Triangle Counting over Signed Graphs with Differential Privacy

Zening Li, Rong-Hua Li, Fusheng Jin

Beijing Institute of Technology, Beijing, China

zening-li@outlook.com; lironghuabit@126.com; jfs21cn@bit.edu.cn

***Abstract*—Triangle counting serves as a foundational operator in graph analysis. Since graph data often contain sensitive information about entities, the release of triangle counts poses privacy concerns. While recent studies have addressed privacy-preserving triangle counting, they mainly concentrate on unsigned graphs. In this paper, we investigate a new problem of developing triangle counting algorithms for signed graphs that adhere to centralized differential privacy and local differential privacy, respectively. The inclusion of edge signs and more classes of triangles leads to increased complexity and overwhelms the statistics with noise. To overcome these problems, we first propose a novel algorithm for smooth-sensitivity computation to achieve differential privacy under the centralized model. In addition, to handle large signed graphs, we devise a computationally efficient function that calculates a smooth upper bound on local sensitivity. Finally, we release the approximate triangle counts after the introduction of Laplace noise, which is calibrated to the smooth upper bound on local sensitivity. In the local model, we propose a two-phase framework tailored for balanced and unbalanced triangle counting. The first phase utilizes the Generalized Randomized Response mechanism to perturb data, followed by a novel response mechanism in the second phase. Extensive experiments conducted over real-world datasets demonstrate that our proposed methods can achieve an excellent trade-off between privacy and utility.**

I. Introduction

Many relationships in the real world can be represented by signed graphs with positive and negative edges, as a result of which signed graph analysis has attracted much attention and nurtured numerous applications [1]–[7]. However, most real-world signed graphs associated with people or human-related activities, such as social and financial networks, hold sensitive information about the relationships between individuals. These positive and negative links reveal details about individual interactions, preferences, and social dynamics. Given the private nature of such data and the pressure to allow controlled release of private data, there is a considerable interest in how to release information with assured privacy.

Differential privacy (DP) [8] has been the dominant model for the protection of individual privacy from powerful and realistic attackers. This model works well for data published as histograms or counts. However, approaches that attempt to directly output a modified version of the input signed graph under DP result in the loss of its inherent properties. The major technical obstacle lies in the fact that the direct application of perturbation mechanism seems to require extensive random modifications to the input signed graphs. As an alternative, we can focus on the release of statistical properties of the signed graph under differential privacy rather than the signed graph itself. Triangle counting is a fundamental primitive in signed graph analysis. A triangle is considered balanced if it contains an odd number of positive edges and unbalanced otherwise [9] (refer to Figure 2 for illustration). Statistics based on balanced and unbalanced triangle counts can reveal important structural information about signed graphs. It is valuable to publish these counts, because there are many stochastic signed graph models which rely on these statistics to construct graphs, such as the Balanced Signed Chung-Lu model [9] and 2-Layer Signed Network model [5]. In addition, these statistics also support various computational tasks in signed graph analysis, such as link prediction [1] and community detection [4], [10].

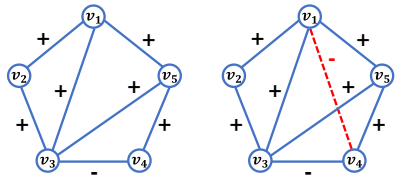


Fig. 1: Illustrations of (a) a financial network and (b) its local sensitivity at distance of the triangle counting query function.

The direct release of exact balanced and unbalanced triangle counts can inadvertently expose sensitive relationship patterns, posing significant privacy risks. For example, consider a financial network, as depicted in Figure 1(a), where nodes represent financial institutions and edges denote lending relationships. The sign of an edge reflects the transaction's health: a positive sign denotes timely repayments, while a negative sign indicates defaults. Suppose institution v_2 knows its own connections and all links (and their signs) between $v_3, \ldots, v_5$. If v_2 seeks to infer the relationships between v_1 and other nodes, the release of exact triangle counts (#balanced=2 and #unbalanced=1) can reveal sensitive information. Specifically, v_2 can deduce that v_1 and v_3 are connected by a positive link, and the node v_1 must be connected to exactly one of v_4 or v_5, with (v_1, v_4) being negative and (v_1, v_5) being positive. Such disclosures compromise the privacy of financial institutions. Thus, developing privacy-preserving triangle counting algorithms is essential.

Differential privacy is commonly classified into two models based on its architectural framework: centralized DP and local DP. In the centralized model, there exists a trusted curator who holds the sensitive data and releases sanitized versions of the statistics. In contrast, local DP assumes that the data curator is untrusted. Under this premise, individual users perturb their

2025 IEEE 41st International Conference on Data Engineering (ICDE) | 979-8-3315-3603-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDE65448.2025.00159

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00159

data before transmission to the curator. Extensive research has explored differentially private algorithms for graph analysis under centralized [11]–[16] and local models [17]–[22]. Despite the strides made in this field, the focus has predominantly been on the unsigned graphs. The exploration of balanced and unbalanced triangle counting in signed graphs, the focus of this paper, remains an underexplored area in the current literature.

To fill this gap, we investigate the problem of balanced and unbalanced triangle counting under centralized DP and local DP, where both the graph structure and the sign of edges are considered private information. However, the inclusion of edge signs introduces unique challenges for privacy preservation. (i) *Noise scale*: a standard DP technique directly adds noise to the statistical estimates. However, as the noise scale is proportional to the number of nodes, it can cause excessive perturbation to the release statistics. (ii) *Complexity of smooth sensitivity*: to reduce the introduction of noise, we explore the technique of smooth sensitivity as an alternative. However, the computation of smooth sensitivity is obviously more complex in signed graphs. (iii) *Scalability*: beyond the computational complexity, the calculation of smooth sensitivity is extremely expensive.

To solve the mentioned issues, we propose several effective methods under centralized DP and local DP, respectively. In the centralized model, in order to reduce the introduction of noise, we adopt the smooth sensitivity framework to calculate the number of balanced and unbalanced triangles in the signed graph. More specifically, we first analyze the local sensitivity and local sensitivity at distance for the triangle counting query. Then, we propose a novel and efficient algorithm to compute smooth sensitivity, with time complexity of $\mathcal{O}(m \cdot d_{max} + n^2)$ In addition, to handle large graphs, we propose a computationally efficient function that could calculate a smooth upper bound on local sensitivity, with a reduced time complexity of $\mathcal{O}(m \cdot d_{max})$. Furthermore, we propose a degree-based pruning technique to further reduce the computational cost and memory consumption for the computation of smooth upper bound. The final step is to release the triangle counts after adding Laplace noise.

In the local model, we propose a two-phase framework tailored for balanced and unbalanced triangle counting. The first phase uses the Generalized Randomized Response mechanism to perturb the data. Then, in the second phase, we propose a novel response mechanism with an unbiased correction to the final statistics. This mechanism injects noise into the query response of each node, which is also calibrated to the smooth upper bound on local sensitivity. At the end, to evaluate the effectiveness of our proposed methods, we conduct extensive experiments on six real-world datasets. The results show that the proposed approaches establish state-of-the-art performance and achieve a superior privacy-utility trade-off. To summarize, the main contributions of this paper are as follows:

- We propose innovative approaches to estimate balanced and unbalanced triangle counts under centralized and local DP. To the best of our knowledge, this is the first exploration of privacy-preserving triangle counting in signed graphs.
- In the centralized model, we propose an efficient approach for smooth sensitivity computation. To handle large signed graphs, we derive a smooth upper bound on local sensitivity for triangle counting queries, which is computationally more efficient and reduces the time complexity from $\mathcal{O}(m \cdot d_{max} + n^2)$ to $\mathcal{O}(m \cdot d_{max})$. We also propose a degree-based pruning technique to further reduce the computational cost and memory usage for smooth upper bound computation.
- In the local model, we propose a two-phase framework for triangle counting. We also derive a smooth upper bound on local sensitivity for the user side and propose a response mechanism to report the statistics in the second phase.
- We conduct extensive experiments across several real-world datasets. The experimental results show that our proposed methods achieve excellent privacy-utility trade-offs in centralized and local models and exhibit efficient performance.

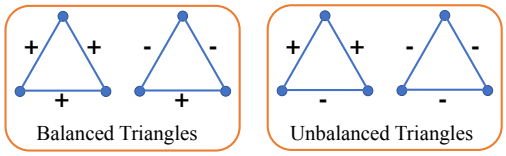


Fig. 2: Examples of balanced and unbalanced triangles.

II. Preliminaries

A. Problem Statement

Consider an undirected signed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ represents the set of n nodes and $\mathcal{E}$ comprises m edges. Each edge in G carries a sign "+" or "-". An edge with the sign "+" denotes a positive edge, whereas an edge with the sign "-" represents a negative edge. The degree of v_i is denoted by d_i, and the maximum degree is denoted by d_{max}. Let $\mathcal{G}$ be the collection of all possible signed graphs on n nodes. Any signed graph $G \in \mathcal{G}$ can also be represented as a symmetric adjacency matrix $\mathbf{A}$, where $a_{ij} = +1$ indicates a positive edge, $a_{ij} = -1$ denotes a negative edge, and $a_{ij} = 0$ implies no edge between nodes v_i and v_j. The adjacency vector $\boldsymbol{a}_i$, which corresponds to the i-th row of $\mathbf{A}$, captures the connectivity profile of vertex v_i with respect to all other vertices.

In this paper, we consider a query function $f_{\triangle}: \mathcal{G} \to (T^b, T^u)$ that takes a signed graph G as input and outputs the number of balanced and unbalanced triangles, represented as T^b and T^u respectively. Note that we assume the set of nodes $\mathcal{V}$ is public, and hence the identities of the nodes are non-private. Similar assumptions are widely utilized in previous research [11]–[13], [20], [22]. Conversely, the edge set $\mathcal{E}$ and the signs of edges are considered sensitive information. Our objective is to develop algorithms that can accurately estimate the number of balanced and unbalanced triangles while protecting individual privacy.

