Subgraph Counting under Edge Local Differential Privacy Based on Noisy Adjacency Matrix

Jintao Guo [*] *Ying Zhou* [†] *Chao Li* [‡] *Guixun Luo* [§]

Abstract

When analyzing connection patterns within graphs, subgraph counting serves as an effective and fundamental approach. Edge-local differential privacy(edge-LDP) and shuffle model have been employed to achieve subgraph counting under a privacy-preserving situation. Existing algorithms are plagued by high time complexity, excessive download costs, low accuracy, or dependence on trusted third parties.

To address the aforementioned challenges, we propose the Noisy Adjacency Matrix(NAM), which combines differential privacy with the adjacency matrix of the graph. NAM offers strong versatility and scalability, making it applicable to a wider range of DP variants, DP mechanisms, and graph types. Based on NAM, we designed 5 algorithms (*TriOR*, *TriTR*, *TriMTR*, *QuaTR*, and *2STAR*) to count 3 types of subgraphs: triangles, quadrangles, and 2-stars. Theoretical and experimental results demonstrate that in triangle counting, *TriOR* maximizes accuracy with reduced time complexity among one-round algorithms, *TriTR* achieves optimal accuracy, *TriMTR* achieves the highest accuracy under low download costs, and *QuaTR* stands as the first quadrangle counting algorithm under pure edge-LDP. We implement edge-LDP for noisy data via a confidence interval-inspired method, providing DP guarantees on randomized data. Our *2STAR* algorithm achieves the highest accuracy in 2-star counting and can be derived as a byproduct of two-round triangle or quadrangle counting algorithms, enabling efficient joint estimation of triangle, quadrangle, and 2-star counts within two query rounds.

1 Introduction

Background. Subgraph counting is a fundamental task to analyze connection patterns in graph data obtained from social networks, communication networks, or transaction networks.

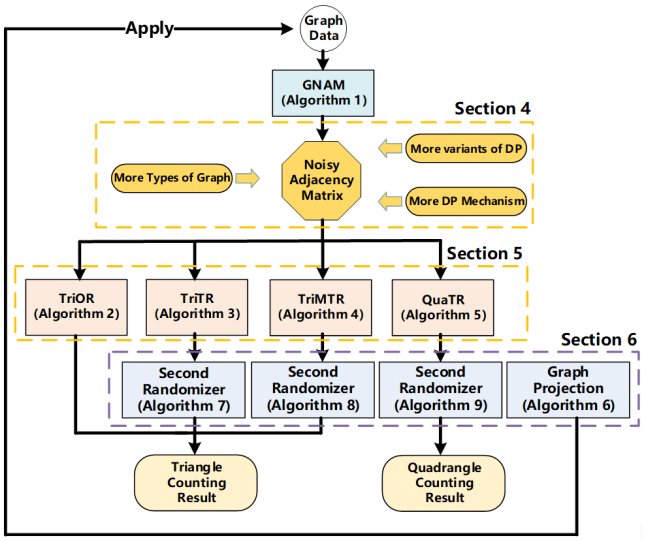

Figure 1: Algorithm Application Processes. The black arrows represent the actual execution workflow. (1) For the one-round algorithm TriOR, the process starts directly from the *Graph Data*. (2) All other algorithms are two-round algorithms, they need to apply ***Graph Projection*** on ***Graph Data*** before subsequent procedures.

Subgraphs such as triangle, quadrangle, and 2-star counting are three of the most basic tasks. The difficulty of triangle and quadrangle counting arises from the limited scope of users(nodes), as it typically assumes that users only know the relationship(edge) with their friends(neighboring nodes). However, the process of analyzing and generating statistical results can lead to the disclosure of sensitive data, with edge privacy being the most representative case [22]. This is because edges have strong real-world implications, representing relationships such as friendships, transactions, and communications. Therefore, a lot of research has been conducted on how to count the number of subgraphs while preserving user's edge privacy [35, 42].

Considering the security risks associated with central

[*]Beijing Jiaotong University
[†]Beijing Jiaotong University
[‡]Beijing Jiaotong University
[§]Beijing Jiaotong University

servers [39], Local Differential Privacy (LDP) [1,26,35,40,54] is commonly employed to protect the privacy of edges in a graph, specifically known as edge-LDP [43, 46].

Existing Solutions and Their Limitations. Since subgraph statistics typically focus only on the binary existence of edges between two nodes, current algorithms [18,22–24] universally employ Randomized Response(RR) [57] to perturb graph data to achieve edge-LDP and then employ empirical estimation methods to obtain an estimator for the number of subgraphs.

This leads to some limitations with existing algorithms. For example, existing one-round algorithms [18, 22] all use the enumeration of triplets of nodes to perform estimation, making their $O(n^3)$ time complexity difficult to handle large-scale networks; two-round algorithms [22, 23] face significant transmission cost, and to address this issue, they employ edge sampling, thereby sacrificing a large amount of useful data, which results in suboptimal algorithm accuracy; currently there is no pure edge-LDP algorithm to estimate the number of quadrangles [24]. The challenges of current algorithms arise from a lack of deep understanding of the intrinsic properties of graphs and the absence of an effective framework that integrates differential privacy mechanisms with the graph's adjacency matrix.

Our Intuition. In the context of privacy-preserving subgraph counting, any perturbation strategy can be viewed as adding noise to the original graph data, and subsequent downstream tasks are performed on the noisy data. In the context of edge-LDP, the core mechanism involves perturbing the adjacency lists of nodes in the graph, which essentially equates to perturbing the entire adjacency matrix representation of the graph. Therefore, our intuition is to directly establish a relationship between differential privacy mechanisms and the adjacency matrix, which we define as the Noisy Adjacency Matrix(NAM). Based on this, we designed 4 algorithms to achieve subgraph counting.

Figure 1 illustrates the main structure of this paper and the algorithm application processes. In Section 4, we define the NAM, analyze its properties, and give the algorithm **GNAM** to generate it. In Section 5, leveraging the properties of NAM, we designed four algorithms, Triangle's One-Round Algorithm (**TriOR**), Triangle's Two-Round Algorithm (**TriTR**), Triangle's Modified Two-Round Algorithm (**TriMTR**), Quadrangle's Two-Round Algorithm (**QuaTR**), to address the triangle and quadrangle counting problems. In Section 6, to address the need to provide edge-LDP for data with added randomness in the first round during the second round, we adopt a method analogous to confidence intervals to ensure edge-LDP for perturbed data. In Section 7, we further conducted theoretical analyzes on the convergence of relative error for selected algorithms, proposed **2STAR**—a 2-star counting algorithm based on **Graph Projection**'s outputs, and theoretically compared all our algorithms with existing methods. In Section 8, we experimentally validate the theoretical performance of our algorithms and evaluate their practical utility.

Our Contribution. Our contributions are as follows:

- **Designing more effective algorithms.** In the field of private subgraph counting under edge-LDP, the algorithms proposed in this paper achieve the following milestones: TriOR stands out as the most accurate and fastest among one-round algorithms. TriTR surpasses all existing algorithms in terms of accuracy. TriMTR achieves the highest accuracy while maintaining acceptable communication costs. QuaTR represents the first algorithm designed for quadrangle counting under pure edge-LDP. 2STAR algorithm achieves highest accuracy in 2-star counting

- **Achieving DP on randomized data.** In the two-round algorithm, ensuring differential privacy protection on the randomized data is a challenge. To address this, we propose the second round's randomizer algorithm, which is inspired by confidence intervals, designed to achieve differential privacy protection with minimal loss in accuracy.

- **Adopting to more variants of DP.** The generation of the noisy adjacency matrix does not specify a particular DP mechanism. It can utilize RR, Laplace mechanism, Gaussian mechanism, or any other DP mechanism capable of producing unbiased estimates. Therefore, both traditional (ε, δ)-DP [16] and f-DP [12] can be used within this framework.

- **Applying to more types of graphs.** Existing subgraph counting under edge LDP is set in the context of undirected graphs, while all of our theorem guarantees can be transferred to directed graphs and weighted graphs with only minor modifications.

2 Related Work

Non-private Subgraph Counting. In a non-private setting [47], the methods for subgraph counting have been extensively studied. The subgraphs primarily include triangles [3, 5, 17, 30, 44, 48, 49, 52, 53], quadrangles (4-cycle) [5, 27, 36, 37], k-stars [2, 21], and k-hop paths [8, 28].

The primary issue in this field is how to efficiently compute the number of subgraphs. For example, in the domain of triangle counting, the exact number of triangles can be obtained by calculating the trace of the cubed adjacency matrix [3, 48] or by enumeration [10, 53]. Nevertheless, these two methods face the problem of high time complexity when dealing with large-scale networks. To reduce the time complexity, there are generally two approaches: one is to accelerate the computation through algorithm design, such as developing more efficient matrix multiplication algorithms to reduce the time complexity [11, 13, 31, 50, 51, 58]. The other is to speed up the subgraph counting by sacrificing some accuracy, for example,

by efficiently estimating the trace of the matrix [3], or by sampling vertices or edges to count a portion of the subgraphs, and then estimating the number of subgraphs in the entire network [17, 30, 52].

Private Subgraph Counting. Differential privacy has been widely applied in graph statistics. Categorized by the type of protected information, it can be divided into edge differential privacy (edge DP) [45, 55] and node differential privacy (node DP) [25, 29]. Based on the noise addition step, it can be classified into local differential privacy (LDP) [7, 41, 59] and central differential privacy (CDP) [6, 41]. Generally speaking, node DP provides stronger privacy protection than edge DP, and LDP offers more robust privacy safeguards than CDP. However, these stronger privacy protections come at the cost of increased estimation errors [14, 25, 29, 45, 55].

As a result, edge LDP has garnered more attention in recent years, and a series of papers have emerged that employ edge LDP to address the problem of subgraph counting: $\text{RR}_{\triangle}$ [18, 22], $\text{2R}_{\triangle}$ [22], $\text{ARR}_{\triangle}$ [23], $\text{2R-Small}_{\triangle}$ [23], $\text{2R-Large}_{\triangle}$ [23], $\text{Wshuffle}_{\triangle}$ [24], and the $\text{Wshuffle}_{\square}$ [24]. These algorithms are designed under the assumption that nodes have no prior knowledge of their neighbors' adjacency lists, a condition that closely mirrors real-world situations.

However, each of the existing algorithms faces distinct challenges. For example, the $\text{RR}_{\triangle}$ algorithm exhibits a time complexity of $O(n^3)$, rendering it impractical for large-scale networks [18, 22, 23]. The $\text{2R-Large}_{\triangle}$ algorithm, an improved version of $\text{2R}_{\triangle}$ through *double-clipping* [23], still faces substantial download cost overhead during the second round [23]. $\text{ARR}_{\triangle}$ and $\text{2R-Small}_{\triangle}$ mitigate time complexity and download cost via $edge-sampling$ [23], but they suffer from reduced accuracy [23, 24]. Furthermore, $\text{Wshuffle}_{\triangle}$ and $\text{Wshuffle}_{\square}$, which leverage the shuffle model [4, 9, 19, 20], a framework between CDP and LDP that employs a third-party shuffler to enhance privacy protection from ε_{Large}-LDP to $(\varepsilon_{Small}, \delta)$-DP by shuffling. If an adversary successfully compromises the shuffler, $\text{Wshuffle}_{\triangle}$ and $\text{Wshuffle}_{\square}$ can only provide ε_{Large}-LDP, where ε_{Large} is typically significantly larger than ε_{Small}, making this level of privacy protection unacceptable.

3 Preliminaries

3.1 Notations

Let $\mathbb{R}$, $\mathbb{R}_{\geqslant 0}$, $\mathbb{N}$, and $\mathbb{Z}_{\geqslant 0}$ denote the sets of real numbers, and non-negative real numbers, natural numbers, non-negative integers. Let $[n] = \{1, 2, ..., n\}, n \in \mathbb{N}$.

Let $\mathcal{G}$ denote the set of all undirected graphs without self-loops. For a graph $G \in \mathcal{G}$, let $G = (V, E)$, where $V = (v_i)$ denotes the set of nodes, $E \subseteq V \times V$ denotes the set of edges. Let n denote the number of nodes in G, and $|E|$ denote the number of undirected edges. If the nodes of G are labeled with indices, let d_i denote the degree of node v_i, and define $d_{\max} = \max(d_1, d_2, \ldots, d_n)$, $d_{\text{avg}} = \text{mean}(d_1, d_2, \ldots, d_n)$.

Let $A = (a_{ij}) \in 0, 1^{n \times n}$ denote the adjacency matrix corresponding to graph G. If $(v_i, v_j) \in E$, then $a_{ij} = 1$, otherwise $a_{ij} = 0$.

Let the i-th row of matrix A, denoted as $\mathbf{a}_i \in \{0, 1\}^n$, represent the adjacency list of node v_i. Let $B = (b_{ij}) = A^2$ and $C = (c_{ij}) = A^3$, which are referred to as the two-step and three-step matrices of graph G, respectively.

Let $f^{\triangle} : \mathcal{G} \to \mathbb{Z}_{\geqslant 0}$, $f^{\square} : \mathcal{G} \to \mathbb{Z}_{\geqslant 0}$ and $f^{\text{2-star}} : \mathcal{G} \to \mathbb{Z}_{\geqslant 0}$ be triangle, quadrangle and 2-star counting functions, respectively. They take graph $G \in \mathcal{G}$ as input and output the corresponding subgraph numbers $f^{\triangle}(G)$, $f^{\square}(G)$ and $f^{\text{2-star}}(G)$.

3.2 Differential Privacy

Definition of Differential Privacy. Differential privacy (DP) was first proposed by Dwork et al., who defined (ε, δ)-DP to measure privacy loss:

Definition 1 ((ε, δ)-DP [16])**.** *Let $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ be a randomized mechanism, where $\mathcal{X}^n$ is the space of datasets containing n data points, and $\mathcal{Y}$ is the output space. For any two neighboring datasets D and D' (i.e., D and D' differ in only one record), and for any subset of outputs $S \subseteq \mathcal{Y}$, if the mechanism $\mathcal{M}$ satisfies:*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta \tag{1}$$

then the mechanism $\mathcal{M}$ is said to satisfy (ε, δ)-differential privacy.

If $\delta = 0$, it is commonly abbreviated as ε-DP. ε is typically referred to as the privacy budget, A smaller ε indicates a stronger privacy protection.

In recent years, Dong et al. proposed a novel concept of DP from the perspective of hypothesis testing, termed f-Differential Privacy (f-DP) [12]. Gaussian Differential Privacy (GDP) emerges as the focal privacy definition within the f-DP framework, effectively characterizing the limiting behavior of privacy mechanisms under composition theorem.

Implementing Differential Privacy. There are numerous approaches to achieving DP [15, 16, 38, 56], in this paper, we employ the most commonly used mechanisms, RR and the Laplace mechanism, to implement DP.

Laplace Mechanism [14]. Let $f : \mathcal{X}^n \to \mathbb{R}^k$ be a function that maps a dataset $D \in \mathcal{X}^n$ to a vector of real numbers. The *global sensitivity* of f, denoted by Δf, is defined as:

$$\Delta f = \max_{D, D'} \|f(D) - f(D')\|_1 \tag{2}$$

where the maximum is taken over all pairs of neighboring datasets D and D' that differ in at most one entry. For a given privacy parameter $\varepsilon > 0$, the Laplace Mechanism releases:

$$\mathcal{M}_L(D, f, \varepsilon) = f(D) + (Y_1, Y_2, \ldots, Y_k) \tag{3}$$

where $Y_1, Y_2, \ldots, Y_k$ are independent and identically distributed (i.i.d.) random variables drawn from the Laplace distribution $\text{Lap}\left(\frac{\Delta f}{\varepsilon}\right)$.

Warner's RR [57]. Given $\varepsilon \in \mathbb{R}_{\geq 0}$, $\mathcal{R}_\varepsilon^W : \{0,1\} \to \{0,1\}$ maps $x \in \{0,1\}$ to $y \in \{0,1\}$ with the probablity:

$$\Pr\left[\mathcal{R}_\varepsilon^W(x) = y\right] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon+1} & (\text{if } x = y) \\ \frac{1}{e^\varepsilon+1} & (\text{otherwise}) \end{cases} \tag{4}$$

3.3 Local Differential Privacy on Graphs

We adopt the definition of ε-edge LDP as our privacy metric:

Definition 2 (ε-edge LDP [45].)**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. A local randomizer $\mathcal{R}$ with domain $\{0,1\}^n$ provides ε-edge LDP if for any two neighbor lists $\mathbf{a}_i, \mathbf{a}_i' \in \{0,1\}^n$ that differ in one bit and any $S \subseteq Range(\mathcal{R})$,*

$$\Pr[\mathcal{R}(\mathbf{a}_i) \in S] \leq e^\varepsilon \Pr[\mathcal{R}(\mathbf{a}_i') \in S]. \tag{5}$$

Interaction among Users and Multiple Rounds. It is common to provide users with auxiliary information through multiple rounds of queries to generate more accurate estimation. Therefore, we leverage the sequential composition of edge LDP to ensure privacy guarantees:

Proposition 1 (Sequential Composition of Edge LDP [23])**.** *For $i \in [n]$, let $\mathcal{R}_i^1$ be a local randomizer of user v_i that takes $\mathbf{a}_i$ as input. Let λ_i be a post-processing algorithm on $\mathcal{R}_i^1(\mathbf{a}_i)$, and $M_i = \lambda_i(\mathcal{R}_i^1(\mathbf{a}_i))$ be its output. Let $\mathcal{R}_i^2(M_i)$ be a local randomizer of v_i that depends on M_i. If $\mathcal{R}_i^1$ provides ε_1-edge LDP and for any $M_i \in \text{Range}(\lambda_i)$, $\mathcal{R}_i^2(M_i)$ provides ε_2-edge LDP, then the sequential composition $(\mathcal{R}_i^1(\mathbf{a}_i), \mathcal{R}_i^2(M_i)(\mathbf{a}_i))$ provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP.*

3.4 Utility Metrics

We evaluate the utility of the algorithm from two perspectives: accuracy and data transmission cost.

