma^2} \right) \right), \\ \rho_i = \binom{B}{i} \left(\frac{N_g}{m}\right)^i \left(1 - \frac{N_g}{m}\right)^{B-i}. \tag{23}$$

Proof. To prove the node-level DP guarantee for our Algorithm 3, we first adopt the definition of the subsampled mechanism used in the DP-SGD algorithm [10], as follows:

Definition 8. *Given an input subgraph container $\mathcal{G}_{\text{sub}}$, the subsampled mechanism constructs a trainset $\mathcal{G}_{\text{tr}} \subseteq \mathcal{G}_{\text{sub}}$ by including each subgraph $G_{\text{sub}} \in \mathcal{G}_{\text{sub}}$ into $\mathcal{G}_{\text{tr}}$ independently with a fixed probability of $q_s \in (0, 1)$. $\mathcal{A}$ is then performed on $\mathcal{G}_{\text{tr}}$ to produce the privatized output. Formally, it is defined as:*

$$\mathcal{M}(\mathcal{G}_{\text{sub}}) \triangleq \mathcal{A}(\text{Subsample}(\mathcal{G}_{\text{sub}})),$$

where $\text{Subsample}(\mathcal{G}_{\text{sub}})$ *denotes the subsampling procedure that constructs the trainset $\mathcal{G}_{\text{tr}}$ from $\mathcal{G}_{\text{sub}}$.*

Here the output distribution $\mathcal{M}$ is essentially a mixture distribution, where the components of the mixture are distributions of $\mathcal{A}(\mathcal{G}_{\text{tr}})$ for all $\mathcal{G}_{\text{tr}} \subseteq \mathcal{G}_{\text{sub}}$. Therefore, we need to carefully analyze both the subsampling procedure and the randomized algorithm $\mathcal{A}$ individually.

In this paper, we focus on the subsampling procedure which samples each subgraph G_{sub} independently with a constant probability $q_s = \frac{1}{m}$, where $|\mathcal{G}_{\text{sub}}| = m$. We know from Lemma 1 that the maximum occurrence of any node across all subgraphs is bounded by N_g, and we define the total number of times subgraphs in N_g are sampled during B independent sampling iterations as ρ. Then the distribution of ρ follows the Binomial distribution $\text{Binomial}(B, \frac{N_g}{m})$:

$$\rho \sim \text{Binomial}(B, \frac{N_g}{m}), \tag{24}$$

$$\rho_i = P\left[\rho = i\right] = \binom{B}{i} \left(\frac{N_g}{m}\right)^i \left(1 - \frac{N_g}{m}\right)^{B-i}. \tag{25}$$

Next we consider the randomized algorithm $\mathcal{A}$. For any iteration t of Algorithm 2, let G, G' be two adjacent original graphs that differ by at most one node, based on Line 8 in Algorithm 2, we have:

$$\tilde{\mathbf{g}}_t = \overline{\mathbf{g}}_t + \mathcal{N}(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d), \\ \tilde{\mathbf{g}'}_t = \overline{\mathbf{g}'}_t + \mathcal{N}(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d), \tag{26}$$

where $\tilde{\mathbf{g}}_t$ and $\tilde{\mathbf{g}'}_t$ represent the private gradients of G and G'. Then we need to show that $\mathcal{D}_\alpha(\tilde{\mathbf{g}}_t \| \tilde{\mathbf{g}'}_t) \leq \gamma_t$. When $\rho = i$, The Rényi divergence between $\tilde{\mathbf{g}}_{t,i}$ and $\tilde{\mathbf{g}'}_{t,i}$ can be upper bounded as:

$$\begin{aligned} &\mathcal{D}_\alpha(\tilde{\mathbf{g}}_{t,i} \| \tilde{\mathbf{g}'}_{t,i}) \\ &= \mathcal{D}_\alpha\left(\overline{\mathbf{g}}_{t,i} + \mathcal{N}\left(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right) \| \overline{\mathbf{g}'}_{t,i} + \mathcal{N}\left(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right)\right) \\ &= \mathcal{D}_\alpha\left(\mathcal{N}\left(\overline{\mathbf{g}}_{t,i}, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right) \| \mathcal{N}\left(\overline{\mathbf{g}'}_{t,i}, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right)\right) \\ &\overset{(a)}{=} \mathcal{D}_\alpha\left(\mathcal{N}\left((\overline{\mathbf{g}}_{t,i} - \overline{\mathbf{g}'}_{t,i}), \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right) \| \mathcal{N}\left(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right)\right) \\ &\overset{(b)}{\leq} \sup_{\|\mathbf{v}\|_2 \leq iC} \mathcal{D}\left(\mathcal{N}\left(\mathbf{v}, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right) \| \mathcal{N}\left(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right)\right) \\ &\overset{(c)}{=} \sup_{\|\mathbf{v}\|_2 \leq iC} \sum_{x=1}^{d} \mathcal{D}\left(\mathcal{N}\left(\mathbf{v}\left[x\right], \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right) \| \mathcal{N}\left(0, \sigma^2\Delta_{\mathbf{g}}^2\mathbb{I}^d\right)\right) \\ &\overset{(d)}{=} \sup_{\|\mathbf{v}\|_2 \leq iC} \sum_{x=1}^{d} \frac{\alpha \mathbf{v}\left[x\right]^2}{2\Delta_{\mathbf{g}}^2\sigma^2} = \sup_{\|\mathbf{v}\|_2 \leq iC} \sum_{x=1}^{d} \frac{\alpha \|\mathbf{v}\|_2^2}{2\Delta_{\mathbf{g}}^2\sigma^2} = \frac{\alpha i^2}{2N_g^2\sigma^2}, \end{aligned} \tag{27}$$

where (a) follows from the invariance of Rényi divergence under invertible transformations [26]; (b), (c), (d) are derived from Lemma 2, Lemma 4, and Lemma 5, respectively. With Eq. 24, 25, 27 and Lemma 6, we can derive the final $\mathcal{D}_\alpha(\tilde{\mathbf{g}}_t \| \tilde{\mathbf{g}'}_t) \leq \gamma_t$ as:

$$\begin{aligned} \mathcal{D}_\alpha(\tilde{\mathbf{g}}_t \| \tilde{\mathbf{g}'}_t) &\leq \log \mathbb{E}_{i \sim \rho}\left[\exp\left((\alpha - 1)\,\mathcal{D}_\alpha(\tilde{\mathbf{g}}_{t,i} \| \tilde{\mathbf{g}'}_{t,i})\right)\right] \\ &= \frac{1}{\alpha - 1} \log \left(\sum_{i=0}^{N_g} \rho_i \exp \left(\frac{\alpha(\alpha-1)i^2}{2N_g^2\sigma^2} \right) \right). \end{aligned} \tag{28}$$

We define $\gamma = \frac{1}{\alpha-1} \log \left(\sum_{i=0}^{N_g} \rho_i \exp \left(\frac{\alpha(\alpha-1)i^2}{2N_g^2\sigma^2} \right) \right)$, where $\rho_i = \binom{B}{i} \left(\frac{N_g}{m}\right)^i \left(1 - \frac{N_g}{m}\right)^{B-i}$ and the proof is completed. □

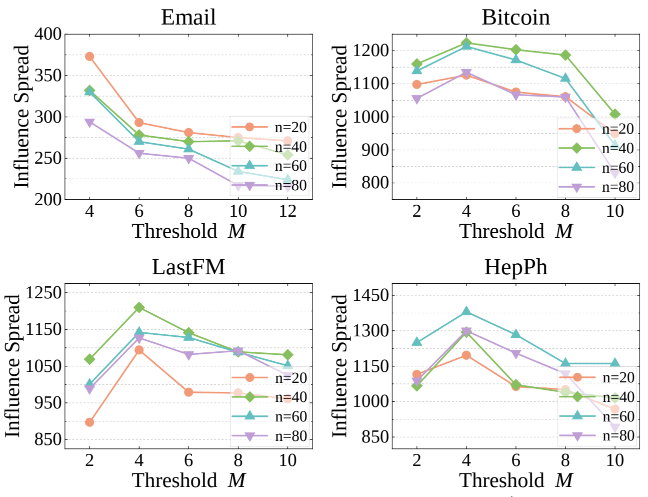


Fig. 10: Impact of threshold M on PrivIM* ($\epsilon = 3$)