B. Differential Privacy for Signed Graphs

Differential privacy [8] has become the standard framework for data release that provides robust privacy protection in the presence of powerful and realistic adversaries. A randomized algorithm satisfies differential privacy when the distributions of its outputs are similar for any pair of neighboring databases. Therefore, the formal definition of differential privacy revolves around the concept of neighboring databases. In this paper, we utilize the edit distance to measure the distinction between two

databases. The edit distance quantifies the minimum number of operations required to transform one object into another. We exclusively focus on edge operations, since the graph structure and edge signs are considered private information.

Centralized Differential Privacy. In the centralized model, a trusted curator first collects the private data from all users and then releases the sanitized statistical data.

Definition 1 (Distance between signed graphs, Neighbors)**.** *The distance between n-node signed graphs G and $\tilde{G}$, denoted as $d(G, \tilde{G})$, is the minimum number of primitive operations required to convert G to $\tilde{G}$. There are three types of primitive edit operations: (1) edge insertion to introduce a new signed edge between a pair of nodes; (2) edge deletion to remove a single edge between a pair of nodes; (3) edge substitution to alter the sign of a given edge. Signed graphs G and $\tilde{G}$ are considered neighbors if $d(G, \tilde{G}) = 1$.*

Definition 2 (Signed Edge Centrialized Differential Privacy)**.** *Given $\epsilon > 0$ and $\delta > 0$, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-signed edge CDP if and only if for any two neighboring signed graphs $G, \tilde{G} \in \mathcal{G}$, and for any possible output $O \in Range(\mathcal{A})$, we have $\Pr[\mathcal{A}(G) = O] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(\tilde{G}) = O] + \delta$.*

The parameter ϵ is called the privacy budget that controls the trade-off between privacy protection level and utility. A lower privacy budget indicates stronger privacy protection but poorer utility. The parameter δ is treated as the probability of failure and is usually chosen to be much smaller than the inverse of the number of data records. When $\delta = 0$, the algorithm satisfies ϵ-signed edge centralized differential privacy.

Local Differential Privacy. In contrast to centralized DP, local differential privacy [23] is predicated on a local model wherein the data curator is not trusted. In this model, each user applies a randomized perturbation algorithm locally to his or her data and then sends the perturbed data to the untrusted data curator.

Definition 3 (Distance between adjacency vectors, Neighbors)**.** *The distance between two adjacency vectors $\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i$, denoted as $d(\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i)$, is defined as the least number of elementary operations required to transform $\boldsymbol{a}_i$ into $\tilde{\boldsymbol{a}}_i$. These operations include (1) the insertion of a signed edge connected to v_i; (2) the removal of an edge from node v_i; (3) the alteration of the sign of an edge linked to v_i. Adjacency vectors $\boldsymbol{a}_i$ and $\tilde{\boldsymbol{a}}_i$ are neighbors when $d(\boldsymbol{a}_i, \tilde{\boldsymbol{a}}_i) = 1$.*

Definition 4 (Signed Edge Local Differential Privacy)**.** *Given $\epsilon > 0$ and $\delta > 0$, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-signed edge LDP if and only if for any two neighboring adjacency vectors $\boldsymbol{a}_i$ and $\tilde{\boldsymbol{a}}_i$, and for any possible output O of $\mathcal{A}$, we have $\Pr[\mathcal{A}(\boldsymbol{a}_i) = O] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(\tilde{\boldsymbol{a}}_i) = O] + \delta$.*

C. Differential Privacy Mechanisms

Numerous approaches have been proposed to achieve differential privacy. One prevalent approach is to introduce carefully calibrated random noise to the actual results. In this article, we employ the Laplace distribution to add noise. A Laplace random variable with mean 0 and standard deviation $\sqrt{2}\lambda$ has density $h(z) = \frac{1}{2\lambda} e^{-|z|/\lambda}$, expressed as $\mathrm{Lap}(\lambda)$. The Laplace mechanism [8], a fundamental technique that satisfies ϵ-DP, introduces independent and identically distributed noise from $\mathrm{Lap}(GS_f/\epsilon)$ to each element of the output produced by a query function $f\colon \mathcal{D} \to \mathbb{R}^k$, where GS_f is the global sensitivity of f.

Definition 5 (Global Sensitivity [8])**.** *Given a query function $f\colon \mathcal{D} \to \mathbb{R}^k$, the global sensitivity of f is defined as $GS_f = \max_{x, \tilde{x} \in \mathcal{D}: d(x,\tilde{x})=1} ||f(x) - f(\tilde{x})||_1$, where $||\cdot||_1$ is the ℓ_1 norm.*

The amount of noise added by the Laplace mechanism depends on GS_f and the privacy budget ϵ. It is crucial to note that GS_f is an inherent characteristic of the function f, independent of any input data. However, for the query functions considered in this article, this mechanism introduces a substantial amount of noise to the results, which can compromise the performance. In [11], the authors propose a local measure of sensitivity.

Definition 6 (Local Sensitivity [11])**.** *The local sensitivity of a function $f\colon \mathcal{D} \to \mathbb{R}^k$ on a database $x \in \mathcal{D}$ can be expressed as $LS_f(x) = \max_{\tilde{x} \in \mathcal{D}: d(x,\tilde{x})=1} ||f(x) - f(\tilde{x})||_1$.*

It is evident that the local sensitivity depends not only on the query function f, but also on the concrete instance. For many problems, $LS_f(x)$ tends to be smaller than GS_f [11], [12], [14], [24]. Unfortunately, local sensitivity does not satisfy the requirement of differential privacy, because $LS_f(x)$ itself contains information about the database. To solve this problem, Nissim et al. [11] employ a β-smooth upper bound on local sensitivity, rather than the local sensitivity itself, for the noise calibration. Specifically, for $\beta > 0$, a function $S_{f,\beta}\colon \mathcal{D} \to \mathbb{R}$ is a β-smooth upper bound on local sensitivity of f, if $\forall x \in \mathcal{D}$, $S_{f,\beta}(x) \geq LS_f(x)$ and $\forall x, \tilde{x} \in \mathcal{D}$ with $d(x, \tilde{x}) = 1$, $S_{f,\beta}(x) \leq e^{\beta} S_{f,\beta}(\tilde{x})$. $LS_f(x)$ may have multiple smooth bounds, and the smooth sensitivity is the smallest one that meets the condition.

Definition 7 (Smooth Sensitivity [11])**.** *For any query function $f\colon \mathcal{D} \to \mathbb{R}^k$ and $\beta > 0$, the β-smooth sensitivity of f at $x \in \mathcal{D}$ is defined as $S^*_{f,\beta}(x) = \max_{\tilde{x} \in \mathcal{D}} (e^{-\beta d(x,\tilde{x})} \cdot LS_f(\tilde{x}))$.*

To calculate the smooth sensitivity efficiently, we introduce a function referred to as the local sensitivity at distance t, denoted as $LS_f(x, t)$. This function is calculated as $LS_f(x, t) = \max_{\tilde{x} \in \mathcal{D}: d(x,\tilde{x}) \leq t} LS_f(\tilde{x})$. Here, $LS_f(x, t)$ is the maximum local sensitivity $LS_f(\tilde{x})$ of all databases $\tilde{x}$ where the maximum distance between $\tilde{x}$ and x is t. In other words, we permit at most t modifications to the database x before computing its local sensitivity. Then the β-smooth sensitivity $S^*_{f,\beta}(x)$ can be expressed in terms of $LS_f(x, t)$ as $S^*_{f,\beta}(x) = \max_{t \in [0,n]} e^{-\beta t} LS_f(x, t)$.

Theorem 1 (Noise Calibration to Smooth Bound [11])**.** *Given a query function $f\colon \mathcal{D} \to \mathbb{R}^k$, suppose $S_{f,\beta}(x)$ is a β-smooth upper bound on local sensitivity of f, where $\beta = \frac{\epsilon}{4(k + \ln(2/\delta))}$. Then the algorithm $\mathcal{A}_{f,\epsilon}(x) = f(x) + (z_1, \ldots, z_k)$ satisfies (ϵ, δ)-DP, where the z_i are drawn i.i.d. from $\mathrm{Lap}(2S_{f,\beta}(x)/\epsilon)$.*

Generalized Randomized Response (GRR) [25] is another perturbation mechanism that satisfies ϵ-LDP. This mechanism returns the true value x with probability $\frac{e^{\epsilon}}{e^{\epsilon} + k - 1}$, or the value $\tilde{x} \neq x$ with probability $\frac{1}{e^{\epsilon} + k - 1}$, where k is the domain size.

III. The Proposed Centralized DP Method

In this section, we propose novel approaches to estimate the number of balanced and unbalanced triangles in signed graphs under centralized DP. Our methods not only protect individual privacy but also maintain excellent query performance. First of all, we introduce techniques to compute the local sensitivity and the local sensitivity at distance t for the triangle counting query function, which is the foundation for the implementation of differential privacy. Then, we propose efficient algorithms for the calculation of smooth upper bounds on local sensitivity, a critical factor in reducing the impact of data perturbations. Among these bounds, smooth sensitivity is the smallest upper bound. However, its calculation becomes impractical for large signed graphs due to the enormous computational overhead. In such cases, we select a computationally feasible smooth upper bound as an alternative to smooth sensitivity. Thus, we devise a computationally efficient function that calculates the smooth upper bound on local sensitivity for triangle counting. Finally, we add noise proportional to the smooth upper bound on local sensitivity into the response of the triangle counting query. For small signed graphs, the approach based on smooth sensitivity provides superior utility than that based on the smooth upper bound on local sensitivity. In contrast, for large signed graphs, the method based on the smooth upper bound not only ensures computational efficiency, but also achieves comparable utility to the smooth-sensitivity-based approach.

A. The Calculation of Local Sensitivity

This section elucidates the approaches employed to compute the local sensitivity and the local sensitivity at distance t of the function $f_\triangle$. This function takes a signed graph G as input and outputs the number of balanced and unbalanced triangles.