Accuracy. We introduce two metrics: Mean Squared Error (MSE) and Relative Error (RE). The MSE is defined as $\text{MSE} = \mathbb{E}\left[(\hat{\theta} - \theta)^2\right]$, where $\hat{\theta}$ represents the estimated value and θ denotes the true value. The RE is defined as $\text{RE} = \frac{|\hat{\theta} - \theta|}{\theta}$. While MSE is convenient for theoretical analysis, RE is often of greater practical interest in real-world applications.

Data transmission cost. Since the node transmission cost of existing subgraph counting algorithms under edge LDP is generally bounded by $O(n)$, this does not typically impose significant constraints on the practical application of such algorithms. The primary limitation arises from the substantial download cost(denoted as Cost_{DL}) incurred by nodes during multiple rounds of the algorithm.

Assuming there are n users and a total of r rounds of queries, let M_i^j denote the download volume of user i in the j-th round. We define the download cost as:

$$\text{Cost}_{DL} = \max_{i=1}^{n} \sum_{j=1}^{r} \mathbb{E}[|M_i^j|] \quad (\text{bits}). \tag{6}$$

4 Noisy Adjacency Matrix

This section introduces the Noisy Adjacency Matrix (NAM), which serves as the foundation for our algorithms. Section 4.1 defines NAM and analyzes its mathematical properties under power operations. Section 4.2 explains how to obtain NAM in real-world applications. Section 4.3 compares how different DP mechanisms affect NAM. Section 4.4 presents efficient computation methods for NAM.

4.1 Definition and Properties

We define the noisy adjacency matrix:

Definition 3 (Noisy adjacency matrix)**.** *$\hat{A}$ is the noisy adjacency matrix of undirected graph $G \in \mathcal{G}$, if $\hat{A}$ satisfies:*

$$\mathbb{E}\left[\hat{A}\right] = A;\ \hat{A} = \hat{A}^T;\ \hat{a}_{ii} = 0, \text{ for any } i \in [n];$$
$$\hat{a}_{ij} \perp \hat{a}_{kl}, \text{ for any } i < j,\ k < l,\ (i,j) \neq (k,l).$$

where A is the adjacency matrix of graph G, n is the number of nodes, $\perp$ denotes the independence between random variables.

For a graph $G \in \mathcal{G}$, the k-th power of its adjacency matrix A has the following property:

Proposition 2. *The value of the element in the i -th row and j -th column of the k -th power of adjacency matrix A equals the number of solutions for node i to reach node j exactly after k steps.*

Proposition 2 establishes the relationship between paths of varying lengths in a graph and its adjacency matrix. Through our research, we have discovered that the noisy adjacency matrix exhibits similar properties.

Theorem 1. *A and $\hat{A}$ be the adjacency matrix and the noisy adjacency matrix of graph G, respectively. Then:*

1. *Let $\hat{B} = \hat{A}^2$, $B = A^2$. $\mathbb{E}\left[\hat{b}_{ij}\right] = b_{ij}$, for any $i \neq j$.*
2. *Let $\hat{C} = \hat{A}^3$, $C = A^3$. $\mathbb{E}\left[\hat{c_{ii}}\right] = c_{ii}$, for any $i \in [n]$.*

Next, we will demonstrate how to integrate the noisy adjacency matrix with edge LDP, which will serve as the foundation for all subsequent algorithm designs.

4.2 Generating Noisy Adjacency Matrix

We designed the Generate Noisy Adjacency Matrix (GNAM) outlined in Algorithm 1. Initially, each user employs a *local randomizer* to apply LDP to their adjacency list. Subsequently,

the noisy adjacency relationships of nodes with a higher index are set to 0, and the final output $\tilde{\mathbf{a}}_u$ is sent to the *data collector*. The data collector then aggregates the $\tilde{\mathbf{a}}_u$, and completes the upper half of the matrix $\tilde{A}$ depending on the symmetry of the undirected graph. Finally, generating unbiased estimates for each element of the matrix $\tilde{A}$, resulting in the noisy adjacency matrix $\hat{A}$.

Algorithm 1 GNAM

Input: $\varepsilon \in \mathbb{R}_{\geq 0}$, graph G's adjacency list $\mathbf{a}_1$,$\mathbf{a}_2$,...,$\mathbf{a}_n \in \{0,1\}^n$.

Output: Noisy adjacency matrix of graph G.

1: **for** *each user* $u = 1$ to n **do:**
2: $\quad \tilde{\mathbf{a}}_u \leftarrow$ *local randomizer* $(\mathbf{a}_u, \varepsilon)$
3: $\quad \tilde{a}_{ui} \leftarrow 0$, *for any* $i \geqslant u$.
4: $\quad$ **send** $\tilde{\mathbf{a}}_u$ to *data collector*

Data collector do:

5: $L \leftarrow (\tilde{\mathbf{a}}_1, ..., \tilde{\mathbf{a}}_n)$
6: $\tilde{A} \leftarrow L + L^T$
7: $\hat{A} \leftarrow$ *estimate algorithm* $(\tilde{A})$
8: **return** $\hat{A}$

Because each user only uploads the perturbed data of the adjacency relationships with nodes that have a smaller index than their own, with the rest set to 0, the relationship between any two nodes in the graph is uploaded only once after being noise-added. So GNAM has the safe-guarantee promise:

Theorem 2. *GNAM satisfies* ε*-edge LDP.*

Any DP mechanism capable of producing unbiased estimates can be employed as the *local randomizer*. If one opts to adopt the f-DP framework, mechanisms such as adding Gaussian noise could be considered. However, since this paper utilizes the ε-edge LDP, we will focus on the most commonly used mechanisms, RR and Laplace mechanism, as examples.

Proposition 3. *Let X be the output of Warner's RR, i.e. $X = \mathcal{R}_{\varepsilon}^{W}(a)$, where $a \in \{0,1\}$, then $Y = \frac{X \cdot (e^{\varepsilon}+1)-1}{e^{\varepsilon}-1}$ satisfies $\mathbb{E}[Y] = a$.*

If choosing RR to provide DP, what we do in line 7 of Algorithm 1 is: if $\tilde{a}_{ij} = 1$, then set $\hat{a}_{ij} \leftarrow \frac{e^{\varepsilon}}{e^{\varepsilon}-1}$; if $\tilde{a}_{ij} = 0$, then set $\hat{a}_{ij} \leftarrow \frac{-1}{e^{\varepsilon}-1}$.

If the Laplace mechanism is chosen to provide edge-LDP, $\tilde{A}$ can be directly used as $\hat{A}$, since $\tilde{A}$ satisfies all the conditions required for a noisy adjacency matrix.

Although numerous DP mechanisms are available for selection, different choices can introduce subtle variations that may impact the performance of subsequent algorithms. In the following discussion, we will continue to use the RR and Laplace mechanism as examples to illustrate the effects of choosing different DP mechanisms.

4.3 Different Noise Analysis

Accuracy. Given ε and the chosen DP mechanism, the variance of each off-diagonal element in the noisy adjacency matrix is fixed and the same for all off-diagonal elements. Let σ^2 denote the variance of every single element in the noisy adjacency matrix. Under a fixed ε, the choice of DP mechanism affects the magnitude of σ^2, which will influence the accuracy of subsequent algorithms.

If employing RR, $\sigma^2 = \frac{e^{\varepsilon}}{(e^{\varepsilon}-1)^2}$. If employing the Laplace mechanism, $\sigma^2 = \frac{2}{\varepsilon^2}$. Figure 2 illustrates the comparative variance between the two mechanisms. The results demonstrate that RR achieves significantly lower σ^2 values than the Laplace Mechanism. RR's variance approximates half that of the Laplace Mechanism (as evidenced by the near-overlap between the blue dot-dash line and the red line), yielding $\sigma^2 = O(\frac{1}{\varepsilon^2})$ for RR. These findings indicate that given the same privacy budgets, RR provides superior estimation accuracy compared to the Laplace Mechanism.

Security. Although both provide ε-edge LDP, their actual protection effectiveness differs when analyzed from an attacker's perspective. We analyze the security comparison in two ways: hypothesis testing and confusion matrices.

To facilitate the discussion, we note the attack function: Let $x \in \{0,1\}$ denote the true value of the data, $\mathcal{R} : \{0,1\} \to \mathcal{X}$ denote the local randomizer in GNAM, $\tilde{x} = \mathcal{R}(x)$ and $f_{attack} : \text{Range}(\mathcal{R}) \to \{0,1\}$ be the attack function that takes the perturbed data $\tilde{x}$ as input and outputs a judgment $y \in \{0,1\}$.

Hypothesis testing. We set up the null hypothesis as $H_0 : x = 1$ and the alternative hypothesis as $H_1 : x = 0$. The probability of a Type I error is given by $\Pr(f_{\text{attack}}(\mathcal{R}(1)) = 0)$, and the probability of a Type II error is given by $\Pr(f_{\text{attack}}(\mathcal{R}(0)) = 1)$.

In attacks against the Laplace mechanism, given a threshold $\kappa \in \mathbb{R}$, if $\tilde{x} > \kappa$, then $f_{\text{attack}}(\tilde{x}) = 1$; otherwise, $f_{\text{attack}}(\tilde{x}) = 0$. For each distinct value of κ, the probabilities of Type I and Type II errors of the Laplace mechanism will vary; then we can plot the trade-off curve. The closer the curve is to both axes, the better the performance of the inference attack.

In attacks against the RR, $f_{attack}(\tilde{x}) = 1$ if $\tilde{x} = 1$, and 0 otherwise. We can draw the inflection point of RR's trade-off curve in Figure 3(a). To plot a whole trade-off curve for RR, we can make an adjustment by introducing a probability parameter $p_0 \in [0,1]$. For the lower portion of the curve, we randomly flip a p_0 proportion of the attack decisions from $f_{\text{attack}}(\tilde{x}) = 1$ to $f_{\text{attack}}(\tilde{x}) = 0$ while varying p_0 from 0 to 1; conversely, the upper portion is generated by flipping a p_0 proportion of decisions from $f_{\text{attack}}(\tilde{x}) = 0$ to $f_{\text{attack}}(\tilde{x}) = 1$ in the same manner, thereby completing the entire trade-off curve.

Figure 3 (a) provides the trade-off curve of RR and Laplace Mechanism when $\varepsilon = 1$. It can be observed that the red curve lies below the entire blue curve. This occurs because, for any given privacy budget ε, RR yields equal Type I and Type II

Table 1: Confusion Matrices for Different Attack Strategies and their Corresponding Precision and Recall.

	Attack Strategy for RR		1'st Attack Strategy for Laplace		2'ed Attack Strategy for Laplace	
	$f_{attack}(\tilde{x})=1$	$f_{attack}(\tilde{x})=0$	$f_{attack}(\tilde{x})=1$	$f_{attack}(\tilde{x})=0$	$f_{attack}(\tilde{x})=1$	$f_{attack}(\tilde{x})=0$
$x=1$	$p\frac{e^{\varepsilon}}{e^{\varepsilon}+1}$	$p\frac{1}{e^{\varepsilon}+1}$	$\frac{1}{2}p$	$\frac{1}{2}p$	$p\frac{2e^{0.5\varepsilon}-1}{2e^{0.5\varepsilon}}$	$p\frac{1}{2e^{0.5\varepsilon}}$
$x=0$	$(1-p)\frac{1}{e^{\varepsilon}+1}$	$(1-p)\frac{e^{\varepsilon}}{e^{\varepsilon}+1}$	$(1-p)\frac{1}{2e^{\varepsilon}}$	$(1-p)\frac{2e^{\varepsilon}-1}{2e^{\varepsilon}}$	$(1-p)\frac{1}{2e^{0.5\varepsilon}}$	$(1-p)\frac{2e^{0.5\varepsilon}-1}{2e^{0.5\varepsilon}}$
Precision	$P_1=\frac{pe^{\varepsilon}}{1-p+pe^{\varepsilon}}$		$P_2=\frac{pe^{\varepsilon}}{1-p+pe^{\varepsilon}}$		$P_3=\frac{2pe^{0.5\varepsilon}-p}{1-2p+2pe^{0.5\varepsilon}}$	
Recall	$R_1=\frac{e^{\varepsilon}}{e^{\varepsilon}+1}$		$R_2=\frac{1}{2}$		$R_3=\frac{2e^{0.5\varepsilon}-1}{2e^{0.5\varepsilon}}$	

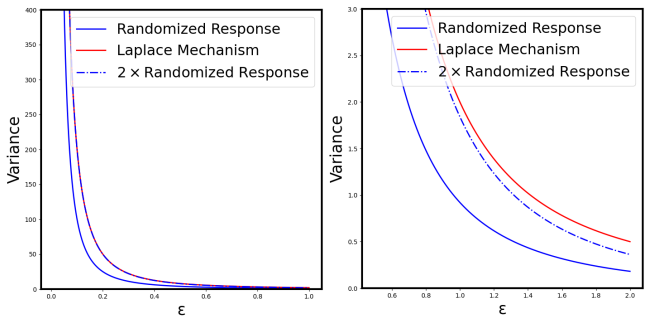

Figure 2: Comparing Variance between RR and Laplace Mechanism. The red line represents the variance of unit position elements in NAM when using the Laplace mechanism, while the solid and dashed blue lines correspond to the variance obtained with the Laplace mechanism and the curve obtained by doubling the variance, respectively.

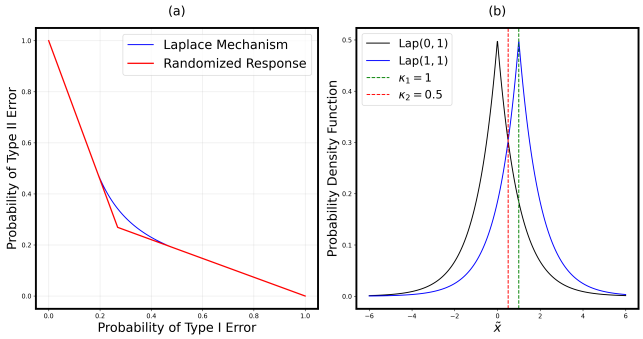

Figure 3: (a) Trade-off Curves of the Laplace Mechanism and RR ($\varepsilon=1$). (b) The First and Second Attack Strategies on the Laplace Mechanism ($\varepsilon=1$). In (b), κ_1 and κ_2 represent the thresholds selected for the two Attack Strategies, respectively.

error probabilities of $1/(1+e^{\varepsilon})$. In contrast, for the Laplace mechanism, when the total error probability is minimized (achieved at $\kappa=0.5$, corresponding to the red line in Figure 3(b)), both error probabilities become $1/(2e^{0.5\varepsilon})$ - a value strictly greater than $1/(1+e^{\varepsilon})$. Therefore, it can be concluded that compared to the Laplace mechanism, RR demonstrates superior attack effectiveness when subjected to adversarial attempts, consequently offering weaker privacy protection.

Confusion matrix. We employ two attack strategies for the Laplace mechanism. The first strategy uses $\kappa_1=1$ as the threshold (corresponding to the green line in Figure 3(b)), while the second strategy uses $\kappa_2=0.5$ (corresponding to the red line in Figure 3(b)). The first strategy achieves the minimum recall while maintaining the maximum precision. The second strategy, mentioned in hypothesis testing, identifies the threshold that minimizes the sum of Type I and Type II errors. For RR, we employ the most basic attack strategy, which does not involve any post-decision flipping operations.

Given the same privacy budget ε and the density of edges in the graph $p=\frac{2|E|}{n*(n-1)}$, the confusion matrix and their corresponding precision and recall for the three attack strategies are shown in Table 1. For clearer comparison of Precision versus Recall, we visualize the Table 1 results in Figure 4.

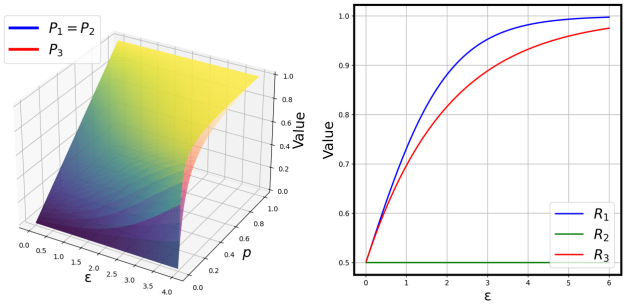

Figure 4: Comparing Precision and Recall. P_1-P_3 and R_1-R_3 in the figures are from Table 1.

Combining Table 1 and Figure 4: comparing RR and the first attack of Laplace mechanism, RR has a higher recall while maintaining the same precision. Comparing RR and the second attack of Laplace mechanism, RR has both higher precision and recall. Therefore, the attack effect for RR is stronger compared to the Laplace mechanism, indicating that the Laplace mechanism provides better protection than RR.

4.4 Acceleration of Noisy Adjacency Matrix

The primary drawback of all existing one-round algorithms is that the time complexity during computation is $O(n^3)$ [18, 22, 23], which makes it challenging to handle counting problems in large-scale graph scenarios. We can significantly reduce the high time complexity by leveraging faster matrix multiplication in our algorithms.

According to research on matrix multiplication [11, 13, 31, 50, 51, 58], the complexity has been reduced from $O(n^3)$ to $O(n^{2.371866})$ [13]. Although this may seem like a small difference, it becomes particularly significant as n increases. For example, when n is 10^5, their running times will differ by a factor of $10^{5\times(3-2.371866))} = 10^{3.14067}$, which is approximately 1000 times. We will also verify this in our experiment section.

5 Subgraph Counting

In this section, we will design four algorithms based on the properties of the noisy adjacency matrix. Among these, the latter three algorithms achieve more accurate estimations through multiple rounds of queries. However, in second round they face the challenge of providing DP for the data perturbed by GNAM. To address this, we employ a unified idea and method in the next section. Therefore, we use the *second randomizer* serves as a placeholder, the specific implementations for *second randomizer* are detailed in the next section. And we only provide accuracy guarantees without the bias caused by the *second randomizer* in this section.

5.1 Triangle's One-Round Algorithm

According to Proposition 1, it can be deduced that the elements on the diagonal of the cube of the adjacency matrix A represent the number of ways a node can return to itself in exactly three steps. Since we assume there are no self-loops in the graph, any path that returns to the starting node in three steps must form a triangle. For each triangle, each node traverses the triangle in two directions, hence the total number of triangles $f^{\triangle}(G) = \mathrm{tr}\left(A^3\right)/(2\cdot 3)$.