F. More Results about Parameter Studies

In this section, we report the results on other datasets. Specifically, Figure 10 and Figure 11 show the impact of threshold M and subgraph size n on the performance of our proposed PrivIM*, respectively. Additionally, Fig. 12 reports the values of our designed indicator and experimental results.

G. GNN Models and Further Discussions

In this section, we describe our used GNN models and provide further discussions about model explainability. In our evaluations, the following five GNN models are leveraged:

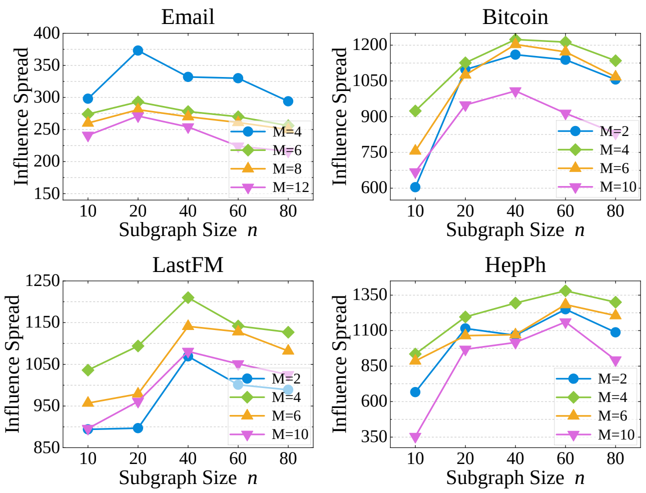


Fig. 11: Impact of subgraph size n on PrivIM* ($\epsilon = 3$)

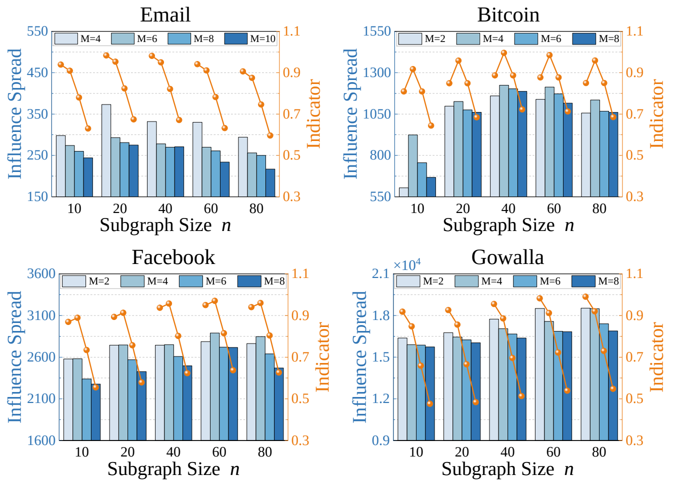


Fig. 12: Theoretical values of the indicator and empirical results of PrivIM* ($\epsilon = 3$)

- GraphSAGE: It learns node representations by sampling neighbors and aggregating their information. The formulation is:

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \text{AGG}\left(\mathbf{h}_u^{(k)} : u \in \mathcal{N}(v)\right), \tag{29}$$

$$\mathbf{h}_v^{(k+1)} = \phi\left(\mathbf{W}^{(k)}\left[\mathbf{h}_v^{(k)} | \mathbf{m}_{\mathcal{N}(v)}^{(k)}\right]\right), \tag{30}$$

where AGG$(\cdot)$ denotes an aggregation function, which can be mean, sum, or max. $[\cdot|\cdot]$ represents concatenation, $\mathbf{h}_v^{(k)}$ is the feature of node v at layer k, $\mathbf{W}^{(k)}$ is the learnable weight matrix at layer k, and ϕ is a non-linear activation function.