Local Sensitivity. The local sensitivity of the triangle counting query function $f_\triangle$, denoted as $LS_\triangle(G)$, is defined as follows:

$$LS_\triangle(G) = \max_{\tilde{G}\in\mathcal{G}:d(G,\tilde{G})\leq 1} |\tilde{T}^b - T^b| + |\tilde{T}^u - T^u|. \quad (1)$$

To efficiently compute the local sensitivity, we introduce a specialized definition of local sensitivity, denoted as $LS_{ij}(G)$. This metric evaluates the maximum impact on sensitivity when modifying the relationship between node v_i and node v_j. Such modifications require the addition of a signed edge (v_i, v_j) in its absence, or the alteration of its sign or deletion of a signed edge if it exists. Let w_{ij}^+ denote the number of positive wedges (two-hop paths) between nodes v_i and v_j where $a_{ik}\cdot a_{jk} = 1$. In contrast, w_{ij}^- is the number of negative wedges where this product is -1. The following lemma details the calculation of the local sensitivity of $f_\triangle$. Due to space constraints, all proofs of the subsequent lemmas and theorems can be found in [26].

Lemma 1. *The local sensitivity of $f_\triangle$ is given by $LS_\triangle(G) = \max_{1\leq i<j\leq n} LS_{ij}(G)$, where*

$$LS_{ij}(G) = \begin{cases} w_{ij}^+ + w_{ij}^-, & \text{if } a_{ij} = 0, \\ \max(w_{ij}^+ + w_{ij}^-, 2|w_{ij}^+ - w_{ij}^-|), & \text{if } a_{ij} \neq 0. \end{cases} \quad (2)$$

Algorithm 1 describes the process of computing the number of wedges between all pairs of nodes. It iterates over each node $v_i \in \mathcal{V}$ and then enumerates all edge pairs connected to v_i to determine the number of positive and negative wedges between specific node pairs. Since the signed graphs are undirected, it follows that $w_{ij}^+ = w_{ji}^+$ and $w_{ij}^- = w_{ji}^-$. Therefore, it is sufficient to compute the number of wedges for the node pairs v_i and v_j whose node indexes satisfy $i < j$. Let Γ represent the number of node pairs where $w_{ij}^+ \neq 0$ or $w_{ij}^- \neq 0$. The time complexity of Algorithm 1 is $\mathcal{O}(m\cdot d_{max})$. The calculation of $LS_\triangle(G)$ can be accomplished in time $\mathcal{O}(\Gamma)$.

Algorithm 1: Edge-Scan for Wedge Counting

Input: signed graph G represented as $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$
Output: $\{(w_{ij}^+, w_{ij}^-)\}_{1\leq i<j\leq n}$

1 Initialize $w_{ij}^+ \leftarrow 0$, $w_{ij}^- \leftarrow 0$ for each node pair (v_i, v_j);
2 **for** *each node* $v_i \in \mathcal{V}$ **do**
3 **for** *each pair of edges* (v_i, v_j), (v_i, v_k) *incident to* v_i **do**
  /* suppose $j < k$ */
4 **if** $a_{ij}\cdot a_{ik} = 1$ **then** $w_{jk}^+ \leftarrow w_{jk}^+ + 1$;
5 **else** $w_{jk}^- \leftarrow w_{jk}^- + 1$;
6 **return** $\{(w_{ij}^+, w_{ij}^-)\}_{1\leq i<j\leq n}$

Example 1. *The local sensitivity of the $f_\triangle$ query for the signed graph in Figure 1(a) is 4. This value is achieved by flipping the sign of edge (v_1, v_3) to the graph, i.e., $LS_\triangle(G) = LS_{13}(G) = 4$, where $w_{13}^+ = 2$ and $w_{13}^- = 0$.*

Local Sensitivity at Distance. In the rest of this section, we explain the computation of the local sensitivity at distance t of $f_\triangle$, denoted as $LS_\triangle(G, t)$. For the special scenario where $t = 0$, $LS_\triangle(G, t)$ represents the local sensitivity of $f_\triangle$ at G. When $s \geq 1$, in order to compute $LS_\triangle(G, t)$, we define $LS_{ij}(G, t)$ as the maximum value of $LS_{ij}(\cdot)$ achieved on signed graphs at a distance of at most t from G, expressed as

$$LS_{ij}(G, t) = \max_{\tilde{G}\in\mathcal{G}:d(G,\tilde{G}\leq t)} LS_{ij}(\tilde{G}). \quad (3)$$

Therefore, $LS_\triangle(G, t) = \max_{1\leq i<j\leq n} LS_{ij}(G, t)$. As shown in Lemma 1, the calculation of local sensitivity for $\tilde{G}$ over the node pair (v_i, v_j) depends on whether $\tilde{a}_{ij}$ is equal to 0 or not. Nonetheless, we can first introduce a signed edge (v_i, v_j) if it is absent. Then, irrespective of the existence of edge (v_i, v_j), we must explore modification strategies that maximize $\tilde{w}_{ij}^+ + \tilde{w}_{ij}^-$ and $2|\tilde{w}_{ij}^+ - \tilde{w}_{ij}^-|$ in $\tilde{G}$, respectively. For clarity, we introduce the notation: w_{ij}^s denotes $w_{ij}^+ + w_{ij}^-$, w_{ij}^d represents $|w_{ij}^+ - w_{ij}^-|$, and w_{ij}^m indicates $\min(w_{ij}^+, w_{ij}^-)$. Moreover, we define $W_{ij}^s(G, t)$ as the maximum of $\tilde{w}_{ij}^s$, and $W_{ij}^d(G, t)$ as the maximum of $2\tilde{w}_{ij}^d$.

To comprehend $W_{ij}^s(G, t)$ for positive t, we modify t edges in G to construct a signed graph $\tilde{G}$ that maximizes $\tilde{w}_{ij}^s$. Among all possible modifications, only the addition of edges adjacent to nodes v_i or v_j can increase w_{ij}^s, where the sign of the added edge is irrelevant. Hence, $\tilde{G}$ is obtained by adding t edges to G. The specific allocation strategy is outlined in Lines 5-6 of Algorithm 2. Let b_{ij} denote the number of nodes connected to exactly one of v_i and v_j. For such nodes v_k, adding one edge, either (v_i, v_k) or (v_j, v_k), can increase w_{ij}^s by 1. For nodes v_k connected to neither, it is required to add two edges to achieve the same increment. Therefore, if $t \leq b_{ij}$, the increase in w_{ij}^s is at most t, as specified in Line 5. However, a nuanced allocation

Algorithm 2: LOCAL SENSITIVITY AT DISTANCE

Input: signed graph G represented as $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, distance $t \geq 1$, $\{(w_{ij}^+, w_{ij}^-)\}_{1 \leq i < j \leq n}$
Output: local sensitivity at distance t: $LS_\triangle(G,t)$

1 Initialize modification volume $c \leftarrow t$, $LS_\triangle(G,t) \leftarrow 0$;
2 **for** *each node pair* (v_i, v_j), *where* $1 \leq i < j \leq n$ **do**
// $w_{ij}^s := w_{ij}^+ + w_{ij}^-, w_{ij}^d := |w_{ij}^+ - w_{ij}^-|, w_{ij}^m := \min(w_{ij}^+, w_{ij}^-)$
3 $t \leftarrow c$;
4 $b_{ij} \leftarrow d_i + d_j - 2\left(w_{ij}^s + |a_{ij}|\right)$;
5 **if** $b_{ij} \geq t$ **then** $W_{ij}^s(G,t) \leftarrow w_{ij}^s + t$;
6 **else** $W_{ij}^s(G,t) \leftarrow w_{ij}^s + \lfloor (t + b_{ij})/2 \rfloor$;
7 **if** $a_{ij} = 0$ **then** $t \leftarrow t - 1$;
8 **if** $w_{ij}^m \geq t$ **then** $W_{ij}^d(G,t) \leftarrow 2\left(w_{ij}^d + 2t\right)$;
9 **else**
10 $W_{ij}^d(G,t) \leftarrow 2\left(w_{ij}^d + 2w_{ij}^m\right)$;
11 $t \leftarrow t - w_{ij}^m$;
12 **if** $b_{ij} \geq t$ **then** $W_{ij}^d(G,t) \leftarrow W_{ij}^d(G,t) + 2t$;
13 **else** $W_{ij}^d(G,t) \leftarrow W_{ij}^d(G,t) + 2\lfloor (t + b_{ij})/2 \rfloor$;
14 $LS_{ij}(G,t) \leftarrow \max\left(W_{ij}^s(G,t), W_{ij}^d(G,t)\right)$;
15 $LS_\triangle(G,t) \leftarrow \max\left(LS_\triangle(G,t), LS_{ij}(G,t)\right)$;
16 **return** $LS_\triangle(G,t)$

approach is necessary when $t > b_{ij}$.

First, we add b_{ij} edges to increase w_{ij}^s by b_{ij}. For the rest of $t - b_{ij}$ modifications, we can find $\lfloor (t - b_{ij})/2 \rfloor$ nodes v_k where $a_{ik} = a_{jk} = 0$ and add edges (v_i, v_k) and (v_j, v_k) to increase w_{ij}^s by $\lfloor (t - b_{ij})/2 \rfloor$. This results in a total increase of w_{ij}^s by $\lfloor (t + b_{ij})/2 \rfloor$, as shown in Line 6. Note that the upper bound of w_{ij}^s is $n - 2$. Next, we present the calculation of $W_{ij}^d(G,t)$. Recall that $W_{ij}^d(G,t)$ is computed only if $a_{ij} \neq 0$. However, if $a_{ij} = 0$, we can first introduce an edge (v_i, v_j). As a result, the total number of required modifications is reduced to $t - 1$.