Algorithm 2 TriOR

Input: $\varepsilon \in \mathbb{R}_{\geq 0}$, graph G .
Output: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
1: $\hat{A} \leftarrow \mathrm{GNAM}(G, \varepsilon)$
2: **return** $\mathrm{tr}\left(\hat{A}^3\right)/6$

By the second property of Theorem 1, we can conclude that $\mathbb{E}\left[\mathrm{tr}\left(\hat{A}^3\right)\right] = \mathrm{tr}\left(A^3\right)$. Consequently, by directly computing $\hat{f}^{\triangle}(G) = \mathrm{tr}\left(\hat{A}^3\right)/6$, we can obtain an unbiased estimate of the number of triangles in the graph. We denote this algorithm as Triangle's One-Round Algorithm (TriOR), because there is only one interaction between users and data collector in GNAM. Algorithm 2 presents the detailed implementation of TriOR. Then we show its security guarantees and accuracy bounds:

Theorem 3. *TriOR provides ε-edge LDP.*

Theorem 4. *TriOR provides an unbiased estimate of $f^{\triangle}(G)$, and its MSE is:*

$$\sigma^2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} b_{ij}^2 + \sigma^4 (n-2)|E| + \frac{1}{6}\sigma^6 n(n-1)(n-2) \quad (7)$$

And it satisfies: $MSE \leqslant O\left(nd_{\max}^3 + n^3\right)$.

5.2 Triangle's Two-Round Algorithm

If a user v_u knows the existence of edges between their neighbors, they can count the number of triangles formed by themselves and their neighbors (denoted as $f_u^{\triangle}$). However, the presence or absence of edges among neighbors is private information, which is not accessible to the user. However, users can estimate edge connections between neighbors based on the noisy adjacency matrix, thus inferring the missing third edge in triangles and subsequently estimating $f_u^{\triangle}$.

Algorithm 3 shows the Triangle's Two-Round Algorithm (TriTR). In the first round, the data collector obtains a noisy adjacency matrix through GNAM. Subsequently, in the second round, each user downloads the whole noisy adjacency matrix $\hat{A}$. Each user obtains sum_u by summing the values between their neighbors in $\hat{A}$. sum_u satisfies $\mathbb{E}[sum_u] = 2f_u^{\triangle}$, so $\mathbb{E}[\frac{1}{6}\sum_{i=u}^{n} sum_u] = f^{\triangle}(G)$. To guarantee ε_2-edge LDP in the second round, each user process their sum_u to derive $\hat{T}_u$ which is then transmitted to the data collector. The collector subsequently sums all $\hat{T}_u$ and divides the total by six to obtain the final triangle count estimate $\hat{f}^{\triangle}(G)$. If the *second randomizer* in line 6 provides ε_2-edge LDP, we have:

Algorithm 3 TriTR

Input: Graph G, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$.
Output: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
#First round:
1: $\hat{A} \leftarrow \mathrm{GNAM}(G, \varepsilon_1)$
#Second round:
2: **for each** *user* $u = 1$ to n **do:**
3: **download** $\hat{A}$
4: $sum_u \leftarrow \sum_{(i,j):a_{ui}=a_{uj}=1} \hat{a}_{ij}$
5: $\hat{T}_u \leftarrow$ *second randomizer* (sum_u, ε_2)
6: **upload** $\hat{T}_u$ to *data collector*
data collector **do:**
7: $\hat{f}^{\triangle}(G) \leftarrow \frac{1}{6}\sum_{i=u}^{n} \hat{T}_u$
8: **return** $\hat{f}^{\triangle}(G)$

Theorem 5. *TriTR provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP.*

Theorem 6. *$\frac{1}{6}\sum_{i=u}^{n} sum_u$ provides an unbiased estimate of $f^{\triangle}(G)$, and its MSE is:*

$$\frac{1}{9}\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2. \tag{8}$$

And it satisfies: MSE $\leqslant O\left(nd_{\max}^3\right)$.

5.3 Triangle's Modified Two-Round Algorithm

TriTR has achieved a significant improvement in accuracy compared to TriOR. However, it faces the challenge of substantial download costs in the second round, where each user is required to download the entire $\hat{A}$. To address this issue, we adopt a trade-off approach by increasing the number of noisy edges in each triangle from one to two. This strategy sacrifices a little on accuracy (as shown in Section 7, they achieve similar relative error performance), while significantly reducing the download cost (from the entire matrix to just one column).

Algorithm 4 TriMTR

Input: Graph G, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$.
Output: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
#First round:
1: $\hat{A} \leftarrow \text{GNAM}(G, \varepsilon_1)$
2: *Data collector* **calculate:** $\hat{B} \leftarrow \hat{A}^2$
#Second round:
3: **for each node** $u = 1$ to n **do:**
4: **download** the u-th column of $\hat{B}$
5: $sum_u \leftarrow \sum_{i:a_{ui}=1} \hat{b}_{iu}$
6: $\hat{T}_u \leftarrow$ *second randomizer*(sum_u, ε_2)
7: **upload** $\hat{T}_u$ to *data collector*
Data collector do:
8: $\hat{f}^{\triangle}(G) \leftarrow \frac{1}{6}\sum_{u=1}^{n} T_u$
9: **return** $\hat{f}^{\triangle}(G)$

Algorithm 4 shows the Triangle's Modified Two-Round Algorithm (TriMTR). The data collector computes the squared matrix $\hat{B} = \hat{A}^2$ in the first round. In the second round, each user v_u downloads the u-th column of $\hat{B}$, sums the neighbor-associated entries in sum_u, applies privacy protection to generate $\hat{T}_u$, and uploads it to the data collector. According to the first property of Theorem 1, $\mathbb{E}[\hat{b}_{iu}] = b_{iu}$. Therefore, if $a_{ui} = 1$, then $\mathbb{E}[a_{ui}\hat{b}_{iu}] = a_{ui}b_{iu}$, which represents the number of triangles formed by the edge (v_u, v_i). And thus $\mathbb{E}[sum_u] = 2f_u^{\triangle}$. Consequently, $\mathbb{E}\left[\frac{1}{6}\sum_{u=1}^{n} sum_u\right] = f^{\triangle}(G)$. TriTR holds the following guarantees:

Theorem 7. *TriMTR provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP.*

Theorem 8. *$\frac{1}{6}\sum_{u=1}^{n} sum_u$ provides an unbiased estimate of $f^{\triangle}(G)$, and its MSE is:*

$$\frac{4}{9}\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2 + \frac{1}{9}\sigma^4(n-2)|E|. \tag{9}$$

And it satisfies: MSE $\leqslant O\left(nd_{\max}^3 + n^2 d_{\max}\right)$.

5.4 Quadrangle's Two-Round Algorithm

User v_u provides the two edges connecting themselves and their neighbors, and then utilizes $\hat{B} = \hat{A}^2$ to derive the noisy two-step relationships among their neighbors. This enables the estimation of the number of quadrangles they are part of (denoted as $f_u^{\square}$). Following this rationale, we designed Quadrangle's Two-Round Algorithm (QuaTR) shown in Algorithm 5.

Algorithm 5 QuaTR

Input: Graph G, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$.
Output: Estimate $\hat{f}^{\square}(G)$ of $f^{\square}(G)$.
#First round:
1: $\hat{A} \leftarrow \text{GNAM}(G, \varepsilon_1)$
2: *Data collector* **calculate:** $\hat{B} \leftarrow \hat{A}^2$
#Second round:
3: **for each node** $u = 1$ to n **do:**
4: **download** $\hat{B}$
5: $sum_u \leftarrow \sum_{(i,j):a_{ui}=a_{uj}=1} \left(\hat{b}_{ij} - 1\right)$
6: $\hat{Q}_u \leftarrow$ *second randomizer*(sum_u, ε_2)
7: **upload** $\hat{Q}_u$ to *data collector*
Data collector do:
8: $\hat{f}^{\square}(G) \leftarrow \frac{1}{8}\sum_{u=1}^{n} \hat{Q}_u$
9: **return** $\hat{f}^{\square}(G)$

The implementation of QuaTR is similar to that of TriTR, with modifications only in lines 4, 5 and 8. The change in line 5 is necessary because user v_u provides two edges $v_u \to v_i$ and $v_j \to v_u$, while the remaining paths of the form $v_i \to v_k \to v_j$ are provided by $\hat{b}_{ij}$. Since $\mathbb{E}[\hat{b}_{ij}] = b_{ij}$ includes the path $v_i \to v_u \to v_j$, which cannot form a quadrangle with $v_u \to v_i$ and $v_j \to v_u$, it is necessary to minus 1 to exclude this invalid path. Since each of the four vertices of a quadrangle will traverse the quadrangle from two distinct directions, we need to change line 8 by dividing by 8. Similarly, we have the following guarantees:

Theorem 9. *QuaTR provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP.*

Theorem 10. *$\frac{1}{8}\sum_{u=1}^{n} sum_u$ provides an unbiased estimate of $f^{\square}(G)$, and its MSE is:*

$$\frac{1}{4}\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}^2\sigma^2 + \frac{1}{16}(n-2)\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2\sigma^4. \tag{10}$$

And it satisfies: MSE $\leqslant O\left(nd_{\max}^5 + n^2 d_{\max}^3\right)$.

5.5 Subgraph Algorithms and Matrix Powers

The intuitive ideas behind the design of these algorithms are illustrated in Figure 5. The black edges represent edges within

the real graph, while the red edges denote noisy edges in $\hat{A}$. The powers of the noisy adjacency matrix correspond to the number of noisy edges within paths of a given length.

Moreover, the variance of each algorithm can be expressed in terms of the powers of the adjacency matrix, which further reveals a profound connection between the design of the algorithm and the matrix. The Frobenius Norm of a matrix A is:

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \tag{11}$$

In this analysis, we focus solely on the variance terms presented in Section 5. By defining the 0-th power of any matrix $M \in \mathbb{R}^{n \times n}$ as the identity matrix I_n and ignoring coefficients, we observe the following patterns:

$$\mathbb{V}(\text{TriTR}) = O(\sigma^2 \|A^2\|_F^2) \tag{12}$$

$$\mathbb{V}(\text{TriMTR}) = O(\sigma^2 \|A^2\|_F^2 + n\sigma^4 \|A^1\|_F^2) \tag{13}$$

$$\mathbb{V}(\text{TriOR}) = O(\sigma^2 \|A^2\|_F^2 + n\sigma^4 \|A^1\|_F^2 + n^2\sigma^6 \|A^0\|_F^2) \tag{14}$$

$$\mathbb{V}(\text{QuaTR}) = O(\sigma^2 \|A^3\|_F^2 + n\sigma^4 \|A^2\|_F^2) \tag{15}$$

6 Second Round's Randomizer

The two-round algorithms need to provide edge LDP for the data randomized in the first round, which makes it challenging to determine Δf and thus prevents directly using the Laplace mechanism to achieve ε_2-edge LDP. In this section, we present our solution to this problem, including the key ideas and implementation details.

6.1 Differential Privacy on Randomized Data

Key idea. Due to the randomness of Δf, we can utilize the distribution of Δf to determine its β quantile. This approach allows us to constrain Δf within a smaller range, thereby providing ε_2-edge LDP.

We employ the clamp function to bound Δf. Let κ denote the bound set for Δf. We set κ equal to the larger absolute value between the upper and lower β quantiles of Δf. Then, using clamp function: $\text{clamp}(\Delta f, \kappa) = \max(\min(\Delta f, +\kappa), -\kappa)$ to restrict the change. After clamping, the $\Delta f = \kappa$, then we can provide ε_2-edge LDP by Laplace mechanism.

If β is too small, it can reduce the error introduced by clamping but may result in a larger $\Delta f = \kappa$, thereby introducing excessive noise in the second round. Conversely, if β is too large, the error caused by clamp operation will increase. Therefore, it is crucial to select an appropriate β. Then the key challenge is to determine the appropriate Δf based on β.

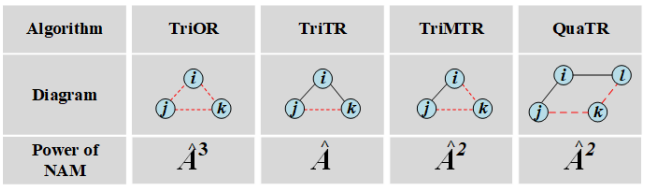

Figure 5: Over Review of Triangle and Quadrangle Algorithms. The intuitive form of Algorithms and the corresponding powers of noisy adjacency matrix.

6.2 Implementation

The majority of real networks holds $d_{\max} \gg d_{\text{avg}}$, if all nodes employ the same Δf, this would result in excessive noise being added to the majority of nodes. To address this issue, for each node v_u, we utilize its noisy degree value $\tilde{d}_u$ to provide a more tailored and tighter bound Δf_u. We denote the privacy budget allocated for obtaining $\tilde{d}_u$ as ε_0. According to Proposition 1, this "customized" assignment of Δf_u to each user still ensures that the overall system provides $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$ edge LDP.

We use GraphProjection (showed in Algorithm 6) to obtain noisy degrees $\tilde{d}_u$. To prevent an excessive number of edges from being removed in the graph, we introduce a parameter α to $\tilde{d}_i$. This ensures that edges of nodes with $d_i \leq \alpha$ remain intact, while nodes with $d_i \geq \alpha$ have a probability of $\frac{1}{2}e^{-\frac{\alpha}{\varepsilon_0}}$ for edge removal.

Algorithm 6 GraphProjection

Input: Adjacency list $\mathbf{a}_i$, degree d_i, ε_0, α.
Output: processed adjacency list $\mathbf{a}_i'$, noisy degree $\tilde{d}_i$.

1: $\tilde{d}_i = \lfloor \alpha + \max\left\{ d_i + \text{Lap}\left(\frac{1}{\varepsilon_0}\right), 0 \right\} \rfloor$
2: Remove $d_i - \tilde{d}_i$ neighbors randomly if $\tilde{d}_i < d_i$ to get $\mathbf{a}_i'$.
3: **return** $\mathbf{a}_i'$, $\tilde{d}_i$

Let Nei_u be the index set of the user v_u's neighbor. $\Phi^{-1} : [0,1] \to \mathbb{R}$ is the inverse distribution function of the standard normal distribution. Then using Algorithm 7 replaces lines 4-5 in TriTR, Algorithm 8 replaces lines 5-6 in TriMTR, and Algorithm 9 replaces lines 5-6 in QuaTR. Each user v_u, bounds the change contributed by any one of v_u's neighbors to sum_u within $(-\Delta f, +\Delta f)$. The sum_u's difference between neighbor lists $\mathbf{a}_u$ and $\mathbf{a}_u'$ is kept below Δf. Consequently, adding $\text{Lap}(\Delta f / \varepsilon_2)$ noise achieves ε_2-edge LDP. Let λ_u denote the first input of each clamp function. Algorithms 7-9 possess the following probabilistic guarantee to constrain the impact of the clamp function on the final accuracy.

Theorem 11. *It can be approximated that:*

$$\Pr[\lambda_u > \Delta f_u] < \beta \tag{16}$$

$$\Pr[\lambda_u < -\Delta f_u] < \beta \tag{17}$$

Algorithm 7 TriTR's Second Randomizer

Input: Noisy adjacency matrix $\hat{A}$; user u's noisy degree $\tilde{d}_u$, and it's neighbor set Nei_u; ε_2.
Output: $\hat{T}_u$.

$\Delta f_u \leftarrow \Phi^{-1}(1-\beta)\cdot\sqrt{\tilde{d}_u\sigma^2}+\tilde{d}_u$
$sum_u \leftarrow \sum_{i\in \mathrm{Nei}_u} \mathrm{clamp}(\sum_{j\in \mathrm{Nei}_u, j<i}\hat{a}_{ij}, \Delta f_u)$
$\hat{T}_u \leftarrow 2(sum_u + \mathrm{Lap}(\Delta f_u/\varepsilon_2))$
return $\hat{T}_u$

Algorithm 8 TriMTR's Second Randomizer

Input: Noisy two-step matrix $\hat{B}$'s u-th column $\hat{\mathbf{b}_u}$; user u's noisy degree $\tilde{d}_u$, and it's neighbor set N_u; ε_2.
Output: $\hat{T}_u$.

$\Delta f_u \leftarrow \Phi^{-1}(1-\beta)\cdot\sqrt{(n-2)\sigma^4+\left(\tilde{d}_u+\tilde{d}_{\max}\right)\sigma^2}+\tilde{d}_u$
$sum_u \leftarrow \sum_{i\in N} \mathrm{clamp}(\hat{b}_{iu}, \Delta f_u)$
$\hat{T}_u \leftarrow sum_u + \mathrm{Lap}(\Delta f_u/\varepsilon_2)$
return $\hat{T}_u$

Algorithm 9 QuaTR's Second Randomizer

Input: Noisy two-step matrix $\hat{B}$; user u's noisy degree $\tilde{d}_u$, and it's neighbor set N_u; ε_2.
Output: $\hat{Q}_u$.

$\Delta f_u \leftarrow \Phi^{-1}(1-\beta)\cdot\sqrt{\tilde{d}_u(2d_{\max}\sigma^2+(n-2)\sigma^4)}+\tilde{d}_u(d_{\max}-1)$
$sum_u \leftarrow \sum_{i\in \mathrm{Nei}_u} \mathrm{clamp}(\sum_{j\in \mathrm{Nei}_u, j<i}\left(\hat{b}_{ij}-1\right), \Delta f_u)$
$\hat{Q}_u \leftarrow 2(sum_u + \mathrm{Lap}(\Delta f_u/\varepsilon_2))$
return $\hat{Q}_u$

Because we use the Central Limit Theorem in the proof of Theorem 11, the term "approximate" appears in the theorem. Next, we present the safety and accuracy guarantees for each two-round algorithm.