- GCN: It aggregates the information from a node's neighbors and normalizes the result. The formulation is:

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \sum\nolimits_{u\in\mathcal{N}(v)} \frac{\mathbf{h}_u^{(k)}}{\sqrt{d_v d_u}}, \tag{31}$$

$$\mathbf{h}_v^{(k+1)} = \phi\left(\mathbf{W}^{(k)}\mathbf{m}_{\mathcal{N}(v)}^{(k)}\right), \tag{32}$$

where d_v is the degree of node v, ϕ is a non-linear activation function, and $\mathbf{W}^{(k)}$ is the learnable weight matrix at layer k. Eq. 31 and Eq. 32 show that GCN has a stable neighbor information aggregation mechanism but suffers from the error introduced by overlapping coverage.

- GAT: It leverages the attention mechanism to assign different weights to neighbors. The formulation is:

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \sum\nolimits_{u\in\mathcal{N}(v)} \alpha_{vu}^{(k)}\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}, \tag{33}$$

$$\mathbf{h}_v^{(k+1)} = \phi\left(\mathbf{m}_{\mathcal{N}(v)}^{(k)}\right), \tag{34}$$

where $\alpha_{vu}^{(k)}$ is the attention coefficient of neighbor u for node v, calculated as:

$$\alpha_{ij}^{(k)} = \frac{\exp\left(e_{ij}\right)}{\sum_{l\in\mathcal{N}(v)}\exp\left(e_{il}\right)}, \tag{35}$$

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^{(k)T}\left[\mathbf{W}^{(k)}\mathbf{h}_i^{(k)} \| \mathbf{W}^{(k)}\mathbf{h}_j^{(k)}\right]\right). \tag{36}$$

In summary, the core of GAT lies in utilizing the attention weight e_{ij} between node pairs, normalizing it to obtain α_{ij}, and then aggregating the neighbors' information by weighting them based on the attention coefficients. Although GAT utilizes an attention mechanism, its normalization approach on target nodes is not well-suited for IM tasks, which require distinguishing the influence differences between nodes.

- GRAT: It is a variant of GAT, which integrates the attention mechanism at the source nodes instead of the destination nodes. Therefore, the formulation is:

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \sum\nolimits_{u\in\mathcal{N}(v)} \alpha_{vu}^{(k)}\mathbf{W}^{(k)}\mathbf{h}_u^{(k)}, \tag{37}$$

$$\mathbf{h}_v^{(k+1)} = \phi\left(\mathbf{m}_{\mathcal{N}(v)}^{(k)}\right). \tag{38}$$

The attention coefficients are normalized for each source node $v \in V$ over its successors $\mathcal{N}^+(v)$, calculated as:

$$\alpha_{ij}^{(k)} = \frac{\exp\left(e_{ij}\right)}{\sum_{l\in\mathcal{N}^+(v)}\exp\left(e_{il}\right)}, \tag{39}$$

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^{(k)T}\left[\mathbf{W}^{(k)}\mathbf{h}_i^{(k)} \| \mathbf{W}^{(k)}\mathbf{h}_j^{(k)}\right]\right). \tag{40}$$

Therefore, if a node's coverage overlaps with that of others, the reward it receives will be reduced by the attention mechanism, which helps mitigate the overlapping effect between different nodes' coverage, resulting in the best performance in IM tasks.

- GIN: It updates the node features with a multi-layer perceptron (MLP). The formulation is:

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \sum\nolimits_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(k)}, \tag{41}$$

$$\mathbf{h}_v^{(k+1)} = \text{MLP}^{(k)}\left(\mathbf{m}_{\mathcal{N}(v)}^{(k)} + (1+\omega^{(k)})\mathbf{h}_v^{(l)}\right), \tag{42}$$

where $\omega^{(k)}$ is a learnable parameter. GIN's advantage in capturing feature differences is attributed to its complex nonlinear aggregation mechanism. However, in IM task, this mechanism may lead to overfitting, which negatively impacts its performance.