Different from $W_{ij}^s(G,t)$, $W_{ij}^d(G,t)$ represents the maximum of $2\tilde{w}_{ij}^d$ across all signed graphs $\tilde{G}$ where $d(G, \tilde{G}) \leq t$. Thus, our modifications to G should aim to maximize $2w_{ij}^d$. The only effective modification is to operate on edges adjacent to nodes v_i or v_j. To be specific, the addition or deletion of an edge can result in a maximum increase of 2, whereas the alteration of the sign of an edge yields an increase of up to 4. In this paper, we adopt a greedy strategy to calculate $W_{ij}^d(G,t)$, as stated in Lines 8-13. The approach prioritizes edge sign alterations until no further modifications yield increments anymore.

For distinct nodes v_i and v_j, two scenarios must be considered due to the absolute value: either $w_{ij}^+ \geq w_{ij}^-$ or $w_{ij}^+ < w_{ij}^-$. The analysis for both cases is consistent. For simplicity, we assume $w_{ij}^+ \geq w_{ij}^-$, which reduces the objective function to $2(w_{ij}^+ - w_{ij}^-)$. If $t \leq w_{ij}^-$, it is possible to increase the objective by $4t$ via flipping the sign of t edges, as stated in Line 8, where each edge belongs to a negative wedge. For the situation where $t > w_{ij}^-$, we can first reverse the sign of w_{ij}^- edges to increase $2(w_{ij}^+ - w_{ij}^-)$ by $4w_{ij}^-$. Then, the only feasible modification to further increase $2(w_{ij}^+ - w_{ij}^-)$ is to add edges connected to either nodes v_i or v_j. Therefore, for the rest of $t - w_{ij}^-$ modifications, the procedure is similar to that utilized in the calculation of $W_{ij}^s(G,t)$, where the difference is that the increment is double. Different from w_{ij}^s, the upper bound of $2(w_{ij}^+ - w_{ij}^-)$ is $2n - 4$.

Once $W_{ij}^s(G,t)$ and $W_{ij}^d(G,t)$ are computed, the local sensitivity at distance t for the node pair (v_i, v_j) can be determined. This procedure is applied iteratively to all pairs of nodes to compute $LS_\triangle(G,t)$, with a time complexity of $\mathcal{O}(n^2)$.

Example 2. *Figure 1(b) demonstrates the local sensitivity at distance $t = 1$ for the signed graph in Figure 1(a). Specifically, we first introduce the dashed negative edge (v_1, v_4) to obtain $\tilde{G}$, where $\tilde{w}_{13}^+ = 3$ and $\tilde{w}_{13}^- = 0$. Then, the local sensitivity of $\tilde{G}$ is 6, which means that $LS_\triangle(G,1) = LS_\triangle(\tilde{G}) = 6$. Furthermore, we observe that $LS_\triangle(G,t)$ does not exceed 6 for any $t \geq 1$.*

B. Smooth Upper Bound on Local Sensitivity

The previous subsection details the calculation of the local sensitivity and the local sensitivity at distance t of the triangle count query function. Next, we present methods that are used to compute the smooth upper bound on local sensitivity.

Smooth Sensitivity. Smooth sensitivity is the smallest upper bound that satisfies the criterion for a smooth upper bound on local sensitivity. Thus, the mechanism that utilizes smooth sensitivity to calibrate noise can add less noise. For $\beta > 0$, the smooth sensitivity, denoted as $S_{\triangle,\beta}^*(G)$, is expressed as

$$S_{\triangle,\beta}^*(G) = \max_{t \in [0, 2n-3]} e^{-\beta t} LS_\triangle(G,t). \tag{4}$$

Given that $LS_\triangle(G,t)$ remains constant for all $t \geq 2n - 3$, it is sufficient to consider $t \leq 2n - 3$. To compute the smooth sensitivity, one can calculate $e^{-\beta t} LS_\triangle(G,t)$ for each relevant t and select the maximal value as the smooth sensitivity of $f_\triangle$. However, the time complexity of $LS_\triangle(G,t)$, denoted by $\mathcal{O}(n^2)$, coupled with the need to evaluate each possible value of t, raises the overall time complexity to $\mathcal{O}(n^3)$. This complexity makes the method impractical for most applications.

In the calculation of smooth sensitivity, the primary source of time complexity is the computation of local sensitivity at distance, which involves extensive redundant and unproductive calculations. To address this problem, we propose an efficient method to compute the smooth sensitivity of $f_\triangle$. This method eliminates duplicate and irrelevant data based on our established criteria and retains only those elements that are critical to the smooth sensitivity computation. Then, we can determine the local sensitivity at every distance t sequentially, thereby substantially reducing computational overhead.

First of all, we prepare the requisite data for the computation of smooth sensitivity, as described in Lines 3-8 of Algorithm 3. For each node pair where $i < j$, we compute w_{ij}^+, w_{ij}^- and b_{ij}. The values w_{ij}^+ and w_{ij}^- are computed by Algorithm 1, and b_{ij} is determined by $b_{ij} = d_i + d_j - 2(w_{ij}^s + |a_{ij}|)$. For node pairs where $w_{ij}^s \neq 0$, we construct a list $\mathcal{Q}$ that is composed of the triples $(w_{ij}^+, w_{ij}^-, b_{ij})$. For node pairs where $w_{ij}^s = 0$, we select the maximum of b_{ij} and add $(0, 0, b_{ij})$ to the list $\mathcal{Q}$.

To compute local sensitivity at distance efficiently, we propose a set of rules to eliminate duplicate and irrelevant triplets in $\mathcal{Q}$. Recall that the computation of the local sensitivity at distance t primarily considers two scenarios: the maximum of $\tilde{w}_{ij}^+ + \tilde{w}_{ij}^-$ and the maximum of $2|\tilde{w}_{ij}^+ - \tilde{w}_{ij}^-|$. Let $W^s(G,t)$ be the maximum of $\tilde{w}_{ij}^+ + \tilde{w}_{ij}^-$ across all $\tilde{G}$ which satisfy $d(G, \tilde{G}) \leq t$, and we define $W^d(G,t)$ as the maximal value of $2|\tilde{w}_{ij}^+ - \tilde{w}_{ij}^-|$ within the same subset of $\tilde{G}$. Thus, it follows that $W^s(G,t) =$

Algorithm 3: SMOOTH SENSITIVITY CALCULATION

Input: signed graph G denoted as $\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$, parameter β
Output: smooth sensitivity $S^*_{\triangle,\beta}(G)$

Initialize $S^*_{\triangle,\beta} \leftarrow 0$, $b_{max} \leftarrow 0$, $\mathcal{Q} \leftarrow \emptyset$;
$\mathcal{L}_c \leftarrow \emptyset$, $t_0^c \leftarrow 0$ for $c \in [1,4]$;
Construct $\{(w_{ij}^+, w_{ij}^-)\}_{1\le i<j\le n}$ via Algorithm 1;
for *each node pair* (v_i, v_j), *where* $1 \le i < j \le n$ **do**
  // $w_{ij}^s := w_{ij}^+ + w_{ij}^-, w_{ij}^d := |w_{ij}^+ - w_{ij}^-|, w_{ij}^m := \min(w_{ij}^+, w_{ij}^-)$
  $b_{ij} \leftarrow d_i + d_j - 2\left(w_{ij}^s + |a_{ij}|\right)$;
  if $w_{ij}^s \neq 0$ **then** $\mathcal{Q}$.push $\left((w_{ij}^+, w_{ij}^-, b_{ij})\right)$;
  else $b_{max} \leftarrow \max(b_{max}, b_{ij})$;
$\mathcal{Q}$.push $((0,0,b_{max}))$;
$\mathcal{L}_1 \leftarrow$ `CalcList` $\left(\mathcal{Q},\ w_{ij}^s,\ b_{ij},\ \text{Eq.7}\right)$;
$\mathcal{L}_2 \leftarrow$ `CalcList` $\left(\mathcal{Q},\ w_{ij}^d,\ w_{ij}^m,\ \text{Eq.9}\right)$;
$\mathcal{L}_3 \leftarrow$ `CalcList` $\left(\mathcal{Q},\ w_{ij}^d,\ 3w_{ij}^m + b_{ij},\ \text{Eq.11}\right)$;
$\mathcal{L}_4 \leftarrow$ `CalcList` $\left(\mathcal{Q},\ w_{ij}^d + w_{ij}^m,\ w_{ij}^m + b_{ij}, \text{Eq.13}\right)$;
Calculate t_i^c for $c \in [1,4]$ by (8), (10), (12), and (14);
for $t = 0$ **to** $2n-3$ **do**
  Select (w_i^+, w_i^-, b_i) from $\mathcal{L}_1$ where $t \in \left[t_{i-1}^1, t_i^1\right)$;
  $W^s(G,t) \leftarrow w_i^s + \lfloor (t + \min(b_i, t))/2 \rfloor$;
  Select $(w_{i,c}^+, w_{i,c}^-, b_{i,c})$ from other $\mathcal{L}_c$ where $t \in [t_{i-1}^c, t_i^c)$;
  $W^d(G,t) \leftarrow$
    $\max_c 2(w_{i,c}^d + \min(w_{i,c}^m, t) + \lfloor (t + \min(t, w_{i,c}^m + b_{i,c}))/2 \rfloor)$;
  $LS_\triangle(G,t) \leftarrow \max(W^s(G,t), W^d(G,t))$;
  $S^*_{\triangle,\beta}(G) \leftarrow \max(S^*_{\triangle,\beta}(G), e^{-\beta t} LS_\triangle(G,t))$;
return $S^*_{\triangle,\beta}(G)$
Function `CalcList` $(\mathcal{Q}, \text{sortKey}, \text{tieKey}, \text{conditionFn})$
  $\mathcal{Q}_c \leftarrow$ sort $\mathcal{Q}$ by sortKey, resolve ties with tieKey;
  for $i = 1$ **to** $|\mathcal{Q}_c|$ **do**
    if conditionFn $(\mathcal{Q}_c[i], \mathcal{Q}_c[i-1])$ **then**
      $\mathcal{L}_c$.push $(\mathcal{Q}_c[i])$
  return $\mathcal{L}_c$