Theorem 12. *TriTR provides $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP, and if we use RR or Laplace mechanism in GNAM, the total MSE satisfies:*

$$MSE \leqslant O\left(\frac{nd_{\max}^3}{\varepsilon_1^2}+\frac{|E|}{\varepsilon_1^2\varepsilon_2^2}+n\frac{d_{\max}^2}{\varepsilon_2^2}\right) \tag{18}$$

Theorem 13. *TriMTR provides $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP, and if we use RR or Laplace mechanism in GNAM, the total MSE satisfies:*

$$MSE \leqslant O\left(\frac{nd_{\max}^3}{\varepsilon_1^2}+\frac{n^2d_{\max}}{\varepsilon_1^4}+\frac{n^2}{\varepsilon_1^4\varepsilon_2^2}+n\frac{d_{\max}^2}{\varepsilon_2^2}\right) \tag{19}$$

Theorem 14. *QuaTR provides $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP, and if we use RR or Laplace mechanism in GNAM, the total MSE satisfies:*

$$MSE \leqslant O\left(\frac{nd_{\max}^5}{\varepsilon_1^2}+\frac{n^2d_{\max}^3}{\varepsilon_1^4}+\frac{n^2d_{\max}}{\varepsilon_1^4\varepsilon_2^2}+n\frac{d_{\max}^4}{\varepsilon_2^2}\right) \tag{20}$$

7 Theoretical Analysis

In this section, we theoretically analyze the proposed algorithms to guide the experimental evaluation. Section 7.1 shows the convergence properties on relative error for both two-round triangle counting algorithms. Section 7.2 develops and analyzes a noisy degree-based estimation algorithm for 2-stars, complete with theoretical performance guarantees. Section 7.3 presents a theoretical comparison between our proposed algorithms and existing algorithms in the literature.

7.1 Convergence on Relative Err

The specific expression of MSE has already been provided in the previous sections, so we now focus on RE.

To analyze the Relative Error, we will employ the **Jensen's Inequality**: $\mathbb{E}\left[|X-\mathbb{E}[X]|\right] \leqslant \sqrt{\mathbb{V}(X)}$. In this way, we can bounded the Absolute Error (ABE). Next, dividing ABE by $f^{\triangle}(G)$, we can bound RE.

Because the graph may not have any triangles, such as in a bipartite graph, which would make RE meaningless. Therefore, we need to make some constraints on the graph structure. Assuming that the *clustering coefficient* (equals to $\frac{3\times\#\text{triangles}}{\#\text{2-stars}}$) exists, the relative error of TriTR and TriMTR satisfies:

Theorem 15. *If the clustering coefficient of the graph exists, the RE of TriTR satisfies:*

$$RE \leqslant O\left(\frac{1}{\varepsilon_1 d_{avg}}+\frac{1}{\varepsilon_1\varepsilon_2\sqrt{n}d_{avg}^{\frac{3}{2}}}+\frac{1}{\varepsilon_2\sqrt{n}d_{avg}}\right); \tag{21}$$

The RE of TriMTR satisfies:

$$RE \leqslant O\left(\frac{1}{\varepsilon_1 d_{avg}}+\frac{1}{\varepsilon_1^2 d_{avg}^{\frac{3}{2}}}+\frac{1}{\varepsilon_1^2\varepsilon_2 d_{avg}^2}+\frac{1}{\varepsilon_2\sqrt{n}d_{avg}}\right). \tag{22}$$

According to Theorem 15, we can conclude that the denser the network (larger d_{avg}), the smaller the relative error. This also draws our attention to the fact that, given an overall privacy budget $\varepsilon=\varepsilon_0+\varepsilon_1+\varepsilon_2$, appropriately allocating more to ε_1 can reduce the overall error. Additionally, the accuracy of TriTR and TriMTR slightly improves as the graph size increases.

Since if all the privacy budget is considered as a constant, both TriTR and TriMTR satisfy $\mathrm{RE} \leq O\left(\frac{1}{d_{\mathrm{avg}}}\right)$, we can conclude that the performance of these two algorithms in relative err is not significantly different. Therefore, it can be seen that

Table 2: Theoretical Comparison of Private Subgraph Counting Algorithms.

Subgraph	Algorithm	Model	Variance	$\text{Time}_{\text{user}}$	$\text{Time}_{\text{data collector}}$	Cost_{DL}
Triangle	**TriOR**	one-round	$O(nd_{\max}^3 + n^3)$	$O(n)$	$O(n^{2.371866})$	0
Triangle	$\text{RR}_{\triangle}$ [22]	one-round	$O(n^4)$	$O(n)$	$O(n^3)$	0
Triangle	$\text{ARR}_{\triangle}$ [23]	one-round	$O(n^6)$	$O(n)$	$O(n^2)$	0
Triangle	**TriTR**	two-rounds	$O(nd_{\max}^3)$	$O(n + d_{\max}^2)$	$O(n^2)$	$O(n^2)$
Triangle	$\text{2R-Large}_{\triangle}$ [23]	two-rounds	$O(nd_{\max}^3)$	$O(n + d_{\max}^2)$	$O(n^2)$	$O(n^2)$
Triangle	**TriMTR**	two-rounds	$O(nd_{\max}^3 + n^2 d_{\max})$	$O(n)$	$O(n^{2.371866})$	$O(n)$
Triangle	$\text{2R-Small}_{\triangle}$ [23]	two-rounds	$O(n^2 d_{\max}^3)$	$O(n)$	$O(n^2)$	$O(n)$
Triangle	$\text{Wshuffle}_{\triangle}$ [24]	shuffle	$O(n^3 d_{\max}^2)$	$O(n)$	$O(n^2)$	0
Quadrangle	**QuaTR**	two-rounds	$O(nd_{\max}^5 + n^2 d_{\max}^3)$	$O(n + d_{\max}^2)$	$O(n^{2.371866})$	$O(n^2)$
Quadrangle	$\text{Wshuffle}_{\square}$ [24]	shuffle	$O(n^3 d_{\max}^2 + n^2 d_{\max}^6)$	$O(n)$	$O(n^2)$	0
2-star	**2STAR**	one-round	$O(\sum_{i=1}^n d_i^2)$	$O(1)$	$O(n)$	0
2-star	$\text{LocalLap}_{2\star}$ [22]	one-round	$O(nd_{\max}^2)$	$O(1)$	$O(n)$	0

TriMTR achieves a reduction in download cost by a factor of $\frac{1}{n}$ at the cost of only a slight decrease in accuracy, which should be considered a good trade-off in real-world applications.

7.2 Two-Star Counting

Through GraphProjection, we obtained $\tilde{d}_i$, which not only serves to mitigate the errors introduced in the second round but also provides the count of 2-stars within the graph. This can be specifically achieved by implementing Algorithm 10.

Algorithm 10 2STAR

Input: Adjacency list $\mathbf{a}_u$, degree d_u, ε_0, α.
Output: estimation of 2-stars in graph G.

1: **for each node** $u = 1$ to n **do:**
2: $\quad \tilde{d}_u = \text{GraphProjection}(\mathbf{a}_u, d_u, \varepsilon_0, \alpha)$
3: $\quad$ **upload** $\hat{d}_u$ to **data collector**

data collector **do:**
4: $\hat{f}^{\text{2-star}}(G) \leftarrow \sum_{u=1}^n \left[(\tilde{d}_u - \alpha)(\tilde{d}_u - \alpha - 1) - 2/\varepsilon_0^2\right]$
5: **return** $\hat{f}^{\text{2-star}}(G)$

We cannot provide its exact theoretical MSE, but we can explain the inspiration behind this algorithm and offer an approximate theoretical guarantee:

Theorem 16. *Let $\hat{d}_u = d_u + Lap(1/\varepsilon_0)$, $f^{\text{2-star}}(G)$ denote the true number of 2-star in graph G, then the estimator:*

$$\hat{f}^{\text{2-star}}(G) \leftarrow \sum_{u=1}^{n} \left[\hat{d}_u(\hat{d}_u - 1) - 2/\varepsilon_0^2\right] \tag{23}$$

has the following properties:

$$\mathbb{E}[\hat{f}^{\text{2-star}}(G)] = f^{\text{2-star}}(G) \tag{24}$$

$$MSE(\hat{f}^{\text{2-star}}(G)) = \frac{8}{\varepsilon_0^2}\sum_{u=1}^{n} d_u^2 - \frac{16}{\varepsilon_0^2}|E| + \frac{2}{\varepsilon_0^2}n + \frac{20}{\varepsilon_0^4}n \tag{25}$$

$$RE \leqslant O\left(\frac{1}{\sqrt{\#\text{2-star}}}\right) \tag{26}$$

Algorithm 10 does not align fully with the above theorem because the GraphProjection step requires ensuring $\tilde{d}_u \geq \alpha$. However, the discrepancy should be minimal. This is because the majority of 2-stars in the graph are contributed by nodes with higher degrees, and for these nodes, $\tilde{d}_u - \alpha$ approximately equals $\hat{d}_u$.

This implies that in practical implementations, the two-round algorithm operates as follows: In the first round, each user applies GraphProjection to obtain $\tilde{d}_u$ then applies GNAM to generate $\tilde{\mathbf{a}}_u$, both $\tilde{d}_u$ and $\tilde{\mathbf{a}}_u$ are uploaded to the data collector. This allows the collector to simultaneously derive the 2-star count estimate before proceeding with the second round for triangle or quadrangle estimation. If downloading the entire matrix is acceptable in the second round, users can download $\hat{B}$ and upload $\hat{T}_u$ and $\hat{Q}_u$ concurrently, allowing simultaneous estimation of triangle, quadrangle, and 2-star counts.

7.3 Theoretical Comparison

Table 2 presents the state-of-the-art subgraph counting algorithms under edge-LDP: $\text{RR}_{\triangle}$ [22], $\text{2R-Large}_{\triangle}$ [23], $\text{ARR}_{\triangle}$ [23], $\text{2R-Small}_{\triangle}$ [23], $\text{Wshuffle}_{\triangle}$ [24], $\text{Wshuffle}_{\square}$ [24], $\text{LocalLap}_{2\star}$ [22] and our algorithms: TriOR, TriTR, TriMTR, QuaTR, 2STAR.

$\text{ARR}_{\triangle}$ with a sampling probability $p_0 = O(n^{-\frac{1}{3}})$ in edge-sampling, the time complexity can be reduced to $O(n^2)$. $\text{2R-Small}_{\triangle}$ (which is $\text{ARROneNS}_{\triangle}$ in [23]) with a sampling probability $p_0 = \frac{e^{\varepsilon}+1}{e^{\varepsilon}\sqrt{n}}$, the Cost_{DL} can be reduced to $O(n \log n)$ [24]. It should be noted that the term $\log(n)$ in Cost_{DL} arises from representing the node indices in binary form, since the n nodes require $\log(n)$ bits. The transmission of our algorithms uses floating-point numbers. If we treat the number of bits required for floating-point numbers and for

representing n nodes as constants, then the Cost_{DL} for both 2R-Small$_\triangle$ and TriMTR is $O(n)$.

Among one-round algorithms, TriOR has achieved an enhancement in accuracy while reducing the time complexity. Although the theoretical analysis of TriTR does not indicate improvement in precision, experiments in next section demonstrate that TriTR has attained the highest accuracy among all algorithms. This discrepancy occurs because TriTR introduces less cumulative noise in the second round compared to 2R-Large$_\triangle$. When the Cost_{DL} is O(n), TriMTR exhibits superior precision compared to 2R-Small$_\triangle$. QuaTR surpasses Wshuffle$_\square$ in terms of accuracy. The 2STAR provides higher accuracy than LocalLap$_{2\star}$ since LocalLap$_{2\star}$ requires first knowing or estimating $d_{\max}$, after which each user computes their local 2-star count, adds Lap$(d_{\max}/\varepsilon)$ noise, and uploads the result. This noise addition leads to larger MSE than 2STAR.

8 Experimental Evaluation

In this section, we will conduct experimental tests on all the algorithms proposed in this paper. The experiments are designed to address the following questions:

RQ1. Does the selection of different DP mechanisms in GNAM affect the accuracy of the algorithm? If so, is the impact significant?

RQ2. For the two-round algorithms in this paper, which parts have a significant impact on the precision of the algorithms? Is it due to the algorithm framework, the GraphProjection, or the noise addition in the second round?

RQ3. How does the comparison of the algorithms provided in this paper with existing algorithms in practical applications? Does it align with the theoretical analysis?

8.1 Experimental Set-up

Datasets. We used the following two real graph datasets, they can be downloaded at [33] :

Facebook [34]. The first dataset is Facebook, which contains 4039 users and 88234 edges. Facebook data was collected from survey participants using the Facebook app. If users i and j are in a friendship relationship, an undirected edge will be connected between them.

CA-AstroPH [32]. The second dataset is Astro Physics collaboration network, which contains 18,772 users and 198,110 edges. CA-AstroPH is from the arXiv e-print and covers scientific collaborations between authors papers submitted to the Astro Physics category. If an author i co-authored a paper with the author j, the graph contains an undirected edge from i to j.

Experimental set up. We performed the experiments on a system equipped with an 11th Gen Intel(R) Core(TM) i7-11700K CPU @ 3.60 GHz, 3600 MHz, 8 cores, and 16 logical processors. We took 12 data points for total privacy budget ε ranging from 0.1 to 2, ran the algorithm 20 times for each data point, and then averaged the experimental results to calculate the MSE and RE. For the two round algorithms in this paper, we uniformly set $\alpha = 20$, $\beta = 0.01$, $\varepsilon_0 = 0.1\varepsilon$, $\varepsilon_1 = 0.8\varepsilon$, and $\varepsilon_2 = 0.1\varepsilon$.

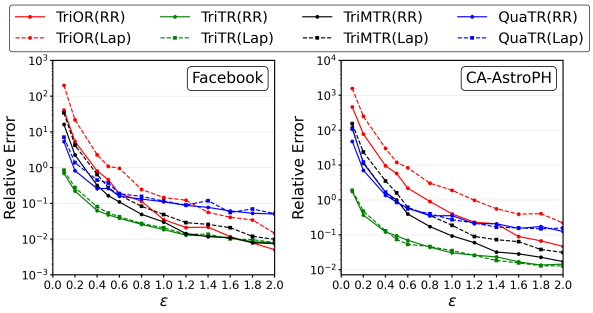

Figure 6: The Relative Error Comparison between RR and Laplace Mechanism. The solid lines all use RR, while the dashed lines use the Laplace mechanism.

To RQ1, each algorithm in this paper is experimentally evaluated respectively under the RR and Laplace mechanisms.

To RQ2, we conduct experiments on each two-round algorithm across four stages: **Stage 1**, the total privacy budget ε is entirely allocated to ε_1 in GNAM, without applying GraphProjection or second-round noise addition. **Stage 2**, based on the previous stage, we reduce the privacy budget of GNAM, setting $\varepsilon_1 = 0.8\varepsilon$. **Stage 3**, GraphProjection is added before GNAM. **Stage 4**, in the last stage, added second-round noise to complete the entire algorithm.

To RQ3, we replicate a subset of the algorithms from Table 2: RR$_\triangle$, 2R-Large$_\triangle$, and Wshuffle$_\triangle$. These three are compared with our TriOR, TriTR, and TriMTR, respectively. The use of Wshuffle$_\triangle$ instead of 2R-Small$_\triangle$ is based on the findings in [24], which demonstrate that Wshuffle$_\triangle$ consistently outperforms 2R-Small$_\triangle$ in practical performance.

8.2 Experimental Results

Figures 6 provides answers to RQ1. A comparison between the RR and Laplace mechanisms shows that, under the same privacy budget, RR consistently achieves higher accuracy than the Laplace mechanism. This is consistent with the theory presented in the paper, as the variance σ^2 of RR is approximately half that of the Laplace mechanism under the same privacy budget. Additionally, it is noticeable that the curves for RR and Laplace corresponding to TriOR, TriMTR, and TriTR diverge more as the privacy budget decreases. This is because their variances are inversely proportional to σ^{-2}, σ^{-4}, and σ^{-6}, respectively. Furthermore, the accuracy of TriOR, TriMTR, and TriTR decreases as ε decreases, which is also due to the same reason. Since RR achieves higher accuracy and existing algorithms also use RR, all following data will use RR in GNAM.

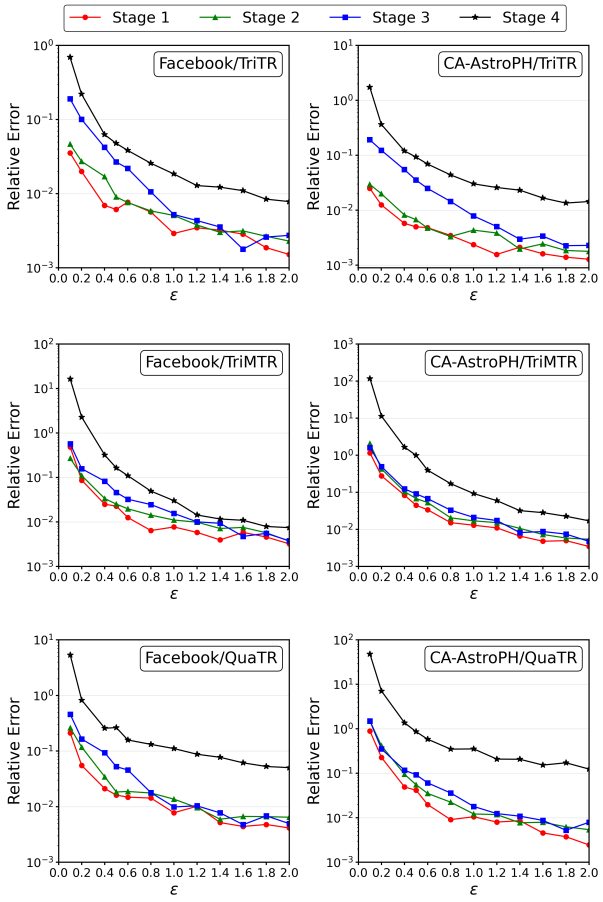

Figure 7: The Relative Error of Two-Round Algorithms at Different Stages. The three figures respectively show the performance of TriTR, TriMTR, and QuaTR.

Figures 7 provides answers to RQ2. A comparison between the red and green lines reveals minimal divergence, indicating that a slight reduction in ε_1 has negligible impact on overall accuracy. Comparative analysis reveals a larger divergence in TriTR between the blue and green trajectories, whereas TriMTR and QuaTR exhibit marginal variations. This shows that GraphProjection is not the primary determinant of accuracy degradation. The most noticeable change occurs after the addition of second-round noise, where the relative error of the three two-round algorithms shows a more significant increase.