H. *Parameters of the Indicator*

In this section, we provide the method for determining the parameters $\psi_n, \psi_M, k_n, k_M, b_n, b_M$ of Eq. 10, 11, 12. We take n as an example, for:

$$\xi(n;\beta_n,\psi_n) = \frac{n^{\beta_n-1}e^{-\frac{n}{\psi_n}}}{\psi_n^{\beta_n}\Gamma(\beta_n)}, \tag{43}$$

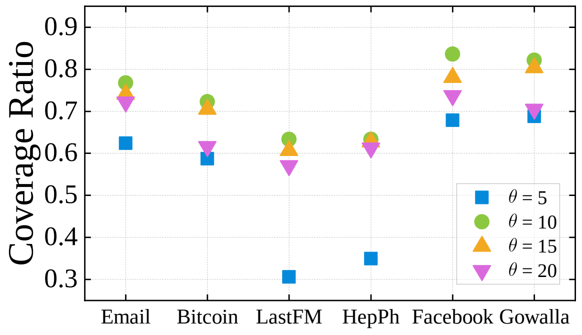


Fig. 13: Coverage Ratio of PrivIM with different θ ($\epsilon = 3$)

our purpose is to minimize the error between its maximum value and the values from the prior experiments in Section V-C. Therefore, we first take the logarithm of Eq. 43:

$$\ln(\xi(n;\beta_n,\psi_n)) = (\beta_n - 1)\ln(n) - \frac{n}{\psi_n} - \beta_n \ln(\psi_n) - \ln(\Gamma(\beta_n)), \tag{44}$$

and we differentiate this equation w.r.t. n:

$$\frac{d}{dn}\ln(\xi(n;\beta_n,\psi_n)) = \frac{\beta_n - 1}{n} - \frac{1}{\psi_n}, \tag{45}$$

then, we set the derivative of the equation equal to zero to find the critical point corresponding to the maximum value:

$$n_{\max} = (\beta_n - 1)\psi_n. \tag{46}$$

As we mentioned before, the variation in dataset size $|V|$ is significantly larger than that of n and M. Therefore, we leverage Eq. 12 to fit the trend accordingly. Combining Eq. 12 and Eq. 46, we have:

$$\frac{n}{\psi_n} = k_n \ln(|V|) + b_n - 1. \tag{47}$$

To obtain the parameters ψ_n, k_n, b_n, we first treat ψ_n as a constant and leverage the least squares method to solve for the values of k_n and b_n as follows:

$$k_n = \frac{t\sum_{i=1}^{t}\ln(|V_i|)\frac{n_i}{\psi_n} - \sum_{i=1}^{t}\ln(|V_i|)\sum_{i=1}^{t}\frac{n_i}{\psi_n}}{t\sum_{i=1}^{t}\ln^2(|V_i|) - \left(\sum_{i=1}^{t}\ln(|V_i|)\right)^2}, \tag{48}$$

$$b_n = \frac{\sum_{i=1}^{t}\frac{n_i}{\psi_n} - k_n\sum_{i=1}^{t}\ln(|V_i|) + t}{t}, \tag{49}$$

similarly, we can achieve k_M, b_M as follows:

$$k_M = \frac{t\sum_{i=1}^{t}\ln(1/|V_i|)\frac{M_i}{\psi_M} - \sum_{i=1}^{t}\ln(1/|V_i|)\sum_{i=1}^{t}\frac{M_i}{\psi_M}}{t\sum_{i=1}^{t}\ln^2(1/|V_i|) - \left(\sum_{i=1}^{t}\ln(1/|V_i|)\right)^2}, \tag{50}$$

$$b_M = \frac{\sum_{i=1}^{t}\frac{M_i}{\psi_M} - k_M\sum_{i=1}^{t}\ln(1/|V_i|) + t}{t}. \tag{51}$$

Based on the above equations, when we fix the values of ψ_n, ψ_M, we can obtain the corresponding values of k_n, b_n, k_M, b_M. For all the datasets, we set $\psi_n = 25, k_n = 0.47, b_n = -1.03$ and $\psi_M = 5, k_M = 4.02, b_M = 1.22$, respectively.