$\max_{1\le i<j\le n} W_{ij}^s(G,t)$, $W^d(G,t) = \max_{1\le i<j\le n} W_{ij}^d(G,t)$, and $LS_\triangle(G,t) = \max(W^s(G,t), W^d(G,t))$. Note that we do not consider whether $a_{ij} = 0$ or not when computing $LS_\triangle(G,t)$. In the cases where $a_{ij} = 0$, the required number of modifications decreases from t to $t-1$. However, our algorithm still executes t modifications to G, which does not compromise privacy, as $LS_\triangle(G,t)$ is a monotonically non-decreasing function of t. In accordance with Algorithm 2, $W_{ij}^s(G,t)$ can be formulated as

$$W_{ij}^s(G,t) = \begin{cases} w_{ij}^s + t, \text{if } t \le b_{ij}, \\ w_{ij}^s + \lfloor (t + b_{ij})/2 \rfloor, \text{if } t > b_{ij}, \end{cases} \tag{5}$$

and $W_{ij}^d(G,t)$ can be expressed as

$$W_{ij}^d(G,t) = \begin{cases} 2w_{ij}^d + 4t, \text{if } t \le w_{ij}^m, \\ 2w_{ij}^d + 2w_{ij}^m + 2t, \text{if } w_{ij}^m < t \le w_{ij}^m + b_{ij}, \\ 2(w_{ij}^d + w_{ij}^m + \lfloor (t + w_{ij}^m + b_{ij})/2 \rfloor), \text{otherwise.} \end{cases} \tag{6}$$

To compute $W^s(G,t)$, we first sort the $\Gamma+1$ pairs (w_{ij}^s, b_{ij}) in a non-increasing order by w_{ij}^s, represented as $\mathcal{Q}_1$. If multiple pairs share the same w_{ij}^s, only the pair with the maximum of b_{ij} is retained. Then, we process $\mathcal{Q}_1$ in the order of descending w_{ij}^s. The pair (w_i^s, b_i) is maintained only if it, relative to the previously retained pair (w_{i-1}^s, b_{i-1}), satisfies the condition:

$$2(w_{i-1}^s - w_i^s) + b_{i-1} < b_i. \tag{7}$$

The set of pairs that meet this condition is represented as $\mathcal{L}_1 = \{(w_1^s, b_1), \ldots, (w_{k_1}^s, b_{k_1})\}$. Equally pivotal in our analysis is the determination of breakpoints, utilized to select the appropriate data from the filtered list to compute $W^s(G,t)$ or $W^d(G,t)$. In what follows, we set the initial breakpoint as 0 and the final breakpoint as $2n-3$. In the computation of $W^s(G,t)$, the other breakpoints are set to

$$t_i^1 = 2(w_i^s - w_{i+1}^s) + b_i, i \in [k_1 - 1]. \tag{8}$$

Then, $W^s(G,t)$ can be computed by the formulas $W^s(G,t) = w_i^s + \lfloor (t + \min(b_i, t))/2 \rfloor$ if $t \in [t_{i-1}^1, t_i^1)$.

The calculation of $W^d(G,t)$ is more complicated, primarily because the independent variable t must be compared with two different quantities: w_{ij}^m and $w_{ij}^m + b_{ij}$. To address this problem, we maintain three different lists. First of all, we sort the $\Gamma+1$ triplets $(w_{ij}^d, w_{ij}^m, b_{ij})$ in a non-increasing order based on two criteria: w_{ij}^d and $w_{ij}^d + w_{ij}^m$. For the first criterion, if multiple triplets exhibit identical w_{ij}^d values, we retain only the triplet with the largest w_{ij}^m. This list is represented as $\mathcal{Q}_2$. In addition, we also utilize the first criterion to construct $\mathcal{Q}_3$ but retain the triplet with the largest $3w_{ij}^m + b_{ij}$ in case of same w_{ij}^d values. Similarly, for the second criterion, if multiple triplets have the same value of $w_{ij}^d + w_{ij}^m$, we keep the triplet with the largest $w_{ij}^m + b_{ij}$, and this list is denoted as $\mathcal{Q}_4$.

Then, we respectively process the sorted lists $\mathcal{Q}_2$, $\mathcal{Q}_3$, and $\mathcal{Q}_4$. The first treatment involves a traversal of the list $\mathcal{Q}_2$ in the order of decreasing w_{ij}^d. Each triplet (w_i^d, w_i^m, b_i) is evaluated for retention based on the following condition:

$$w_{i-1}^d - w_i^d + w_{i-1}^m < w_i^m, \tag{9}$$

where $(w_{i-1}^d, w_{i-1}^m, b_{i-1})$ is the previous retained triplet. The collection of triplets that satisfy this criterion is represented as $\mathcal{L}_2 = \{(w_1^d, w_1^m, b_1), \ldots, (w_{k_2}^d, w_{k_2}^m, b_{k_2})\}$. The breakpoints are calculated via the formula:

$$t_i^2 = w_i^d - w_{i+1}^d + w_i^m, i \in [k_2 - 1]. \tag{10}$$

Go through the list $\mathcal{Q}_3$ in the order of decreasing w_{ij}^d. Each triplet (w_i^d, w_i^m, b_i) is retained only if it satisfies the condition

$$2(w_{i-1}^d - w_i^d) + 3w_{i-1}^m + b_{i-1} < 3w_i^m. \tag{11}$$

These triplets form $\mathcal{L}_3 = \{(w_1^d, w_1^m, b_1), \ldots, (w_{k_3}^d, w_{k_3}^m, b_{k_3})\}$. To determine the breakpoints for $\mathcal{L}_3$, we set each breakpoint as:

$$t_i^3 = (2(w_i^d - w_{i+1}^d) + 3w_i^m + b_i)/3, i \in [k_3 - 1]. \tag{12}$$

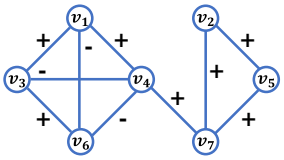
For the sorted triplet list $\mathcal{Q}_4$, we first traverse this list in the order of decreasing $w_{ij}^d + w_{ij}^m$. For each triplet, keep it only if it satisfies the following criterion:

$$2(w_{i-1}^d - w_i^d) + 3w_{i-1}^m - 3w_i^m + b_{i-1} < b_i \tag{13}$$

Sequentially, the set of triplets that fulfill this condition constitutes $\mathcal{L}_4 = \{(w_1^d, w_1^m, b_1), \ldots, (w_{k_4}^d, w_{k_4}^m, b_{k_4})\}$. The breakpoints for $\mathcal{L}_4$ are determined by the formula:

$$t_i^4 = 2(w_i^d - w_{i+1}^d) + 3w_i^m - 2w_{i+1}^m + b_i, i \in [k_4 - 1]. \tag{14}$$

To complete the calculation of $W^d(G,t)$, it is imperative to choose the relevant triplets from the three filtered lists. Given a specific distance t, similar to the computation of $W^s(G,t)$, we choose three triplets from $\mathcal{L}_2$, $\mathcal{L}_3$, and $\mathcal{L}_4$, respectively, based on the comparison of t with their respective breakpoint sequences, represented as $(w_{i,c}^d, w_{i,c}^m, b_{i,c})$, where $c \in \{2,3,4\}$. Then, $W^d(G,t)$ can be calculated by the equation: $W^d(G,t) = \max_c 2w_{i,c}^d + 2\min(w_{i,c}^m, t) + 2\lfloor (t + \min(t, w_{i,c}^m + b_{i,c}))/2 \rfloor$.



$\mathcal{Q}$ = {(0,2,0), (1,1,1), (1,1,0), (1,0,4), (1,0,4), (1,0,0), (1,0,1), (1,1,1), (1,1,0), (0,1,4), (1,0,4), (0,2,1), (1,0,1), (0,1,4)(0,0,5)}

$\mathcal{Q}_1$ = {(1,1,1), (1,0,4), (0,0,5)}, $\mathcal{L}_1$ = {(1,1,1), (1,0,4)}

$\mathcal{Q}_2$ = {(0,2,0), (1,0,4), (1,1,1)}, $\mathcal{L}_2$ = {(0,2,0)}

$\mathcal{Q}_3$ = {(0,2,1), (1,0,4), (0,0,5)}, $\mathcal{L}_3$ = {(0,2,1)}

$\mathcal{Q}_4$ = {(0,2,1), (1,0,4), (0,0,5)}, $\mathcal{L}_4$ = {(0,2,1), (1,0,4)}

Fig. 3: Examples of the smooth sensitivity calculation.

In order to compute the smooth sensitivity of the function $f_\triangle$, we sequentially compute $e^{-\beta t} LS_\triangle(G,t)$ for each distance t, where $LS_\triangle(G,t) = \max(W^s(G,t), W^d(G,t))$, and select the maximum as the smooth sensitivity. This calculation is detailed in the pseudo-code presented in Algorithm 3.

Example 3. *Figure 3 illustrates the calculation of the smooth sensitivity for a signed graph G with $n = 7$ nodes. Specifically, we first prepare the data $\mathcal{Q}$ needed to compute SS, and then filter out irrelevant elements to derive the lists $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, and $\mathcal{L}_4$, which enhance the efficiency of the smooth sensitivity calculation. Subsequently, we can iterate through the elements in these lists to sequentially calculate LS at each distance, and finally determine the smooth sensitivity SS of the function $f_\triangle$.*

Furthermore, we have established a theorem that delineates an explicit condition under which the maximal value of $W^s(G)$ and $W^d(G)$ is equal to the smooth sensitivity, where $W^s(G) = \max_{1\le i<j\le n} w^s_{ij}$, and $W^d(G) = \max_{1\le i<j\le n} 2w^d_{ij}$. Under this condition, our randomized algorithm $\mathcal{A}$ introduces noise that is proportional to $\max(W^s(G), W^d(G))$, without the necessity to calculate the local sensitivity at distance t.