Figures 8 provides answers to RQ3. As demonstrated, all solid lines remain consistently below their dashed counterparts (except TriOR and $\text{RR}_\triangle$), which is perfectly consistent with the theoretical analysis presented in Section 7. The observed anomalies in TriOR and $\text{RR}_\triangle$ arise from the scaling operation applied to $\text{RR}_\triangle$ during the theoretical MSE derivation process. The triplet enumeration based $\text{RR}_\triangle$ required 63,033 s for 240 estimations on Facebook dataset (vs. TriOR's 780 s). In CA-AstroPH, TriOR required 17, 955 s, while $\text{RR}_\triangle$'s

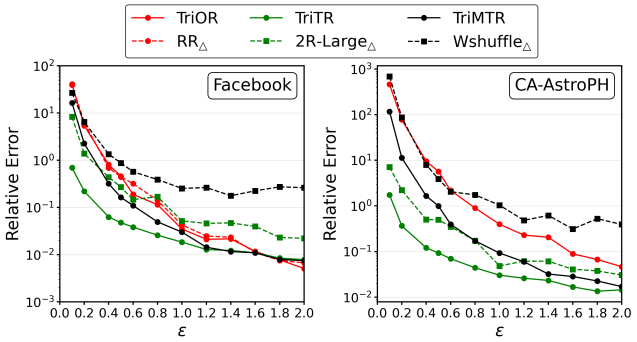

Figure 8: Comparison of Relative Errors among Existing Private Triangle Counting Algorithms. Comparative curves employ same coloration, where solid lines represent our proposed algorithm, and dashed lines denote the existing algorithms.

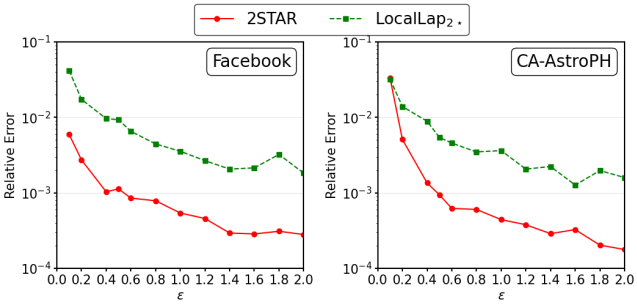

Figure 9: The Relative Error Comparison between 2-Star Counting Algorithms

prohibitive computational cost prevented its evaluation.

Figure 9 compares the 2-star counting algorithms. It can be observed that 2STAR achieves higher accuracy than $\text{LocalLap}_{2\star}$. On CA-AstroPH, 2STAR exhibits a faster decreasing rate, which is due to its lower d_{avg}. This causes the $\frac{20}{\varepsilon^4}n$ term in its MSE to play a more significant role, resulting in a higher rate of change with respect to ε.

Tables 3 provides the accuracy, computation time and Cost_{DL} of our algorithms. The data is stored in float64 format. From the perspective of relative error, the four algorithms achieve high accuracy (RE < 0.4). It can also be observed that the algorithm achieves higher accuracy on Facebook compared to CA-AstroPH. For TriOR, this is due to the increase in n, as the actual number of triangles struggles to keep up with the growth of the term $O(n^3)$ in MSE. As for the two-round algorithms, this is primarily because Facebook has larger d_{avg}.

9 Conclusion

This paper proposes TriOR, TriTR, TriMTR, and QuaTR algorithms based on the properties of noisy adjacency matrices. We implement edge LDP guarantees for the second round of two-round algorithms using a confidence interval-inspired

Table 3: Performance of the Proposed Algorithms.

(a) Facebook

Algorithm	RE ($\varepsilon = 1.0$)	RE ($\varepsilon = 2.0$)	Time	Cost_{DL}
TriOR	3.49×10^{-2}	5.07×10^{-3}	3.25 s	0
TriTR	1.85×10^{-2}	7.82×10^{-3}	5.58 s	124.97 MB
TriMTR	3.01×10^{-2}	7.45×10^{-3}	2.20 s	31.58 KB
QuaTR	1.11×10^{-1}	5.03×10^{-2}	3.53 s	124.97 MB
2STAR	5.41×10^{-4}	2.81×10^{-4}	0.01 s	0

(b) CA-AstroPH

Algorithm	RE ($\varepsilon = 1.0$)	RE ($\varepsilon = 2.0$)	Time	Cost_{DL}
TriOR	3.96×10^{-1}	4.65×10^{-2}	74.8 s	0
TriTR	3.03×10^{-2}	1.44×10^{-2}	22.6 s	2.66 GB
TriMTR	9.26×10^{-2}	1.70×10^{-2}	65.8 s	147.46 KB
QuaTR	3.53×10^{-1}	1.24×10^{-1}	67.1 s	2.66 GB
2STAR	4.42×10^{-4}	1.78×10^{-4}	0.04 s	0

mechanism, while designing the 2STAR algorithm for 2-star counting. Theoretical upper bounds on MSE are established for every algorithm. The experimental evaluation demonstrates that TriOR achieves an accuracy comparable to the existing one-round algorithm while reducing the time complexity to $O(n^{2.371866})$. Other proposed algorithms exhibit precision improvements over existing algorithms. All Algorithms maintain relative errors below 0.4 with reasonable privacy budgets $\varepsilon = 1$.

In future work, we will integrate the noisy adjacency matrix with the graph convolutional neural network to enable model training while preserving users' edge privacy.

References

[1] ACHARYA, J., SUN, Z., AND ZHANG, H. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), PMLR, pp. 1120–1129.

[2] ALIAKBARPOUR, M., BISWAS, A. S., GOULEAKIS, T., PEEBLES, J., RUBINFELD, R., AND YODPINYANEE, A. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica 80* (2018), 668–697.

[3] AVRON, H. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications* (2010), vol. 10, p. 9.

[4] BALLE, B., BELL, J., GASCÓN, A., AND NISSIM, K. The privacy blanket of the shuffle model. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39* (2019), Springer, pp. 638–667.

[5] BERA, S. K., AND SESHADHRI, C. How the degeneracy helps for triangle counting in graph streams. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2020), pp. 457–467.

[6] BERNAU, D., ROBL, J., GRASSAL, P. W., SCHNEIDER, S., AND KERSCHBAUM, F. Comparing local and central differential privacy using membership inference attacks. In *IFIP Annual Conference on Data and Applications Security and Privacy* (2021), Springer, pp. 22–42.

[7] BI, M., WANG, Y., CAI, Z., AND TONG, X. A privacy-preserving mechanism based on local differential privacy in edge computing. *China Communications 17*, 9 (2020), 50–65.

[8] BJÖRKLUND, A., LOKSHTANOV, D., SAURABH, S., AND ZEHAVI, M. Approximate counting of k-paths: Deterministic and in polynomial space. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)* (2019), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[9] CHEU, A., SMITH, A., ULLMAN, J., ZEBER, D., AND ZHILYAEV, M. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38* (2019), Springer, pp. 375–403.

[10] CHIBA, N., AND NISHIZEKI, T. Arboricity and subgraph listing algorithms. *SIAM Journal on computing 14*, 1 (1985), 210–223.

[11] COPPERSMITH, D., AND WINOGRAD, S. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (1987), pp. 1–6.

[12] DONG, J., ROTH, A., AND SU, W. J. Gaussian differential privacy. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 84*, 1 (2022), 3–37.

[13] DUAN, R., WU, H., AND ZHOU, R. Faster matrix multiplication via asymmetric hashing. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)* (2023), IEEE, pp. 2129–2138.

[14] DWORK, C. Differential privacy. In *International colloquium on automata, languages, and programming* (2006), Springer, pp. 1–12.

[15] DWORK, C., MCSHERRY, F., NISSIM, K., AND SMITH, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3* (2006), Springer, pp. 265–284.

[16] DWORK, C., ROTH, A., ET AL. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science 9*, 3–4 (2014), 211–407.

[17] EDEN, T., LEVI, A., RON, D., AND SESHADHRI, C. Approximately counting triangles in sublinear time. *SIAM Journal on Computing 46*, 5 (2017), 1603–1646.

[18] EDEN, T., LIU, Q. C., RASKHODNIKOVA, S., AND SMITH, A. Triangle counting with local edge differential privacy. *arXiv preprint arXiv:2305.02263* (2023).

[19] ERLINGSSON, Ú., FELDMAN, V., MIRONOV, I., RAGHUNATHAN, A., TALWAR, K., AND THAKURTA, A. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (2019), SIAM, pp. 2468–2479.

[20] FELDMAN, V., MCMILLAN, A., AND TALWAR, K. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)* (2022), IEEE, pp. 954–964.

[21] GONEN, M., RON, D., AND SHAVITT, Y. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics 25*, 3 (2011), 1365–1411.

[22] IMOLA, J., MURAKAMI, T., AND CHAUDHURI, K. Locally differentially private analysis of graph statistics. In *30th USENIX security symposium (USENIX Security 21)* (2021), pp. 983–1000.

[23] IMOLA, J., MURAKAMI, T., AND CHAUDHURI, K. {Communication-Efficient} triangle counting under local differential privacy. In *31st USENIX security symposium (USENIX Security 22)* (2022), pp. 537–554.

[24] IMOLA, J., MURAKAMI, T., AND CHAUDHURI, K. Differentially private triangle and 4-cycle counting in the shuffle model. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022), pp. 1505–1519.

[25] JIAN, X., WANG, Y., AND CHEN, L. Publishing graphs under node differential privacy. *IEEE Transactions on Knowledge and Data Engineering 35*, 4 (2021), 4164–4177.

[26] KAIROUZ, P., BONAWITZ, K., AND RAMAGE, D. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning* (2016), PMLR, pp. 2436–2444.

[27] KALLAUGHER, J., MCGREGOR, A., PRICE, E., AND VOROTNIKOVA, S. The complexity of counting cycles in the adjacency list streaming model. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2019), pp. 119–133.

[28] KARTUN-GILES, A. P., AND KIM, S. Counting k-hop paths in the random connection model. *IEEE Transactions on Wireless Communications 17*, 5 (2018), 3201–3210.

[29] KASIVISWANATHAN, S. P., NISSIM, K., RASKHODNIKOVA, S., AND SMITH, A. Analyzing graphs with node differential privacy. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings* (2013), Springer, pp. 457–476.

[30] KOLOUNTZAKIS, M. N., MILLER, G. L., PENG, R., AND TSOURAKAKIS, C. E. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics 8*, 1-2 (2012), 161–185.

[31] LE GALL, F. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation* (2014), pp. 296–303.

[32] LESKOVEC, J., KLEINBERG, J., AND FALOUTSOS, C. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 2–es.

[33] LESKOVEC, J., AND KREVL, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[34] LESKOVEC, J., AND MCAULEY, J. Learning to discover social circles in ego networks. *Advances in neural information processing systems 25* (2012).

[35] LIU, Y., ZHAO, S., LIU, Y., ZHAO, D., CHEN, H., AND LI, C. Collecting triangle counts with edge relationship local differential privacy. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)* (2022), IEEE, pp. 2008–2020.

[36] MANJUNATH, M., MEHLHORN, K., PANAGIOTOU, K., AND SUN, H. Approximate counting of cycles in streams. In *Algorithms–ESA 2011: 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings 19* (2011), Springer, pp. 677–688.

[37] MCGREGOR, A., AND VOROTNIKOVA, S. Triangle and four cycle counting in the data stream model. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (2020), pp. 445–456.

[38] MCSHERRY, F. D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (2009), pp. 19–30.

[39] MORRIS, C. The number of data breaches in 2021 has already surpassed last year's total, 2021.

[40] MURAKAMI, T., AND KAWAMOTO, Y. {Utility-optimized} local differential privacy mechanisms for distribution estimation. In *28th USENIX Security Symposium (USENIX Security 19)* (2019), pp. 1877–1894.

[41] NASERI, M., HAYES, J., AND DE CRISTOFARO, E. Local and central differential privacy for robustness and privacy in federated learning. *arXiv preprint arXiv:2009.03561* (2020).

[42] NGUYEN, D., HALAPPANAVAR, M., SRINIVASAN, V., AND VULLIKANTI, A. Faster approximate subgraph counts with privacy. *Advances in Neural Information Processing Systems 36* (2024).

[43] NISSIM, K., RASKHODNIKOVA, S., AND SMITH, A. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing* (2007), pp. 75–84.

[44] ORTMANN, M., AND BRANDES, U. Triangle listing algorithms: Back from the diversion. In *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)* (2014), SIAM, pp. 1–8.

[45] QIN, Z., YU, T., YANG, Y., KHALIL, I., XIAO, X., AND REN, K. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (2017), pp. 425–438.

[46] RASKHODNIKOVA, S., AND SMITH, A. Differentially private analysis of graphs. *Encyclopedia of Algorithms* (2016).

[47] RIBEIRO, P., PAREDES, P., SILVA, M. E., APARICIO, D., AND SILVA, F. A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR) 54*, 2 (2021), 1–36.

[48] SATISH, N., SUNDARAM, N., PATWARY, M. M. A., SEO, J., PARK, J., HASSAAN, M. A., SENGUPTA, S., YIN, Z., AND DUBEY, P. Navigating the maze of graph analytics frameworks using massive graph datasets. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (2014), pp. 979–990.

[49] SESHADHRI, C., PINAR, A., AND KOLDA, T. G. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM international conference on data mining* (2013), SIAM, pp. 10–18.

[50] STOTHERS, A. J. On the complexity of matrix multiplication.

[51] STRASSEN, V. Gaussian elimination is not optimal. *Numerische mathematik 13*, 4 (1969), 354–356.

[52] TSOURAKAKIS, C. E., KANG, U., MILLER, G. L., AND FALOUTSOS, C. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), pp. 837–846.

[53] WANG, N., ZHANG, J., TAN, K.-L., AND TUNG, A. K. On triangulation-based dense neighborhood graph discovery. *Proceedings of the VLDB Endowment 4*, 2 (2010), 58–68.

[54] WANG, T., BLOCKI, J., LI, N., AND JHA, S. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)* (2017), pp. 729–745.

[55] WANG, T., MEI, Y., JIA, W., ZHENG, X., WANG, G., AND XIE, M. Edge-based differential privacy computing for sensor–cloud systems. *Journal of Parallel and Distributed computing 136* (2020), 75–85.

[56] WANG, Y., WU, X., AND HU, D. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops* (2016), vol. 1558, pp. 0090–6778.

[57] WARNER, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association 60*, 309 (1965), 63–69.

[58] WILLIAMS, V. V. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing* (2012), pp. 887–898.

[59] YANG, M., GUO, T., ZHU, T., TJUAWINATA, I., ZHAO, J., AND LAM, K.-Y. Local differential privacy and its applications: A comprehensive survey. *Computer Standards & Interfaces 89* (2024), 103827.

A Directed Graph Application

In the main text, we mentioned that our algorithm can be easily adapted to directed graphs. This is because, by simply removing the symmetry condition in the definition of the Noisy Adjacency Matrix, it can be seamlessly applied to directed graphs while still preserving all its properties.

Here, we provide the definition of the noisy adjacency matrix for directed graphs:

Definition 4 (Noisy adjacency matrix)**.** *$\hat{A}$ is the noisy adjacency matrix of directed graph $G \in \mathcal{G}$, if $\hat{A}$ satisfies:*

$$\mathbb{E}\left[\hat{A}\right] = A;\ \hat{a}_{ii} = 0,\ \textit{for any } i \in [n];$$
$$\hat{a}_{ij} \perp \hat{a}_{kl},\ \textit{for any } i < j,\ k < l,\ (i,j) \neq (k,l).$$

where A is the adjacency matrix of graph G, n is the number of nodes.

It still retains the properties stated in Theorem 1. Moreover, the subsequent algorithm design is largely similar to that for undirected graphs, with the only necessary adjustment being the consideration of directionality. Here, if we define triangles and quadrangles in directed graphs as 3/4-step loops without traversing the same edge more than once, the corresponding algorithms are as follows:

Algorithm 11 GNAM'

Input: $\varepsilon \in \mathbb{R}_{\geq 0}$, graph G's adjacency list $\mathbf{a}_1$,$\mathbf{a}_2$,...,$\mathbf{a}_n \in \{0,1\}^n$.
Output: Noisy adjacency matrix of graph G.
for *each user* $u = 1$ to n **do:**
  $\tilde{\mathbf{a}}_u \leftarrow$ *local randomizer* $(\mathbf{a}_u, \varepsilon)$
  send $\tilde{\mathbf{a}}_u$ to *data collector*
Data collector do:
$\hat{A} \leftarrow$ *estimate algorithm* $\left(\tilde{A}\right)$
return $\hat{A}$

Algorithm 12 TriOR'

Input: $\varepsilon \in \mathbb{R}_{\geq 0}$, graph G .
Output: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
$\hat{A} \leftarrow \mathrm{GNAM}(G, \varepsilon)$
return $\mathrm{tr}\left(\hat{A}^3\right)/3$

Algorithm 13 TriTR'

Input: Graph G, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$.
Output: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
#First round:
$\hat{A} \leftarrow \mathrm{GNAM}(G, \varepsilon_1)$
#Second round:
for each *user* $u = 1$ to n **do:**
  download $\hat{A}$
  $sum_u \leftarrow \sum_{(i,j):a_{ui}=a_{ju}=1} \hat{a}_{ij}$
  $\hat{T}_u \leftarrow$ *second randomizer* (sum_u, ε_2)
  upload $\hat{T}_u$ to *data collector*
data collector **do:**
$\hat{f}^{\triangle}(G) \leftarrow \frac{1}{3}\sum_{i=u}^{n} \hat{T}_u$
return $\hat{f}^{\triangle}(G)$

Algorithm 14 TriMTR'

Input: Graph G, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$.
Output: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
#First round:
$\hat{A} \leftarrow \mathrm{GNAM}(G, \varepsilon_1)$
Data collector **calculate:** $\hat{B} \leftarrow \hat{A}^2$
#Second round:
for each node $u = 1$ to n **do:**
  download the u-th column of $\hat{B}$
  $sum_u \leftarrow \sum_{i:a_{ui}=1} \hat{b}_{iu}$
  $\hat{T}_u \leftarrow$ *second randomizer* (sum_u, ε_2)
  upload $\hat{T}_u$ to **data collector**
Data collector do:
$\hat{f}^{\triangle}(G) \leftarrow \frac{1}{3}\sum_{u=1}^{n} T_u$
return $\hat{f}^{\triangle}(G)$

Algorithm 15 QuaTR'

Input: Graph G, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$.
Output: Estimate $\hat{f}^{\square}(G)$ of $f^{\square}(G)$.
#First round:
$\hat{A} \leftarrow \mathrm{GNAM}(G, \varepsilon_1)$
Data collector **calculate:** $\hat{B} \leftarrow \hat{A}^2$
#Second round:
for each node $u = 1$ to n **do:**
  download $\hat{B}$
  $sum_u \leftarrow \sum_{(i,j):a_{ui}=a_{ju}=1} \hat{b}_{ij}$
  $\hat{Q}_u \leftarrow$ *second randomizer* (sum_u, ε_2)
  upload $\hat{Q}_u$ to **data collector**
Data collector do:
$\hat{f}^{\square}(G) \leftarrow \frac{1}{4}\sum_{u=1}^{n} \hat{Q}_u$
return $\hat{f}^{\square}(G)$

Among these changes, there are only three modifications: (1) the coefficient by which each algorithm divides at the end

Table 4: Relative Err of Estimations for Triangle, Quadrangle and 2-Star ($\varepsilon = 1.1$).