I. Prior Experiments for θ

In this section, we evaluate the impact of different θ values on the performance of PrivIM. As shown in Figure 13, we vary θ from $\{5, 10, 15, 20\}$ and assess the performance across six datasets. Our results indicate that both very small value ($\theta = 5$) and very large value ($\theta = 20$) negatively affect the coverage ratio. Specifically, a small θ undermines the structural information of subgraphs, while a large θ introduces excessive noise. The best performance is generally observed when $\theta = 10$, which we adopt for our overall experiments.

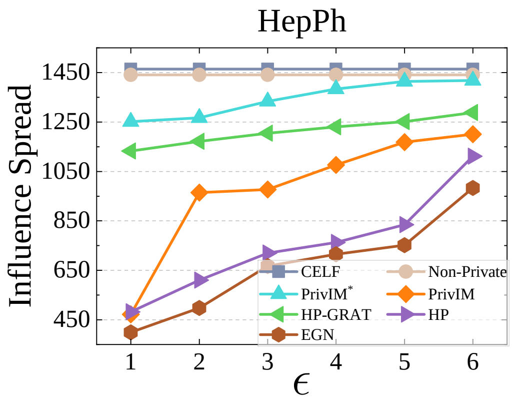


Fig. 14: Influence spread of all the methods on HepPh by varying privacy budget ϵ

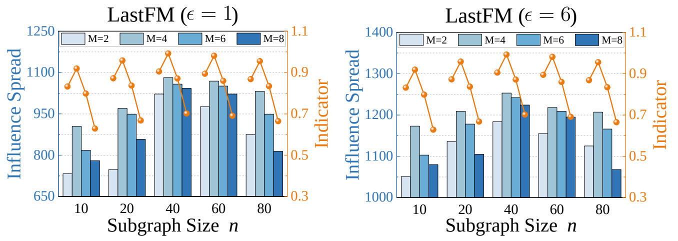


Fig. 15: Theoretical values of the indicator and empirical results of PrivIM* ($\epsilon = 1$ and $\epsilon = 6$)

J. Overall Results on HepPh dataset

In this section, we evaluate the performance of all competitors on HepPh and report the influence spread in Fig 14. Our method PrivIM* consistently surpasses other methods, particularly under a more restrictive privacy budget.

K. More Results about Indicator Effectiveness

We evaluate our indicator with different privacy budgets ($\epsilon = 1$ and $\epsilon = 6$) on LastFM, and report the values in Figure 15. Bars show the empirical results of influence spread by our PrivIM*, and curves present the theoretical values returned by our indicator. We observe that our indicator still captures the similar trends of experimental results under different privacy budgets, which provides compelling evidence for the effectiveness of our designed indicator.

L. Descriptions of Datasets

- **Email**: This dataset is collected from a large European research institution to record all incoming and outgoing emails between members of the research institution.
- **Bitcoin**: This dataset is a who-trusts-whom network of people who trade using Bitcoin on a platform called Bitcoin OTC.

- **LastFM**: This dataset is collected from the public API to record the relationships from LastFM users in March 2020.
- **HepPh**: This dataset contains scientific collaborations between authors papers submitted to High Energy Physics - Phenomenology (HepPh) from January 1993 to April 2003.
- **Facebook**: This dataset is collected through the Facebook Graph API in November 2017. In this dataset, nodes represent official Facebook pages while the links are mutual likes between sites.
- **Gowalla**: This dataset is collected from a location-based service to record the check-in behaviors between users by their public API.
- **Friendster**: This dataset is collected from a social networking site where users can form friendship edge each other.