Theorem 2. *Given a signed graph G, if $W^s(G) \ge \frac{1}{\beta}$ and $W^d(G) \ge \frac{4}{\beta}$, then $S^*_{\triangle,\beta}(G) = \max(W^s(G), W^d(G))$.*

The time complexity of smooth sensitivity computation is analyzed methodically over several phases. In the data preparation phases, the time complexity is described as $\mathcal{O}(m \cdot d_{max} + n^2)$. This complexity comes from the $\mathcal{O}(m \cdot d_{max})$ time required to compute w^+_{ij} and w^-_{ij}, and the time required to calculate b_{ij}, quantified as $\mathcal{O}(n^2)$. In the phase dedicated to the elimination of duplicate and irrelevant data, the time complexity remains at $\mathcal{O}(\Gamma)$, as both the creation of sorted lists by Bucket Sort and the construction of filtered lists are achievable within $\mathcal{O}(\Gamma)$. The computation of smooth sensitivity in the final phase requires $\mathcal{O}(n)$ time. Hence, the overall time complexity of our proposed method is established as $\mathcal{O}(m \cdot d_{max} + n^2)$.

Smooth Upper Bound. For large signed graphs, computing the smooth sensitivity of the triangle count query is non-trivial due to the excessive time complexity. To address this problem, we relax the requirements for smooth sensitivity computation and instead construct a computationally efficient function that provides a smooth upper bound on local sensitivity. Given an arbitrary database x and a query function f, LS_f has multiple smooth bounds, and the proposition below provides a approach to determine the smooth upper bounds on local sensitivity.

Proposition 1. *[11] Define $S_{f,\beta}(x) = \max_{t\in[0,n]} e^{-\beta t} U(x,t)$, where $U(x,t)$ satisfies $LS_f(x) \le U(x,0)$ for all x and $U(x,t) \le U(\tilde{x}, t+1)$ for all $x, \tilde{x}$ such that $d(x,\tilde{x}) = 1$. Then $S_{f,\beta}(x)$ is a β-smooth upper bound on local sensitivity.*

Algorithm 4: SMOOTH UPPER BOUND ON LS

Input: signed graph G denoted as $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, parameter β
Output: smooth upper bound on local sensitivity $S_{\triangle,\beta}(G)$

1 $W^s(G) \leftarrow 0$; $W^d(G) \leftarrow 0$;
2 Construct $\{(w^+_{ij}, w^-_{ij})\}_{1\le i<j\le n}$ via Algorithm 1;
3 **for** *each pair (w^+_{ij}, w^-_{ij}), where $w^+_{ij} \neq 0$ or $w^-_{ij} \neq 0$* **do**
4 $W^s(G) \leftarrow \max(W^s(G), w^s_{ij})$;
5 $W^d(G) \leftarrow \max(W^d(G), 2w^d_{ij})$;
6 $t_1 \leftarrow \frac{1}{\beta} - W^s(G)$; $t_2 \leftarrow \frac{1}{\beta} - W^d(G)/4$;
7 **if** $t_1 \le 0$ **then** $S_{\triangle,\beta}(G) \leftarrow W^s(G)$;
8 **else**
9 $S_{\triangle,\beta}(G) \leftarrow e^{-\beta\lceil t_1\rceil}(W^s(G) + \lceil t_1\rceil)$;
10 $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), e^{-\beta\lfloor t_1\rfloor}(W^s(G) + \lfloor t_1\rfloor))$;
11 **if** $t_2 \le 0$ **then** $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), W^d(G))$;
12 **else**
13 $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), e^{-\beta\lceil t_2\rceil}(W^d(G) + 4\lceil t_2\rceil))$;
14 $S_{\triangle,\beta}(G) \leftarrow \max(S_{\triangle,\beta}(G), e^{-\beta\lfloor t_2\rfloor}(W^d(G) + 4\lfloor t_2\rfloor))$;
15 **return** $S_{\triangle,\beta}(G)$

Inspired by Theorem 2, we develop a function that not only meets the criteria for a smooth upper bound on $LS_\triangle$, but also enhances computational efficiency.

Theorem 3. *Define $U(G,t) = \max(U^s(G,t), U^d(G,t))$, where $U^s(G,t) = W^s(G) + t$ and $U^d(G,t) = W^d(G) + 4t$. Then the function $S_{\triangle,\beta}(G) = \max_{t\in[0,2n-3]} e^{-\beta t} U(G,t)$ is a β-smooth upper bound on local sensitivity of $f_\triangle$.*

Actually, it is feasible to compute $S_{\triangle,\beta}(G)$ efficiently. The function $S_{\triangle,\beta}(G)$ can also be expressed as

$$S_{\triangle,\beta}(G) = \max_{t\in[0,2n-3]} \{e^{-\beta t}(W^s(G)+t), e^{-\beta t}(W^d(G)+4t)\}. \quad (15)$$

It can be straightforwardly deduced that the peak values of the functions $h(t) = e^{-\beta t}(W^s(G)+t)$ and $g(t) = e^{-\beta t}(W^d(G)+4t)$ occur at $t = \frac{1}{\beta} - W^s(G)$ and $t = \frac{1}{\beta} - W^d(G)/4$, respectively. Given the discrete nature and bounded domain of t, the smooth sensitivity $S_{\triangle,\beta}(G)$ can be determined in constant time based on the precomputed values of $W^s(G)$ and $W^d(G)$, as described in Lines 6-14 of Algorithm 4. And $W^s(G)$ and $W^d(G)$ can be computed in time $\mathcal{O}(m \cdot d_{max})$. The process of the computation of the smooth upper bound on LS is presented in Algorithm 4, and the worst-case time complexity is $\mathcal{O}(m \cdot d_{max})$.

To compute the smooth upper bound on LS, it is necessary to determine the maximum values of w^s_{ij} and w^d_{ij} for all $i < j$. The method mentioned earlier is to first compute w^+_{ij} and w^-_{ij} for all node pairs, and then traverse these counts to determine the maxima. However, it is infeasible to store all the statistics in memory for extremely large graphs. An alternative approach is to enumerate all node pairs and then compute these counts on-the-fly from the intersection of two adjacent vectors. While this reduces memory consumption, it incurs a time complexity of $\mathcal{O}(mn)$. To solve this issue, we propose an efficient degree-based pruning technique to reduce the number of node pairs to be processed. The intuition behind this technique stems from the fact that nodes with smaller degrees are unlikely to contribute much to w^s_{ij} and w^d_{ij}, since the relationship between the degrees and these values satisfies $w^d_{ij} \le w^s_{ij} \le \min(d_i, d_j)$.

Algorithm 5: COMPUTE MAX WEDGE METRICS

Input: signed graph G represented as $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$
Output: maximum wedge metrics $W^s(G)$ and $W^d(G)$

$W^s(G) \leftarrow 0$, $W^d(G) \leftarrow 0$;
$S \leftarrow \emptyset$;
sortedNodes $\leftarrow$ sort the nodes in descending order by d_i;
for *each node* $v_i \in$ sortedNodes **do**
  if $d_i \leq \min(W^s(G), W^d(G)/2)$ **then**
    break;
  for *each node* $v_j \in S$ **do**
    $w_{ij}^+, w_{ij}^- \leftarrow$ INTERSECTIONCOUNTING$(\boldsymbol{a}_i, \boldsymbol{a}_j)$;
    $W^s(G) \leftarrow \max(W^s(G), w_{ij}^s)$;
    $W^d(G) \leftarrow \max(W^d(G), 2w_{ij}^d)$;
  $S \leftarrow S \cup \{v_i\}$;
return $W^s(G), W^d(G)$

Algorithm 5 describes the computation of $W^s(G)$ and $W^d(G)$. To be specific, we first sort the nodes in descending order by their degrees, and traverse them in this order. Once the degree of node v_i is smaller than $\max(W^s(G), W^d(G)/2)$, the traversal terminates, which enhances the computational efficiency.

The empirical results of the experiments demonstrate that our proposed methods, based on the smooth upper bound on LS, is more efficient than that based on the smooth sensitivity. Moreover, the approach used to compute $W^s(G)$ and $W^d(G)$ in Algorithm 5 outperforms the method described in Algorithm 4 in terms of both runtime and memory consumption.

Privacy Guarantee. The algorithms used to calculate smooth upper bounds on $LS_\triangle$ can be employed in conjunction with Theorem 1 to obtain efficient differentially private algorithms for the estimation of balanced and unbalanced triangle counts.

Theorem 4. *Given $\epsilon > 0$ and $\delta > 0$, let $\beta = \frac{\epsilon}{8+4\ln(2/\delta)}$. The randomized algorithm $\mathcal{A}(G) = f_\triangle(G) + (2S_{\triangle,\beta}(G)/\epsilon) \cdot Z$ satisfies (ϵ, δ)-signed edge centralized differential privacy where Z is sampled from 2-dimensional Laplace distribution.*

IV. THE PROPOSED LOCAL DP METHOD

Local differential privacy has been extensively accepted and adopted in academic research and industry applications due to its robust framework for privacy preservation. Unlike centralized DP, LDP provides enhanced privacy protection since data obfuscation is performed at the node level rather than by a data curator. In this section, we explore the techniques employed to compute the number of balanced and unbalanced triangles in a signed graph under local DP. Inspired by the previous work of Imola et al. [20], we propose a two-phase framework tailored for balanced and unbalanced triangle counting. The first phase uses Generalized Randomized Response mechanism to perturb the adjacency vectors locally. In the second phase, we propose an innovative perturbation mechanism. This mechanism injects noise into the response of each node, where the added noise is calibrated to the smooth upper bound. This two-phase method achieves a better trade-off between data utility and privacy.

The Two-Phase Framework. In the local model, each node v_i independently maintains its own data, denoted as an adjacency vector $\boldsymbol{a}_i$. This vector contains information about the connections of node v_i and the signs of these connections. However, when the query function is $f_\triangle$, nodes cannot locally calculate and submit the obfuscated counts of balanced and unbalanced triangles due to their limited insight into the complete graph structure. Specifically, node v_i is inherently unable to perceive any triangle (v_i, v_j, v_k), because it lacks information about the existence of the edge (v_j, v_k) and its sign in the signed graph.