Dataset	TriMTR	QuaTR	2STAR
Facebook	0.0301	0.1108	0.0052
CA-AstroPH	0.0926	0.3531	0.0322

is adjusted to account for the single directionality of edges in directed graphs, as opposed to the bidirectional nature in undirected graphs; (2) in the quadrangle algorithm, there is no need to subtract 1 in line 5, as the directed nature of the edges prevents traversal back along the original path; and (3) in TriTR and QuaTR, when calculating sum_u, the noise relationship added changes from $a_{ui} = a_{uj} = 1$ to $a_{ui} = a_{ju} = 1$.

Furthermore, although their variances do not match the specific expressions in undirected graphs, they still satisfy the Variance in Matrix Form when coefficients are ignored. The proofs for directed graphs are not elaborated in detail here, as the proof methodology closely resembles that used for undirected graphs.

B Weighted Graph Application

If the original adjacency matrix $A \in \mathbb{R}^{n\times n}$, it still retains the properties stated in Theorem 1, as the proof of Theorem 1 does not require A to belong to $\{0,1\}^{n\times n}$. When dealing with weighted graphs, GNAM can utilize the Laplace mechanism for edge-LDP.

C Estimate Three Subgraphs in Two Rounds

Table 4 shows the estimation performance for three subgraphs obtained by two-round query. We allocated $\varepsilon_0 = 0.1$ to GraphProjection, $\varepsilon_1 = 0.8$ to GNAM, $\varepsilon_2 = 0.1$ to the second round of TriMTR, and $\varepsilon_3 = 0.1$ to the second round of QuaTR. Thus, a total privacy budget of $\varepsilon = 1.1$ suffices to estimate all three subgraph counts while achieving the accuracy shown in the table.

D Second Round Risk without Randomizer

At the beginning of Chapter 6, we mentioned the necessity of addressing the issue of adding noise in the second round, as directly uploading sum_u is not feasible. Uploading sum_u directly would not only fail to provide ε_2-edge LDP in the second round but also expose to the following attack risks:

If the DP mechanism employed in GNAM is a continuous noise mechanism, such as the Laplace Mechanism or Gaussian Mechanism. Because the probability of the continuous random variable taking any specific value is zero, this characteristic enables the following potential attack strategy:

Assume adversary can obtain the noisy adjacency matrix $\hat{A}$. The adversary can enumerate node v_u's possible neighbor list: Nei'_u. Denote the true neighbor list of user v_u as Nei_u. If a certain Nei'_u satisfies: For TriTR, $\sum_{i,j\in\text{Nei}'_u} \hat{a}_{ij} = sum_u$; For TriMTR, $\sum_{i\in\text{Nei}'_u} \hat{b}_{iu} = sum_u$; For QuaTR, $\sum_{i,j\in\text{Nei}'_u} \hat{b}_{ij} = sum_u$. Then the adversary can determine that this $N(u)'$ is the correct neighbor list (Nei_u) of v_u.

E Proof of Statements

E.1 Proof of Theorem 1

Given that the noisy adjacency matrix $\hat{A}$ satisfies all the conditions of Definition 7, we can view $\hat{A}$ from an alternative perspective: $\hat{A} = A + X$, where A is the adjacency matrix of the original graph, and X is a noise matrix that satisfies the following properties:

$$\begin{gathered}\mathbb{E}[X] = 0;\\ X^T = X;\\ x_{ii} = 0, \quad \text{for any } i \in [n];\\ x_{ij} \perp x_{kl}, \quad \text{for any } i < j, k < l, (i,j) \neq (k,l)\end{gathered}$$

First properity. $\hat{B} = (A+X)^2 = A^2 + AX + XA + X^2$.

For each $i, j, i \neq j$:

$$\tilde{b}_{ij} = b_{ij} + \sum_{k=1}^{n} a_{ik}x_{kj} + \sum_{k=1}^{n} x_{ik}a_{kj} + \sum_{k=1}^{n} x_{ik}x_{kj}.$$

Focusing on the second item:

$$\mathbb{E}[\sum_{k=1}^{n} a_{ik}x_{kj}] = \sum_{k=1}^{n} \mathbb{E}[a_{ik}x_{kj}] = \sum_{k=1}^{n} 0 = 0.$$

The third item is similar to the second item. Then the forth item, since $i \neq j$, x_{ik} and x_{kj} are independent. Therefore, we have:

$$\mathbb{E}[\sum_{k=1}^{n} x_{ik}x_{kj}] = \sum_{k=1}^{n} \mathbb{E}[x_{ik}x_{kj}] = \sum_{k=1}^{n} \mathbb{E}[x_{ik}]\mathbb{E}[x_{kj}] = 0$$

Therefore, For each $i, j, i \neq j$, $\mathbb{E}[\hat{b}_{ij}] = b_{ij}$.

Second peoperity. $\tilde{C} = (A+X)^3 = A^3 + AAX + AXA + XAA + AXX + XAX + XXA + X^3$.

For each i:

$$\begin{aligned}\tilde{c}_{ii} =& c_{ii} + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}a_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}a_{jk}a_{ki}\\ &+ \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}a_{jk}x_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}a_{ki}\\ &+ \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki}\end{aligned}$$

The expected value of each subsequent term is zero except for the first term, .

For the second, third, and fourth terms, each summation contains only a single x. Since $\mathbb{E}[x] = 0$, the expectation of these terms is also zero.

For the fifth, sixth, and seventh terms, each summation contains two terms x_{ij} and x_{jk}. Given $\mathbb{E}[x_{ij}] = 0$ and $\mathbb{E}[x_{jk}] = 0$, if these terms are independent, their product has an expectation of zero. If they are not independent, $(i,j) = (j,k)$ or $(i,j) = (k,j)$, there exist $i = k$ or $j = k$, because $a_{ii} = 0$ for any $i \in [n]$, we have $\mathbb{E}[a_{ki}x_{ij}x_{jk}] = 0$.

Therefore, the expectation of the fifth, sixth, and seventh terms is zero.

For the eighth term, each summation includes $x_{ij}x_{jk}x_{ki}$. If these three terms are independent, the expectation of their product is the product of their expectations, which equals zero. If at least two among x_{ij}, x_{jk}, and x_{ki} are not independent, then at least one pair among i, j, and k must be equal. Then for any i, $x_{ii} = 0$, which consequently ensures that the product of them is zero. Therefore, in all cases, the expectation of the eighth term is zero. Then for each i, $\mathbb{E}[\hat{c}_{ii}] = c_{ii}$.

E.2 Proof of Theorem 2

In the adjacency list $\mathbf{a}_i$ of node v_i, $a_{ij} = 1$ indicates that node v_i can reach node v_j in one step, while 0 indicates that it cannot. Thus, for any two adjacent adjacency lists $\mathbf{a}_i$ and $\mathbf{a}_i'$, they differ by at most one bit, $\Delta f = 1$. According to the Laplace mechanism theorem as guaranteed by [15], we have:

The local randomizer $\mathcal{R}$ satisfies: for any two neighboring lists $\mathbf{a}_i, \mathbf{a}_i' \in \{0,1\}^n$ that differ by one bit and any $S \subseteq \text{Range}(\mathcal{R})$,

$$\Pr[\mathcal{R}(\mathbf{a}_i) \in S] \leq e^{\varepsilon} \Pr[\mathcal{R}(\mathbf{a}_i') \in S].$$

Thus, uploading $\tilde{a}_i$ can provide ε-edge LDP.

Each node only uploads the connection information with nodes that have a smaller index. This approach is taken to avoid the scenario where the information of the same edge is uploaded by both of its endpoints. According to the composition theorem [15], if an adversary gains access to information uploaded by both of them, it would only guarantee 2ε-edge differential privacy [43].

E.3 Proof of Proposition 3

If $X = 1$, then $Y = \frac{e^{\varepsilon}}{e^{\varepsilon}-1}$. If $X = 0$, then Y takes $Y = -\frac{1}{e^{\varepsilon}-1}$. Given the original true value is 1, we have the expected value $\mathbb{E}[Y] = \frac{e^{\varepsilon}}{e^{\varepsilon}+1} \cdot \frac{e^{\varepsilon}}{e^{\varepsilon}-1} - \frac{1}{e^{\varepsilon}+1} \cdot \frac{1}{e^{\varepsilon}-1} = 1$. Given the original true value is 0, we have the expected value $\mathbb{E}[Y] = \frac{e^{\varepsilon}}{e^{\varepsilon}+1} \cdot \frac{-1}{e^{\varepsilon}-1} + \frac{1}{e^{\varepsilon}+1} \cdot \frac{e^{\varepsilon}}{e^{\varepsilon}-1} = 0$.

E.4 Proof of Theorem 3

Given that GNAM provides ε_1-edge LDP, and based on the immunity to post-processing [14], it follows that TriTR also provides ε_1-edge LDP.

E.5 Lemma 1

To facilitate subsequent proofs, let us get Lemma 1.

Lemma 1. *Let $Z_1, Z_2, \ldots$ be a sequence of mutually independent random variables, and for each i, $\mathbb{E}[Z_i] = 0$. Then, for two finite integer sequences $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{n1})$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_{n2})$, and their corresponding exponent sequences $k = (k_1, \ldots, k_{n_1}) \in \{1,2\}^{n_1}$, $l = (l_1, \ldots, l_{n_2}) \in \{1,2\}^{n_2}$ satisfies:*

$$\mathbb{E}\left(\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i}\right) = 0, \mathbb{E}\left(\prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right) = 0$$

And there exists a number $m \in \alpha \cup \beta$, $m \notin \alpha \cap \beta$, and the corresponding exponent q for that number m is 1, it holds that:

$$\mathbb{V}\left(\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i} + \prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right) = \mathbb{V}\left(\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i}\right) + \mathbb{V}\left(\prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right)$$

Proof.

$$\begin{aligned}
&\mathbb{V}\left(\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i} + \prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right) \\
&= \mathbb{E}\left(\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i} + \prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right)^2 \\
&= \mathbb{E}\left[\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i}\right]^2 + \mathbb{E}\left[\prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right]^2 + 2\mathbb{E}\left[\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i} \cdot \prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right] \\
&= \mathbb{V}\left(\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i}\right) + \mathbb{V}\left(\prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right)
\end{aligned}$$

The expectation $\mathbb{E}\left[\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i} \cdot \prod_{i=1}^{n_2} Z_{\beta_i}^{k_i'}\right] = 0$ holds because there exists an $m \in \alpha \cup \beta$, $m \notin \alpha \cap \beta$ and the exponent of Z_m is 1. Because the independence among Z_i, Z_m can be factored out. Let Q denote the remaining product. Then we have:

$$\mathbb{E}\left[\prod_{i=1}^{n_1} Z_{\alpha_i}^{k_i} \cdot \prod_{i=1}^{n_2} Z_{\beta_i}^{l_i}\right] = \mathbb{E}[Z_m] \cdot \mathbb{E}[Q] = 0$$

E.6 Proof of Theorem 4

In the proof that follows, we continue to use the notation in Theorem 1, denoting $\hat{A} = A + X$.

The proof of unbiasedness follows easily from the second clause of Theorem 3:

$$\mathbb{E}\left[\frac{\mathrm{tr}\left(\hat{A}^3\right)}{6}\right] = \frac{\mathrm{tr}\left(A^3\right)}{6} = f^{\triangle}(G)$$

Then the MSE:

$$\begin{aligned}\mathrm{MSE}(\hat{f}^{\triangle}(G)) &= \mathbb{E}\left[\hat{f}^{\triangle}(G) - \mathbb{E}\left[\hat{f}^{\triangle}(G)\right]\right]^2 \\ &= \mathbb{E}\left[\hat{f}^{\triangle}(G) - f^{\triangle}(G)\right]^2 = \mathbb{V}(\hat{f}^{\triangle}(G)).\end{aligned}$$

On the diagnal:

$$\begin{aligned}\hat{c}_{ii} =& c_{ii} + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}a_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}a_{jk}a_{ki} \\ &+ \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}a_{jk}x_{ki} + \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}a_{ki} \\ &+ \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki}\end{aligned}$$

Therefore:

$$\begin{aligned}\mathbb{V}\left(\mathrm{tr}\left(\hat{A}^3\right)\right) &= \mathbb{V}\left(\sum_{i=1}^{n}\hat{c}_{ii}\right) \\ &= \mathbb{V}\Bigg(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}a_{ki} + 3\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki} \\ &\quad + 3\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki}\Bigg)\end{aligned}$$

According to Lemma 1, the variance of the aforementioned expression is equal to the sum of the variances of its individual components:

$$\begin{aligned}&\mathbb{V}\left(\mathrm{tr}\left(\hat{A}^3\right)\right) = \\ &\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}a_{ki}\right) + 9\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki}\right) \\ &+ 9\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki}\right) + \mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki}\right)\end{aligned}$$

Next, the overall variance can be calculated by determining the variance of these four components. The first:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}a_{ki}$$

This is a constant, equal to $\mathrm{tr}(A^3)$, and its variance is 0.
The second item:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki}$$

From the perspective of the graph, $a_{ij}a_{jk}$ means that there exist a path starting from v_i and passing through v_j to v_k in the actual graph. If this path exists, then x_{ki} will be added to this term. So x_{ij} will be added b_{ij} times. Since this is in an undirected graph, x_{ji} will also be added b_{ij} times, and $x_{ij} = x_{ji}$, therefore:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki} = 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}x_{ij}$$

Given that $\mathbb{V}(x_{ij}) = \mathbb{V}(a_{ij}) = \sigma^2$, we have:

$$\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki}\right) = 4\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2$$

Next, we analyze the third term:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\sum_{k=1}^{n} x_{jk}x_{ki}$$

If there exists an undirected edge (v_i, v_j) in the graph, then $\sum_{k=1}^{n} x_{jk}x_{ki}$ will be added into the third term. Let $I_{\text{proposition}} = 1$ if the proposition holds true, and $I_{\text{proposition}} = 0$ otherwise. In addition, beacuse $x_{ii} = 0$, therefore:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\sum_{k=1}^{n} x_{jk}x_{ki} = 2\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(I_{(v_i,v_j)\in E}\cdot\sum_{k=1,k\neq i,j}^{n} x_{ik}x_{kj}\right)$$

According to Lemma 1, and $\mathbb{V}(x_{ik}x_{kj}) = \mathbb{V}(x_{ik})\cdot\mathbb{V}(x_{kj}) = \sigma^4$. Therefore:

$$\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki}\right) = 4\sigma^4(n-2)|E|$$

The fourth term:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki}$$

Since in an undirected graph, $x_{ij} = x_{ji}$, $x_{ii} = 0$, thus:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki} = 6\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n} x_{ij}x_{jk}x_{ki}$$

Based on Lemma 1, and $\mathbb{V}(x_{ij}x_{jk}x_{ki}) = \mathbb{V}(x_{ij})\cdot\mathbb{V}(x_{jk})\cdot\mathbb{V}(x_{ki}) = \sigma^6$, we get:

$$\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ij}x_{jk}x_{ki}\right) = 6\sigma^6 n(n-1)(n-2)$$

According to all the above equations, we get:

$$\begin{aligned}&\mathrm{MSE}(\hat{f}^{\triangle}(G)) \\ &= \mathbb{V}(\hat{f}_1^{\triangle}(G)) = \mathbb{V}\left(\mathrm{tr}\left(\hat{A}^3\right)/6\right) \\ &= \sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2 + \sigma^4(n-2)|E| + \frac{1}{6}\sigma^6 n(n-1)(n-2)\end{aligned}$$

Even if each node has $d_{\max}$ edges, there can be at most $n \cdot d_{\max}^2$ paths of the form $v_i \to v_j \to v_k$ in the graph. Each $b_{ij} \leqslant d_{\max}$. Given a fixed number of paths like $v_i \to v_j \to v_k$, concentrating all the paths on $nd_{\max}$ b_{ij}, each $b_{ij} = d_{\max}$, can maximize $\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2$ to $nd_{\max}^3$. In undirected graph, $|E| = \frac{1}{2}nd_{\text{avg}} \leqslant \frac{1}{2}nd_{\max}$, and $n(n-1)(n-2) \leqslant n^3$. Therefore:

$$\text{MSE}\left(\hat{f}^{\triangle}(G)\right) \leqslant \sigma^2 nd_{\max}^3 + \frac{1}{2}\sigma^4 n^2 d_{\max} + \frac{1}{6}\sigma^6 n^3$$

Therefore,

$$\text{MSE}\left(\hat{f}^{\triangle}(G)\right) \leqslant O(nd_{\max}^3 + n^3).$$

E.7 Proof of Theorem 5

According to Theorem 3, GNAM provides ε_1-edge LDP, and since the *second randomizer* provides ε_2-edge LDP, by the composition theorem, the entire TriTR provides $(\varepsilon_1+\varepsilon_2)$-edge LDP.

E.8 Proof of Theorem 6

Let us denote $\hat{A} = A + X$, where X follows the same property as in Theorem 1. For each u:

$$\begin{aligned} sum_u &= \sum_{(i,j):a_{ui}=a_{uj}=1} \hat{a}_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{uj}\hat{a}_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{uj}(a_{ij}+x_{ij}) = 2{f_u}^{\triangle} + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{uj}x_{ij}. \end{aligned}$$

where ${f_u}^{\triangle}$ is the number of triangles in which user u is involved. Therefore:

$$\sum_{u=1}^{n} sum_u = 2\sum_{u=1}^{n} {f_u}^{\triangle} + \sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{uj}x_{ij}$$

The first item $\sum_{u=1}^{n} {f_u}^{\triangle} = 3f^{\triangle}(G)$. For the second item, because $A = A^T$, $X = X^X$, we get:

$$\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{uj}x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki}$$

In the proof of Theorem 4, we get:

$$\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki}\right] = 0$$

$$\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki}\right) = 4\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2$$

Therefore:

$$\mathbb{E}\left[\frac{1}{6}\sum_{i=1}^{n} sum_u\right] = f^{\triangle}(G)$$

$$\mathbb{V}\left[\frac{1}{6}\sum_{i=1}^{n} sum_u\right] = \frac{1}{9}\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2$$

Finally we get:

$$\text{MSE}\left(\frac{1}{6}\sum_{i=1}^{n} sum_u\right) = \frac{1}{9}\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2$$

In the proof of Theorem 6, we have $\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2 \leqslant nd_{\max}^3$, therefore:

$$\text{MSE}\left[\frac{1}{6}\sum_{i=1}^{n} sum_u\right] \leqslant O(nd_{\max}^3)$$

E.9 Proof of Theorem 7

Since GNAM provides an ε_1-edge LDP and the second round's *second randomizer* provides ε_2-edge LDP, therefore, according to the composition theorem, TriMTR provides $(\varepsilon_1+\varepsilon_2)$-edge LDP.

E.10 Proof of Theorem 8

Align with Theorem 1, $\hat{A} = A + X$. For each u:

$$\begin{aligned} sum_u &= \textstyle\sum_{i:a_{ui}=1} \hat{b}_{iu} = \sum_{i=1}^{n} a_{ui}\hat{b}_{iu} \\ \hat{b}_{iu} &= \textstyle\sum_{j=1}^{n} \hat{a}_{ij}\hat{a}_{ju} = \sum_{j=1}^{n}(a_{ij}+x_{ij})(a_{ju}+x_{ju}) \end{aligned}$$

Therefore:

$$\begin{aligned} &\sum_{u=1}^{n} sum_u \\ &= \sum_{u=1}^{n}\sum_{i=1}^{n} a_{ui}\sum_{j=1}^{n}(a_{ij}+x_{ij})(a_{ju}+x_{ju}) \\ &= \sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}(a_{ij}+x_{ij})(a_{ju}+x_{ju}) \\ &= \sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{ij}a_{ju} + a_{ui}x_{ij}a_{ju} + a_{ui}a_{ij}x_{ju} + a_{ui}x_{ij}x_{ju} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}a_{ki} + 2\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki} \\ &\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki} \\ &= 6f^{\triangle}(G) + 2\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}a_{jk}x_{ki} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}x_{jk}x_{ki} \end{aligned}$$

In the proof of Theorem 4, we get:

$$\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ij}a_{jk}x_{ki}\right]=0$$

$$\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ij}a_{jk}x_{ki}\right)=4\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n}b_{ij}^2$$

$$\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ij}x_{jk}x_{ki}\right]=0$$

$$\mathbb{V}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ij}x_{jk}x_{ki}\right)=4\sigma^4(n-2)|E|$$

Therefore:

$$\mathbb{E}\left[\frac{1}{6}\sum_{i=1}^{n}sum_u\right]=f^{\triangle}(G)$$

$$\mathbb{V}\left[\frac{1}{6}\sum_{i=1}^{n}sum_u\right]=\frac{4}{9}\sigma^2\sum_{i=1}^{n}\sum_{j=i+1}^{n}b_{ij}^2+\frac{1}{9}\sigma^4(n-2)|E|$$

In the proof of Theorem 6, we have $\sum_{i=1}^{n}\sum_{j=i+1}^{n}b_{ij}^2\leqslant nd_{\max}^3$, $|E|\leqslant nd_{\max}$, therefore:

$$\mathrm{MSE}\left[\frac{1}{6}\sum_{i=1}^{n}sum_u\right]\leqslant O(nd_{\max}^3+n^2d_{\max}).$$

E.11 Proof of Theorem 9

Since GNAM provides an ε_1-edge LDP and the *second randomizer* provides ε_2-edge LDP, therefore, according to the composition theorem, QuaMTR provides $(\varepsilon_1+\varepsilon_2)$-edge LDP.

E.12 Proof of Theorem 10

Align with Theorem 1, $\hat{A}=A+X$. For each u:

$$\begin{aligned}
&sum_u\\
&=\sum_{(i,j)\in W}\left(\hat{b}_{ij}-1\right)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}\left(\hat{b}_{ij}-1\right)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}\left(\sum_{k=1}^{n}\hat{a}_{ik}\hat{a}_{kj}-1\right)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}\left(\sum_{k=1}^{n}\left(a_{ik}+x_{ik}\right)\left(a_{kj}+x_{kj}\right)-1\right)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}(\sum_{k=1}^{n}a_{ik}a_{kj}+\sum_{k=1}^{n}a_{ik}x_{kj}+\sum_{k=1}^{n}x_{ik}a_{kj})\\
&+\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}(\sum_{k=1}^{n}x_{ik}x_{kj}-1)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{ij}a_{jk}a_{ku}-\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}\\
&+2\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{ij}a_{jk}x_{ku}+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{ij}x_{jk}x_{ku}
\end{aligned}$$

It should be noted that the physical meaning of $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{uj}a_{ik}a_{kj}$ is the number of ways node u can return to itself after 4 steps. From the perspective of taking four steps, $\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}$ encompasses two scenarios: one is $u\to i\to u\to i\to u$, which corresponds to the case where the same edge is traversed four times, specifically when $i=j$; the other scenario is $u\to i\to u\to j\to u$, which involves traversing one edge twice and then another edge twice, specifically when $i\neq j$. Therefore,

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{uj}a_{ik}a_{kj}-\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ui}a_{uj}=2f_u^{\square}.$$

Here, $f_u^{\square}$ represents the number of quadrangles that node u is part of. This is because subtracting the two scenarios where the path does not involve four distinct nodes from all possible four-step paths leaves us with paths that traverse a quadrangle. Since each quadrangle is traversed in both directions, the count is twice that of $f_u^{\square}$. Therefore:

$$\mathbb{E}\left[\frac{1}{8}\sum_{i=1}^{n}sum_u\right]=f^{\square}(G)$$

Then focus on the MSE (equals to Variance) part:

$$\begin{aligned}
&\mathbb{V}\left[\sum_{i=1}^{n}sum_u\right]\\
=&\mathbb{V}\left[2\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{ij}a_{jk}x_{ku}+\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}a_{ui}a_{ij}x_{jk}x_{ku}\right]
\end{aligned}$$

According to Lemma 1, we get:

$$\mathbb{V}\left[\sum_{i=1}^{n} sum_u\right]$$
$$=4\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}a_{jk}x_{ku}\right]+\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}x_{jk}x_{ku}\right]$$

The first item:

$$\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}a_{jk}x_{ku}\right]$$

Let c_{ij} represent the number of ways to reach node j from node i in exactly three steps, i.e. $c_{ij}=\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ik}a_{kl}a_{lj}$.

Similarly to Theorem 6, based on Lemma 1 and the symmetry of A and $\hat{A}$, we have the following:

$$\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}a_{jk}x_{ku}\right]$$
$$=\mathbb{V}\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} 2c_{ij}x_{ij}\right]$$
$$=4\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}^2\sigma^2$$

The second item:

$$\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}x_{jk}x_{ku}\right]$$

Let's make some changes to the above expression.

$$\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}x_{jk}x_{ku}\right]$$
$$=\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ui}a_{ij}\sum_{k=1}^{n} x_{jk}x_{ku}\right]$$
$$=\mathbb{V}\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} 2b_{ij}\sum_{k=1}^{n} x_{ik}x_{kj}\right]$$

From a graph perspective, $\sum_{k=1}^{n} x_{jk}x_{ki}$ represents the sum of the product of the adjacency relationships' noise between node i and node j with the remaining $n-2$ nodes. Therefore, for different pairs of nodes (i,j) and (l,m), $\sum_{k=1}^{n} x_{ik}x_{kj}$ and $\sum_{k=1}^{n} x_{lk}x_{km}$ satisfy the conditions of Lemma 1. Hence, we have:

$$\mathbb{V}\left[\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}x_{jk}x_{ku}\right]$$
$$=4(n-2)\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2\sigma^4$$

Then based on Lemma 1, we get:

$$\mathbb{V}\left[\sum_{i=1}^{n} sum_u\right]$$
$$=\mathbb{V}\left[2\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}a_{jk}x_{ku}+\sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ui}a_{ij}x_{jk}x_{ku}\right]$$
$$=16\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}^2\sigma^2+4(n-2)\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2\sigma^4$$

Therefore:

$$\mathrm{MSE}=\mathbb{V}\left[\frac{1}{8}\sum_{i=1}^{n} sum_u\right]$$
$$=\frac{1}{4}\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}^2\sigma^2+\frac{1}{16}(n-2)\sum_{i=1}^{n}\sum_{j=i+1}^{n} b_{ij}^2\sigma^4.$$

In the given graph, each edge (denoted as (i,j)) represents the product of the degrees of its two vertices, which physically signifies the number of c_{ij} that have this edge as the middle edge. Taking all the edges in the graph, we can obtain the total number of such c_{ij} as follows:

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}=\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{ij}\cdot d_i\cdot d_j\le nd_{\max}^3$$

The equation holds as an equality if and only if every node in the graph has $d_{\max}$ edges.

Due to the maximum value of each c_{ij} being only $d_i\cdot d_j$, this situation occurs when every neighbor of node i is connected to every neighbor of node j. Therefore, we have $c_{ij}\le d_{\max}^2$. To maximize $\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}^2$, we can set each $c_{ij}=d_{\max}^2$. As mentioned above, $\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}\le nd_{\max}^3$, and therefore there can be at most $nd_{\max}$ instances of $c_{ij}=d_{\max}^2$. At this point, $\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{ij}^2$ reaches its maximum value of $nd_{\max}^5$.

Therefore, MSE $\leqslant O(nd_{\max}^5+n^2d_{\max}^3)$.

E.13 Proof of Theorem 11

Let us first address Algorithm 7.

Denote neighbor adjacency list $\mathbf{a}'_u$ of $\mathbf{a}_u$, which is derived by either adding or removing a single neighbor from the original list of v_u, let k denote the index of the newly added neighbor. The resulting change in sum_u can be expressed as $\sum_{j\in\mathrm{Nei}_u}\hat{a}_{kj}$. Assuming that the newly added neighbor is connected by edges to all existing neighbors of user v_u, the resulting increment in sum_u attains its maximum value. By the Central Limit Theorem:

$$\sum_{j\in\mathrm{Nei}_u}\hat{a}_{kj}\overset{\text{approx}}{\sim}N(d_i,d_i\sigma^2)$$

The upper and lower β quantiles are given by:

$$\Phi^{-1}(1-\beta)\cdot\sqrt{d_i\sigma^2}+d_i$$
$$-\Phi^{-1}(1-\beta)\cdot\sqrt{d_i\sigma^2}+d_i$$

After Graph Projection on graph G, for any vertex $u\in[n]$, we have $\tilde{d_u}\geq d_u$. Therefore, the absolute values of the upper and lower β quantiles are less than:

$$\Phi^{-1}(1-\beta)\cdot\sqrt{\tilde{d_u}\sigma^2}+\tilde{d_u}.$$

Therefore:

$$\Pr\left[\sum_{j\in \mathrm{Nei}_u, j<i}\hat{a}_{ij}>\Delta f_u\right]<\beta$$
$$\Pr\left[\sum_{j\in \mathrm{Nei}_u, j<i}\hat{a}_{ij}<-\Delta f_u\right]<\beta$$

The change in sum_u resulting from removing a neighbor from the original neighbor list of v_u is less significant than the maximum change induced by adding a neighbor. Therefore, we focus solely on analyzing the impact of adding a neighbor on sum_u. By the same reasoning, the subsequent proofs will also restrict their discussion to the scenario of adding an edge to the adjacency list, as the case for removing an edge follows a similar logic.

Then deal with the Algorithm 8.

$$\hat{b}_{ij}=\sum_{k=1}^{n}\hat{a}_{ik}\hat{a}_{kj}=\sum_{k=1}^{n}a_{ik}a_{kj}+a_{ik}x_{kj}+x_{ik}a_{kj}+x_{ik}x_{kj}$$

According to Lemma 1, we get:

$$\begin{aligned}\mathbb{V}\left[\hat{b}_{ij}\right]&=\mathbb{V}\left[\sum_{j=k}^{n}a_{ik}x_{kj}\right]+\mathbb{V}\left[\sum_{j=1}^{n}x_{ik}a_{kj}\right]+\mathbb{V}\left[\sum_{j=1}^{n}x_{ik}x_{kj}\right]\\&=(d_i+d_j)\sigma^2+(n-2)\sigma^4\end{aligned}$$

Since for different k, the terms $\hat{a}_{ik}\hat{a}_{kj}$ are mutually independent, and their variances differ by at most $2\sigma^2$, it follows from the Central Limit Theorem for independent but non-identically distributed random variables that:

$$\hat{b}_{ij}\overset{\text{approx}}{\sim}N(b_{ij},(d_i+d_j)\sigma^2+(n-2)\sigma^4)$$

Its upper and lower β quantiles are given by:

$$\Phi^{-1}(1-\beta)\cdot\sqrt{(d_i+d_j)\sigma^2+(n-2)\sigma^4}+b_{ij}$$

$$-\Phi^{-1}(1-\beta)\cdot\sqrt{(d_i+d_j)\sigma^2+(n-2)\sigma^4}+b_{ij}$$

Similarly, their absolute values are both less than:

$$\Phi^{-1}(1-\beta)\cdot\sqrt{(n-2)\sigma^4+\left(\tilde{d_i}+\tilde{d_j}\right)\sigma^2}+\tilde{d_u}$$

Set $\Delta f_u=\Phi^{-1}(1-\beta)\cdot\sqrt{(n-2)\sigma^4+\left(\tilde{d_i}+\tilde{d}_{\max}\right)\sigma^2}+\tilde{d_u}$ for user v_u, we have:

$$\Pr\left[\hat{b}_{ui}>\Delta f_u\right]<\beta$$
$$\Pr\left[\hat{b}_{ui}<-\Delta f_u\right]<\beta$$

Then deal with the Algorithm 9. Readers may have noticed that calculating Δf_u does not require determining its exact distribution. Instead, one can use looser bounds on the expectation and variance of the distribution, which ensures that the determined Δf_u satisfies the two probability inequalities.

Any element in $\hat{B}$ satisfies:

$$\mathbb{E}\left[\hat{b}_{ij}\right]=b_{ij}\leqslant d_{\max}$$

$$\mathbb{V}\left[\hat{b}_{ij}\right]\leqslant 2d_{\max}\sigma^2+(n-2)\sigma^4$$

If a neighbor v_k is added, the change in sum_u can be expressed as:$\sum_{j\in \mathrm{Nei}_u}(\hat{b}_{kj}-1)$. Assuming that each b_{kj} is equal to $d_{\max}$, and the variance of each b_{kj} is $2d_{\max}\sigma^2+(n-2)\sigma^4$, the approximate distribution of $\hat{b}_{kj}$ can be described as follows:

$$N\left(d_{\max},\,2d_{\max}\sigma^2+(n-2)\sigma^4\right),$$

According to Lemma 1 and the properties of the sum of multivariate normal distributions, even though the $\hat{b}_{ij}$ are not independent, we still obtain: the approximate distribution of $\sum_{j\in \mathrm{Nei}_u}(\hat{b}_{kj}-1)$ is:

$$N(\tilde{d_u}(d_{\max}-1),\tilde{d_u}(2d_{\max}\sigma^2+(n-2)\sigma^4))$$

Therefore, set Δf_u as:

$$\Phi^{-1}(1-\beta)\cdot\sqrt{\tilde{d_u}(2d_{\max}\sigma^2+(n-2)\sigma^4)}+\tilde{d_u}(d_{\max}-1)$$

satisfies:

$$\Pr\left[\sum_{j\in \mathrm{Nei}_u, j<i}\hat{b}_{ij}>\Delta f_u\right]<\beta$$
$$\Pr\left[\sum_{j\in \mathrm{Nei}_u, j<i}\hat{b}_{ij}<-\Delta f_u\right]<\beta$$

E.14 Proof of Theorem 12

We proof the privacy guarantee firstly. If a user v_u adds a neighbor v_k, in line 2 of Algorithm 7, the maximum increase to sum_u would be $\sum_{j\in \mathrm{Nei}_u}\hat{a}_{kj}$. In line 2, the impact on sum_u is constrained to the range $[-\Delta f,\Delta f]$ by the clamp function. Consequently, the maximum change to sum_u caused by adding a node is also $\leq\Delta f_u$. Therefore, according to the Laplace mechanism, adding noise $\mathrm{Lap}(\Delta f/\varepsilon_2)$ ensures ε_2-edge LDP.

The multiplication by 2 at the end serves to ensure that the expected value of the final sum_u remains $2\hat{f}_u^{\Delta}$, thereby allowing it to be seamlessly integrated into Algorithm 3.

The composition theorem to be applied here is Proposition 1, where Δf_u is derived from the output ($tilded_u$) of the GraphProjection function. In GraphProjection, ε_0-edge LDP is provided, in GNAM, ε_1-edge LDP is provided, and

in Algorithm 7, ε_2-edge LDP is provided. Thus, according to Proposition 1, the overall system provides $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP.

The total MSE can be divided into four parts. The first part is the error introduced by GraphProjection; the second part is that in the second round, Less than β proportion of the data is clamped. The parameters affecting these two parts include α, β, and ε_0. Setting α and ε_0 to a larger value can reduce the number of edges removed. By setting β to a smaller value, the amount of data that is clamped can be minimized. These two parts have a relatively small impact on the final MSE.

The main part of the MSE comes from the third and fourth parts. The third part is the error inside the $\sum_{u=1}^{n} sum_u$, and the fourth part is the error introduced by the newly added $\mathrm{Lap}(\Delta f_u/\varepsilon_2)$ in the second round. In Theorem 8, we have already obtained the MSE of the third part. Therefore, we focus on the fourth part.