To address this issue, we propose a two-phase framework to privately response to the query function $f_\triangle$. In the first phase, each node $v_i \in \mathcal{V}$ independently invokes the GRR mechanism to perturb the entities a_{ij} in its adjacency vector (where $i > j$). These elements represent the connections between node v_i and nodes with smaller IDs. Given the allocated privacy budget ϵ_1, the perturbation scheme of the GRR mechanism is as follows:

$$\Pr[\hat{a}_{ij}|a_{ij}] = \begin{cases} \frac{e^{\epsilon_1}}{e^{\epsilon_1}+2}, & \text{if } \hat{a}_{ij} = a_{ij}, \\ \frac{1}{e^{\epsilon_1}+2}, & \text{if } \hat{a}_{ij} \neq a_{ij}. \end{cases} \tag{16}$$

The data curator then collects the distorted data from all nodes and constructs a noisy signed graph $\hat{G}$, under the premise of an undirected structure. In fact, the data curator can compute the number of balanced and unbalanced triangles directly from $\hat{G}$. However, this estimation method infuses substantial noise into the statistical query results since each edge and its associated sign are modified with a certain probability. This indicates that the state of any triangle in $\hat{G}$ is influenced by three independent random variables. As a matter of fact, for any triangle which involves a certain node, only one edge and its associated sign are unknown to that node. If the data collector publishes the noisy signed graph $\hat{G}$, nodes can obtain information about the edges they are unaware of from $\hat{G}$. Therefore, we can introduce an additional round of interaction between nodes and the data collector to minimize noise injection.

In the second phase, each node v_i computes the number of noisy triangles formed by (v_i, v_j, v_k) where the information of the edge (v_j, v_k) can be obtained from $\hat{G}$. Here, a noisy triangle is classified as balanced if the product $a_{ij} \cdot a_{ik} \cdot \hat{a}_{jk}$ equals 1, and as unbalanced if it is -1. This approach ensures that only one edge in each noisy triangle is obfuscated, which enhances the overall utility, albeit at the cost of increased communication overhead per node. However, the direct release of these noisy triangle counts may lead to two major problems. The first issue is the introduction of statistical bias into the estimates of these triangle counts. This bias comes from the noise added during the adjacency vector obfuscation process, which can skew the final estimates. Secondly, and perhaps more importantly, direct publication of such data potentially jeopardizes the privacy of the edges and their respective signs in G. This concern stems from the fact that each node still employs its real data when it calculates balanced and unbalanced triangle counts.

To address the first issue, we employ an empirical estimation technique [20], [21], [27] that yields an unbiased estimate of $f_\triangle(G)$ from these noisy statistics. Let T_i^b and T_i^u be the counts of balanced and unbalanced triangles with noise computed by v_i, where the IDs of the three nodes in each triangle satisfy $i > j > k$. In addition, node v_i is also required to calculate the

number of 2-stars centered on itself under the same constraints of node IDs, denoted as s_i. The data curator then estimates the number of balanced and unbalanced triangles in G, represented as $\bar{T}^b$ and $\bar{T}^u$, based on the data collected from each node. Let $q = 1/(e^{\epsilon_1}+2)$, and the estimates can be formulated as:

$$\bar{T}^b = \frac{1}{1-3q}\sum_{i=1}^{n}(T_i^b - q\cdot s_i), \bar{T}^u = \frac{1}{1-3q}\sum_{i=1}^{n}(T_i^u - q\cdot s_i). \quad (17)$$

Lemma 2. *The values $\bar{T}^b$ and $\bar{T}^u$ as shown in Equation (17), are unbiased estimates of the true counts of balanced and unbalanced triangles in G, i.e., $\mathbb{E}[\bar{T}^b] = T^b$ and $\mathbb{E}[\bar{T}^u] = T^u$.*

To address the privacy concerns, each node needs to perturb its computed statistics via an LDP mechanism and then sends them to the data curator. A simple implementation is to utilize the Laplace mechanism for the data perturbation. Nevertheless, this approach is less practical due to the high global sensitivity, quantified as $2(n-2)$. To address this problem, Imola et al. [20] employ graph projection to reduce sensitivity. This approach removes some neighbors from the adjacency vector so that the maximum degree d_{max} is bounded by a predetermined threshold $\hat{d}_{max}$. As a result, the global sensitivity is reduced to $2(\hat{d}_{max}-1)$. However, there are two main limitations of this approach. The first limitation is that the determination of the threshold $\hat{d}_{max}$ requires an additional privacy budget. In addition, since real-world graphs obey power-law degree distributions [3], [22], the direct addition of Laplace noise calibrated to $2(\hat{d}_{max}-1)$ would substantially diminish the utility of the data. To alleviate these limitations, we propose an innovative approach that employs a smooth upper bound on local sensitivity to calibrate the noise.

Given the presence of empirical estimates, the release statistics for each node v_i are adjusted to $(T_i^b - qs_i, T_i^u - qs_i)$. Thus, we need to devise a function that can efficiently calculate the smooth upper bound on local sensitivity for such statistical estimates. For node v_i, the local sensitivity of these estimates can be expressed as follows:

$$LS_{\triangle}(\boldsymbol{a}_i) = \max_{d(\boldsymbol{a}_i,\tilde{\boldsymbol{a}}_i)\le 1} |\tilde{T}_i^b - T_i^b - q(\tilde{s}_i - s_i)| + |\tilde{T}_i^u - T_i^u - q(\tilde{s}_i - s_i)|.$$

Let d_i' represent the number of nodes that are connected to v_i and have smaller IDs than v_i. A specific smooth upper bound on local sensitivity is described in the subsequent theorem.

Theorem 5. *For any node v_i, let $U(\boldsymbol{a}_i, t) = \max(d_i'+t, 2(d_i'+t-1))$. The function $S_{\triangle,\beta}(\boldsymbol{a}_i) = \max_{t\in[0,i-1-d_i']} e^{-\beta t}U(\boldsymbol{a}_i, t)$ is a β-smooth upper bound on local sensitivity for node v_i.*

For the node v_i, the smooth upper bound on local sensitivity, denoted by $S_{\triangle,\beta}(\boldsymbol{a}_i)$, can be expressed as

$$S_{\triangle,\beta}(\boldsymbol{a}_i) = \max_{t\in[0,i-1-d_i']}\{e^{-\beta t}(d_i'+t), 2e^{-\beta t}(d_i'+t-1)\}. \quad (18)$$

The computation of $S_{\triangle,\beta}(\boldsymbol{a_i})$ is similar to that of the smooth upper bound on local sensitivity under the centralized DP. For any number d_i', the maximum values of the functions $h(t) = e^{-\beta t}(d_i'+t)$ and $g(t) = 2e^{-\beta t}(d_i'+t-1)$ are achieved at $t = \frac{1}{\beta} - d_i'$ and $t = \frac{1}{\beta} - d_i' + 1$, respectively. Given the discrete constraints and a specified interval for t, the smooth upper bound of node v_i, $S_{\triangle,\beta}(\boldsymbol{a_i})$, can be determined in constant time.

Algorithm 6: THE TWO-PHASE FRAMEWORK

Input: signed graph G represented as adjacency vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, the privacy budget ϵ_1, ϵ_2, the invalidation probability δ
Output: balanced and unbalanced triangle counts $\hat{T}^b$, $\hat{T}^u$

Initialize $q \leftarrow \frac{1}{e^{\epsilon_1}+2}$, $\beta \leftarrow \frac{\epsilon_2}{8+4\ln(2/\delta)}$;
for *each node* $v_i \in \mathcal{V}$ **do**
  $\hat{\boldsymbol{a}}_i \leftarrow \{GRR_{\epsilon_1}(a_{ij})\}$ where $i > j$;
  Send the perturbed data $\hat{\boldsymbol{a}}_i$ to the server;
Collect noisy data $\{\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_n\}$ and construct $\hat{G}$;
for *each node* $v_i \in \mathcal{V}$ **do**
  $T_i^b \leftarrow |\{(v_i, v_j, v_k) : i > j > k, a_{ij}\cdot a_{ik}\cdot \hat{a}_{jk} = 1\}|$;
  $T_i^u \leftarrow |\{(v_i, v_j, v_k) : i > j > k, a_{ij}\cdot a_{ik}\cdot \hat{a}_{jk} = -1\}|$;
  $s_i \leftarrow |\{(v_i, v_j, v_k) : i > j > k, a_{ij}\cdot a_{ik} \neq 0\}|$;
  $\hat{T}_i^b \leftarrow (T_i^b - q\cdot s_i) + \mathrm{Lap}(2S_{\triangle,\beta}(\boldsymbol{a}_i)/\epsilon_2)$;
  $\hat{T}_i^u \leftarrow (T_i^u - q\cdot s_i) + \mathrm{Lap}(2S_{\triangle,\beta}(\boldsymbol{a}_i)/\epsilon_2)$;
  Send the perturbed data $\hat{T}_i^b, \hat{T}_i^u$ to the server;
$\hat{T}^b \leftarrow \frac{1}{1-3q}\sum_{i=1}^{n}\hat{T}_i^b$; $\hat{T}^u \leftarrow \frac{1}{1-3q}\sum_{i=1}^{n}\hat{T}_i^u$;
return $\hat{T}^b, \hat{T}^u$

Algorithm 6 describes the overall protocol of our two-phase framework. The algorithm takes as input a signed graph G, the privacy budgets ϵ_1 and ϵ_2 for the first and second phases, and an invalidation probability δ. In the first phase, each node v_i applies $\mathrm{GRR}_{\epsilon_1}$ to the elements a_{ij} in his/her adjacency vector $\boldsymbol{a}_i$, where the node IDs satisfy $j < i$. Then, node v_i submits the obfuscated data $\hat{\boldsymbol{a}}_i$ to the server. Finally, the server constructs a noisy $\hat{G}$ based on the data collected from all nodes. In the second phase, node v_i first computes the statistics T_i^b, T_i^u and s_i under the condition where node IDs satisfy $i > j > k$. Then, it adds the noise $\mathrm{Lap}(2S_{\triangle,\beta}(\boldsymbol{a}_i)/\epsilon_2)$ to $T_i^b - qs_i$ and $T_i^u - qs_i$ and reports the perturbed statistics $\hat{T}_i^b$ and $\hat{T}_i^u$ to the untrusted server. Finally, the server releases the estimates $\hat{T}^b$ and $\hat{T}^u$. In this framework, the primary computational cost is on the client side. For each node v_i, the perturbation takes $\mathcal{O}(n)$ time, while the computation of T_i^b and T_i^u takes $\mathcal{O}(d_i^2)$ time. On the server side, it takes $\mathcal{O}(n)$ time to collect the perturbed statistics.