Because the Laplace noise added in the second round for each node is not only independent of the sum_u part but also independent among the Laplace noises themselves, the variance of this part can be calculated as the sum of the variances of each individual Laplace noise.

Let the Laplace noise added in the second round by node i be denoted as Y_i. For the sake of simplicity in the formula, let's denote $\Phi^{-1}(1-\beta)$ as γ. Once β is determined, γ is a constant and will not be too large. For example, when β is taken as $1/1000$, $\gamma \approx 3.09$.

$$\mathbb{V}[Y_i] = \frac{2}{\varepsilon_2^2}\left(\gamma^2\tilde{d}_i\sigma^2 + 2\gamma\tilde{d}_i\sqrt{\tilde{d}_i\sigma^2} + \tilde{d}_i^2\right)$$

In the previous analysis, if using RR or Laplace mechanism, we have $\sigma^2 = O\left(\frac{1}{\varepsilon_1^2}\right)$, and we have $\tilde{d}_i = O(d_i)$, therefore:

$$\mathbb{V}[Y_i] = O\left(\frac{d_i}{\varepsilon_1^2\varepsilon_2^2} + \frac{d_i^2}{\varepsilon_2^2}\right)$$

Therefore:

$$\mathbb{V}\left[\sum_{i=1}^{n} Y_i\right] = O\left(\frac{|E|}{\varepsilon_1^2\varepsilon_2^2} + \sum_{i=1}^{n}\frac{d_i^2}{\varepsilon_2^2}\right) \leqslant O\left(\frac{|E|}{\varepsilon_1^2\varepsilon_2^2} + n\frac{d_{\max}^2}{\varepsilon_2^2}\right)$$

Combining this with Theorem 8, and $\sigma^2 = O(1/\varepsilon_1^2)$, we get:

$$\mathrm{MSE} \leqslant O\left(\frac{nd_{\max}^3}{\varepsilon_1^2} + \frac{|E|}{\varepsilon_1^2\varepsilon_2^2} + n\frac{d_{\max}^2}{\varepsilon_2^2}\right)$$

E.15 Proof of Theorem 13

We proof the privacy guarantee firstly. In essence, during the second round, user v_u is presented with a sequence of data $(\hat{b}_{1u}, \hat{b}_{2u}, ..., \hat{b}_{nu})$, from which v_u selects d_u numbers to sum and upload. The privacy to be protected here lies in the specific choice of numbers selected by v_u. After applying the clamp function, the range of these numbers is effectively constrained to $[-\Delta f, \Delta f]$. If user v_u gains or loses a neighbor, it essentially means adding or removing one number from this sequence. Since the impact of such a change on sum_u is bounded by Δf, the mechanism ensures ε_2-edge LDP.

Following a reasoning similar to the proof of Theorem 12, and by applying Proposition 1, the overall system provides $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP.

Then the total MSE.

Because the value of $\hat{b}_{ui}$ after clamping will not exceed Δf_u, the second round will provide ε_2-edge LDP. By combining this with the composition theorem, TriMTR will provide a total of $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP.

Using the same notation as in Theorem 14, let Y_i be the Laplace noise added to node i in the second round, and $\gamma = \Phi^{-1}(1-\beta)$. We have:

$$\begin{aligned}\mathbb{V}[Y_i] =& \frac{2}{\varepsilon_2^2}\gamma^2\left((n-1)\right)\sigma^4 + \left(\tilde{d}_u + \tilde{d}_{\max}\right)\sigma^2) \\ &+ \frac{4}{\varepsilon_2^2}\gamma\tilde{d}_i\sqrt{(n-1)\sigma^4 + \left(\tilde{d}_u + \tilde{d}_{\max}\right)\sigma^2} + \frac{2}{\varepsilon_2^2}\tilde{d}_i^2\end{aligned}$$

In the previous analysis, if using RR or Laplace mechanism, we have $\sigma^2 = O\left(\frac{1}{\varepsilon_1^2}\right)$, and we have $\tilde{d}_i = O(d_i)$, therefore:

$$\mathbb{V}[Y_i] = O\left(\frac{n}{\varepsilon_1^4\varepsilon_2^2} + \frac{d_i^2}{\varepsilon_2^2}\right)$$

Therefore:

$$\mathbb{V}\left[\sum_{i=1}^{n} Y_i\right] = O\left(\frac{n^2}{\varepsilon_1^4\varepsilon_2^2} + \sum_{i=1}^{n}\frac{d_i^2}{\varepsilon_2^2}\right) \leqslant O\left(\frac{n^2}{\varepsilon_1^4\varepsilon_2^2} + n\frac{d_{\max}^2}{\varepsilon_2^2}\right)$$

Combining this with Theorem 8, and $\sigma^2 = O(1/\varepsilon_1^2)$, we get:

$$\mathrm{MSE} \leqslant O\left(\frac{nd_{\max}^3}{\varepsilon_1^2} + \frac{n^2 d_{\max}}{\varepsilon_1^4} + \frac{n^2}{\varepsilon_1^4\varepsilon_2^2} + n\frac{d_{\max}^2}{\varepsilon_2^2}\right)$$

E.16 Proof of Theorem 14

The method to prove the provision of $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP is similar to the methods used in Theorem 2, so we will not elaborate further here. Next, we will bound its MSE.

Using the same notation as in Theorem 14, let Y_i be the Laplace noise added to node i in the second round, and $\gamma = \Phi^{-1}(1-\beta)$. We have:

$$\begin{aligned}\mathbb{V}[Y_i] =& \frac{2}{\varepsilon_2^2}\gamma^2\tilde{d}_i\left((n-1)\sigma^4 + 2\tilde{d}_{\max}\sigma^2\right) \\ &+ \frac{4}{\varepsilon_2^2}\gamma(\tilde{d}_{\max}-1)\tilde{d}_i^{\frac{3}{2}}\sqrt{\left((n-1)\sigma^4 + 2\tilde{d}_{\max}\sigma^2\right)} \\ &+ \frac{2}{\varepsilon_2^2}(\tilde{d}_{\max}-1)^2\tilde{d}_i^2\end{aligned}$$

In the previous analysis, if using RR or Laplace mechanism, we have $\sigma^2 = O\left(\frac{1}{\varepsilon_1^2}\right)$, and we have $\tilde{d}_i = O(d_i)$, therefore:

$$\mathbb{V}[Y_i] = O\left(\frac{nd_i}{\varepsilon_1^4\varepsilon_2^2} + \frac{d_i^2 d_{\max}^2}{\varepsilon_2^2}\right)$$

Therefore:

$$\mathbb{V}\left[\sum_{i=1}^n Y_i\right] = O\left(\sum_{i=1}^n \frac{nd_i}{\varepsilon_1^4\varepsilon_2^2} + \sum_{i=1}^n \frac{d_i^2 d_{\max}^2}{\varepsilon_2^2}\right) \leqslant O\left(\frac{n^2 d_{\max}}{\varepsilon_1^4\varepsilon_2^2} + n\frac{d_{\max}^4}{\varepsilon_2^2}\right)$$

Combining this with Theorem 8, and $\sigma^2 = O(1/\varepsilon_1^2)$, we get:

$$\text{MSE} \leqslant O\left(\frac{nd_{\max}^5}{\varepsilon_1^2} + \frac{n^2 d_{\max}^3}{\varepsilon_1^4} + \frac{n^2 d_{\max}}{\varepsilon_1^4\varepsilon_2^2} + n\frac{d_{\max}^4}{\varepsilon_2^2}\right)$$

E.17 Proof of Theorem 15

Given the assumption that the clustering coefficient of a graph remains constant, the number of triangles in the graph is proportional to the number of 2-stars in the graph. This relationship can be denoted as $f^{\triangle}(G) = k\#\text{2-star}$.

Considering a 2-star to be composed of a central node and its two incident edges, the total number of 2-stars in the graph can be calculated by the following formula:

$$\#\text{2-star} = \sum_{i=1}^n d_i \cdot (d_i - 1) = \sum_{i=1}^n d_i^2 - 2|E| \approx \sum_{i=1}^n d_i^2$$

Therefore, $f^{\triangle}(G) \approx k\sum_{i=1}^n d_i^2$.

Here, we focus specifically on the impact of the privacy budget and the structure of the graph (number of nodes n and the degree values d_i) on convergence. We primarily consider the order of magnitude and treat other terms as constant coefficients. Therefore, we get $f^{\triangle}(G) = O(\sum_{i=1}^n d_i^2)$.

First, we address TriTR, whose more precise expression for MSE is:

$$\text{MSE} \leqslant O\left(\frac{\sum_{i=1}^n \sum_{j=i+1}^n b_{ij}^2}{\varepsilon_1^2} + \frac{|E|}{\varepsilon_1^2\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}\right)$$

Because $b_{ij} = \sum_{k=1}^n a_{ik}a_{kj} \leqslant \sum_{k=1}^n a_{ik} = d_i$, Therefore:

$$\sum_{i=1}^n \sum_{j=i+1}^n b_{ij}^2 \leqslant \sum_{i=1}^n \sum_{j=i+1}^n d_i^2 \leqslant n\sum_{i=1}^n d_i^2$$

Therefore:

$$\text{MSE} \leqslant O\left(\frac{n\sum_{i=1}^n d_i^2}{\varepsilon_1^2} + \frac{\sum_{i=1}^n d_i}{\varepsilon_1^2\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}\right)$$

According to Jensen's inequality, we have:

$$ABE \leqslant O\left(\sqrt{\frac{n\sum_{i=1}^n d_i^2}{\varepsilon_1^2} + \frac{\sum_{i=1}^n d_i}{\varepsilon_1^2\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}}\right)$$

Then the RE:

$$\begin{aligned}
RE &\leqslant O\left(\frac{\sqrt{\frac{n\sum_{i=1}^n d_i^2}{\varepsilon_1^2} + \frac{\sum_{i=1}^n d_i}{\varepsilon_1^2\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}}}{\sum_{i=1}^n d_i^2}\right) \\
&= O\left(\sqrt{\frac{\frac{n\sum_{i=1}^n d_i^2}{\varepsilon_1^2} + \frac{\sum_{i=1}^n d_i}{\varepsilon_1^2\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}}{\left(\sum_{i=1}^n d_i^2\right)^2}}\right) \\
&\leqslant O\left(\sqrt{\frac{n}{\varepsilon_1^2\sum_{i=1}^n d_i^2}} + \sqrt{\frac{\sum_{i=1}^n d_i}{\varepsilon_1^2\varepsilon_2^2\left(\sum_{i=1}^n d_i^2\right)^2}} + \sqrt{\frac{1}{\varepsilon_2^2\sum_{i=1}^n d_i^2}}\right)
\end{aligned}$$

Because $nd_{\text{avg}}^2 = \sum_{i=1}^n d_{\text{avg}}^2 \leqslant \sum_{i=1}^n d_i^2$, with equality if and only if $d_i = d_{\text{avg}}$ for any $i \in [n]$. Therefore:

$$\begin{aligned}
RE &\leqslant O\left(\sqrt{\frac{n}{\varepsilon_1^2 nd_{\text{avg}}^2}} + \sqrt{\frac{nd_{\text{avg}}}{\varepsilon_1^2\varepsilon_2^2 n^2 d_{\text{avg}}^4}} + \sqrt{\frac{1}{\varepsilon_2^2 nd_{\text{avg}}^2}}\right) \\
&\leqslant O\left(\sqrt{\frac{1}{\varepsilon_1^2 d_{\text{avg}}^2}} + \sqrt{\frac{1}{\varepsilon_1^2\varepsilon_2^2 nd_{\text{avg}}^3}} + \sqrt{\frac{1}{\varepsilon_2^2 nd_{\text{avg}}^2}}\right) \\
&= O\left(\frac{1}{\varepsilon_1 d_{\text{avg}}} + \frac{1}{\varepsilon_1\varepsilon_2\sqrt{n}d_{\text{avg}}^{\frac{3}{2}}} + \frac{1}{\varepsilon_2\sqrt{n}d_{\text{avg}}}\right)
\end{aligned}$$

Next, we address TriMTR. Because $nd_{\text{avg}} = 2|E|$, the more precise expression for the MSE of TriTR is:

$$\text{MSE} \leqslant O\left(\frac{\sum_{i=1}^n \sum_{j=i+1}^n b_{ij}^2}{\varepsilon_1^2} + \frac{n^2 d_{\text{avg}}}{\varepsilon_1^4} + \frac{n^2}{\varepsilon_1^4\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}\right)$$

Using a similar scaling method as TriTR, we obtain:

$$\text{MSE} \leqslant O\left(\frac{n\sum_{i=1}^n d_i^2}{\varepsilon_1^2} + \frac{n^2 d_{\text{avg}}}{\varepsilon_1^4} + \frac{n^2}{\varepsilon_1^4\varepsilon_2^2} + \frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}\right)$$

Therefore, the Relative Err:

$$
\begin{aligned}
RE &\leqslant O\left(\sqrt{\frac{\frac{n\sum_{i=1}^n d_i^2}{\varepsilon_1^2}+\frac{n^2 d_{\text{avg}}}{\varepsilon_1^4}+\frac{n^2}{\varepsilon_1^4\varepsilon_2^2}+\frac{\sum_{i=1}^n d_i^2}{\varepsilon_2^2}}{\left(\sum_{i=1}^n d_i^2\right)^2}}\right) \\
&\leqslant O\left(\frac{1}{\varepsilon_1}\sqrt{\frac{n}{nd_{\text{avg}}^2}}+\frac{1}{\varepsilon_1^2}\sqrt{\frac{n^2 d_{\text{avg}}}{n^2 d_{\text{avg}}^4}}+\frac{1}{\varepsilon_1^2\varepsilon_2}\sqrt{\frac{n^2}{n^2 d_{\text{avg}}^4}}+\frac{1}{\varepsilon_2}\sqrt{\frac{1}{nd_{\text{avg}}^2}}\right) \\
&= O\left(\frac{1}{\varepsilon_1}\sqrt{\frac{1}{d_{\text{avg}}^2}}+\frac{1}{\varepsilon_1^2}\sqrt{\frac{1}{d_{\text{avg}}^3}}+\frac{1}{\varepsilon_1^2\varepsilon_2}\sqrt{\frac{1}{d_{\text{avg}}^4}}+\frac{1}{\varepsilon_2}\sqrt{\frac{1}{nd_{\text{avg}}^2}}\right) \\
&= O\left(\frac{1}{\varepsilon_1 d_{\text{avg}}}+\frac{1}{\varepsilon_1^2 d_{\text{avg}}^{\frac{3}{2}}}+\frac{1}{\varepsilon_1^2\varepsilon_2 d_{\text{avg}}^2}+\frac{1}{\varepsilon_2\sqrt{n}d_{\text{avg}}}\right)
\end{aligned}
$$

E.18 Proof of Theorem 16

Proof. Given the true number:

$$
f^{2\text{-star}}(G)=\sum_{u=1}^n d_u(d_u-1)
$$

Let $Y_u \sim \text{Lap}(1/\varepsilon_0)$, then $\mathbb{E}[Y_u]=0$, $\mathbb{E}[Y_u^2]=\mathbb{V}[Y_u]=\frac{2}{\varepsilon_0^2}$, $\mathbb{E}[Y_u^3]=0$, $\mathbb{E}[Y_u^4]=\frac{24}{\varepsilon_0^4}$, the first property:

$$
\begin{aligned}
&\mathbb{E}[\hat{f}^{2\text{-star}}(G)] \\
=&\mathbb{E}[\sum_{u=1}^n\left[(d_u+Y_u)(d_u-1+Y_u)-\frac{2}{\varepsilon_0^2}\right]] \\
=&\sum_{u=1}^n\left[\mathbb{E}\left[(d_u+Y_u)(d_u-1+Y_u)\right]-\frac{2}{\varepsilon_0^2}\right] \\
=&\sum_{u=1}^n\left[d_u(d_u-1)+(2d_u-1)\mathbb{E}[Y_u]+\mathbb{E}[Y_u^2]-\frac{2}{\varepsilon_0^2}\right] \\
=&\sum_{u=1}^n d_u(d_u-1) \\
=&f^{2\text{-star}}(G)
\end{aligned}
$$

Second property:

$$
\begin{aligned}
&\mathbb{V}(\hat{f}^{2\text{-star}}(G)) \\
=&\mathbb{V}(\sum_{u=1}^n\left[(d_u+Y_u)(d_u-1+Y_u)-\frac{2}{\varepsilon_0^2}\right]) \\
=&\sum_{u=1}^n\left[\mathbb{V}\left((d_u+Y_u)(d_u-1+Y_u)\right)-\frac{2}{\varepsilon_0^2}\right] \\
=&\sum_{u=1}^n\left[\mathbb{V}((2d_u-1)Y_u+Y_u^2)\right] \\
=&\sum_{u=1}^n\left[\mathbb{E}[(2d_u-1)Y_u+Y_u^2)^2]-\frac{4}{\varepsilon_0^4}\right] \\
=&\sum_{u=1}^n\left[\mathbb{E}[(2d_u-1)^2Y_u^2+2(2d_u-1)Y_u^3+Y_u^4]-\frac{4}{\varepsilon_0^4}\right] \\
=&\sum_{u=1}^n\left[\mathbb{E}[(2d_u-1)^2Y_u^2]+\mathbb{E}[Y_u^4]-\frac{4}{\varepsilon_0^4}\right] \\
=&\sum_{u=1}^n\left[2(2d_u-1)^2\frac{1}{\varepsilon_0^2}+\frac{20}{\varepsilon_0^4}\right] \\
=&\frac{8}{\varepsilon_0^2}\sum_{u=1}^n d_u^2-\frac{16}{\varepsilon_0^2}|E|+\frac{2}{\varepsilon_0^2}n+\frac{20}{\varepsilon_0^4}n
\end{aligned}
$$

Treating ε_0 as a constant, we obtain:

$$
\text{MSE}=O(\sum_{u=1}^n d_u^2)=O(\#2\text{-star})
$$

Thus:

$$
\text{RE}=O(\frac{\sqrt{\#2\text{-star}}}{\#2\text{-star}})=O(\frac{1}{\sqrt{\#2\text{-star}}})
$$