Theorem 6. *Algorithm 6 satisfies $(\epsilon_1+\epsilon_2, \delta)$-signed edge LDP.*

Remark. In this paper, we assume node identities are public and protect the privacy of edges and their signs, a widely used assumption [11]–[13], [20]. This assumption is reasonable and applicable to real-world applications, such as social networks, where user identities are public, but connections and their signs are private, and financial networks, where bank identities are public, but transactions and their statuses are private. However, in scenarios like healthcare networks, where node identities are also private, a new privacy model is needed, such as node DP, which protects each node and its adjacent edges. It provides a stronger guarantee than edge DP but requires more noise. We leave the exploration of node DP for future work.

V. EXPERIMENTS

A. Experimental Setup

Datasets. Our experiments make use of six real-world datasets. Key statistics are summarized in Table I, with details provided in [26]. Wiki-vote, epinions, and wikisigned are signed graphs.

TABLE I: Dataset Statistics

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	T^b	T^u
wiki-vote (WV)	7,115	100,693	458,597	148,682
epinions (EP)	131,580	711,210	4,368,206	541,870
wikisigned (WS)	138,587	715,883	2,659,365	318,661
youtube (YT)	1,134,890	2,987,624	1,626,212	1,430,174
pokec (PK)	1,632,803	30,622,564	17,321,674	15,235,784
dbpedia (DB)	18,268,991	126,890,209	174,860,389	153,883,022

The datasets youtube, pokec, and dbpedia are used to evaluate the scalability of our proposed approaches. In accordance with the procedures stated in [3], we randomly allocate 70% of the edges as positive and the remainder as negative.

Competitors and Parameter Selection. To the best of our knowledge, we are the first to study the problem of privacy-preserving triangle counting in signed graphs. Since there is no previous research on the subject, we present baseline methods and evaluate our proposed approaches on six datasets.

In the centralized model, our proposed approaches introduce Laplace noise into the query response, and the difference lies in the noise scale. The approach that calibrates the noise based on the smooth sensitivity is represented as CentralSS, while the one that calibrates the noise based on the smooth upper bound on local sensitivity is referred to as CentralSU. Furthermore, we implement a baseline approach, CentralGS, which injects Laplace noise proportional to the global sensitivity into the response. To provide a fair comparison, the baseline approach releases true counts with a probability of δ, and the perturbed counts otherwise. All approaches satisfy (ϵ, δ)-CDP, which has similar semantics to ϵ-CDP when $1/\delta$ is at least the number of possible edges in the signed graph [12]. In our experiments, δ is set as $1/(10\binom{n}{2})$, where n denotes the number of nodes.

In the local model, our proposed two-phase framework is called as LocalTwoSU. To evaluate the performance of our method, we compare it with two other approaches: LocalOne and LocalTwoGS. LocalOne estimates the statistical counts from $\hat{G}$. LocalTwoGS, similar to LocalTwoSU, operates within a two-phase framework. Nevertheless, the distinction arises in the second phase, where LocalTwoGS utilizes graph projection to diminish global sensitivity. Subsequently, each node reports the true counts with a probability of δ and, alternatively, provides the counts distorted by Laplace noise in other instances. Note that LocalOne satisfies ϵ-local DP, while LocalTwoGS and LocalTwoSU satisfy (ϵ, δ)-local DP. For parameter selection, we set $\epsilon_1 = 0.5\epsilon$ and $\epsilon_2 = 0.5\epsilon$ for the first phase and second phase, respectively. For the invalidation probability δ, we set δ to $\frac{1}{10n}$, where n represents the total number of nodes.

Evaluation Metric. We assess the performance of all methods by the *Relative Error* (RE) [18], [20], [21], which is defined as $\frac{|\hat{T}^b - T^b| + |\hat{T}^u - T^u|}{T^b + T^u}$. To ensure statistical reliability, each experiment is replicated 100 times, with the mean outcomes reported. All methods are implemented in C++ and executed on a server with an AMD Ryzen 3995WX CPU and 256GB RAM.

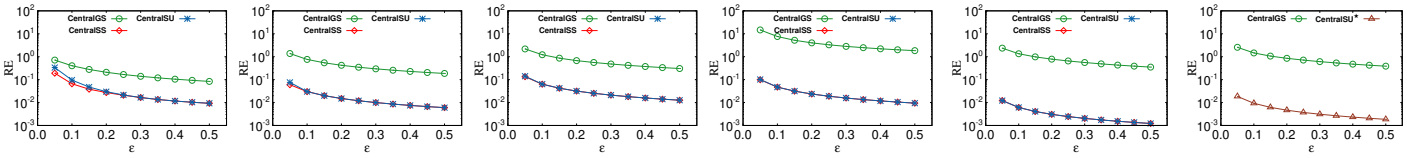
B. Experimental Results

Centralized Model. In the first set of experiments, we evaluate the performance of various methods as the privacy cost ϵ varies between 0.05 and 0.5. The relative errors for each mechanism are presented in Figure 4. Our proposed approaches, CentralSS and CentralSU, achieve superior accuracy across all datasets. For $\epsilon = 0.5$, their relative errors are maintained below or close to 1%. As ϵ decreases, the accuracy declines, but the errors stay below 15% even at $\epsilon = 0.05$, except the smaller dataset wiki-vote. Our methods consistently outperform the benchmark due to lower noise injection and the improvement is remarkable. In addition, CentralSS performs better on smaller datasets and at lower ϵ values, while CentralSU is comparable in other cases. For example, in the wiki-vote dataset, CentralSS yield better accuracy than CentralSU when $\epsilon \leq 0.25$. This is because the smooth sensitivity is optimal over all smooth bounds, whereas the noise added in CentralSU relies on the smooth upper bound on LS, whose stationary point is subject to the joint effect of ϵ, $W^s(G)$, and $W^d(G)$. When these factors take on small values, it could exceed the smooth sensitivity. Note that for extremely large datasets, such as dbpedia, these two methods cannot run on machines with 256GB RAM due to their extreme memory consumption. By contrast, our proposed approach, CentralSU*, employs pruning techniques and can easily handle data of this scale. The only difference between CentralSU and CentralSU* is the computation of $W^s(G)$ and $W^d(G)$, which ensures that CentralSU* also achieves competitive accuracy.

In the second set of experiments, we evaluate the influence of the number of nodes n on the relative error and runtime. We randomly sample 20%-80% nodes from the full dataset to construct four small data. Figure 5 summarizes the performance of CentralSS and CentralSU at $\epsilon = 0.1$. Similarly, for dbpedia dataset, we only report the results of CentralSU* due to the memory limitation. For smaller datasets, such as the ratio of sampled nodes $\leq 60\%$, CentralSS achieves better performance than CentralSU on all datasets. As n increases, the execution time of both methods increases, but CentralSU remains within a relatively acceptable runtime. For example, in the full pokec dataset, the runtime of CentralSU is about 180 seconds, while CentralSS takes about 10^4 seconds time. For dbpedia dataset, CentralSU* also achieves competitive accuracy and runtime. Hence, we conclude that the smooth sensitivity-based method is preferable for smaller datasets, while the method based on smooth upper bound on LS is better suited for larger datasets.

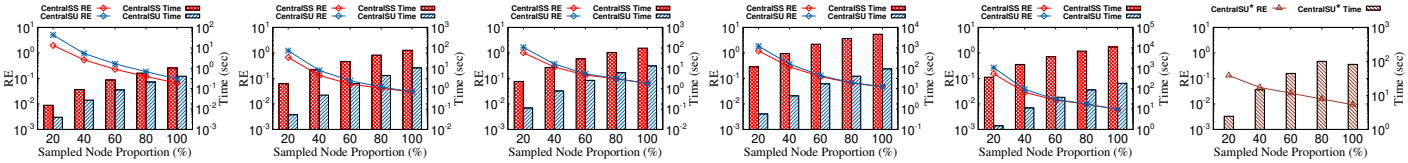
In Table II, we compare the runtime and memory consumption of methods CentralSU and CentralSU* across all datasets, except dbpedia. The results show that CentralSU* considerably reduces memory consumption and runtime. In youtube dataset, for example, CentralSU* consumes about 0.12 GB of memory and costs about one second to run, compared to 17.66 GB and 84 seconds for CentralSU. This confirms the effectiveness of pruning techniques in terms of memory and runtime reduction.

In addition, we randomly sample 20%-80% edges from the youtube dataset to assess the impact of sparsity on scalability. Figure 6(a) shows that the execution time of CentralSS and CentralSU* are relatively stable across sparsity levels, because the time cost of CentralSS is dominated by $\mathcal{O}(n^2)$ and sparsity does not influence the relative distribution of node degrees. In contrast, the runtime of CentralSU decreases with increased



(a) wiki-vote (b) epinions (c) wikisigned (d) youtube (e) pokec (f) dbpedia

Fig. 4: Trade-offs between privacy and relative error of various mechanisms under centralized DP.

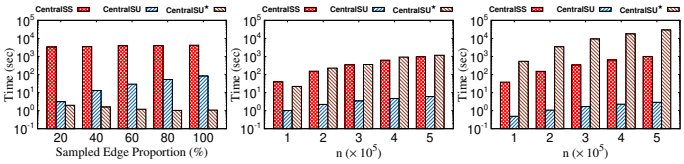


(a) wiki-vote (b) epinions (c) wikisigned (d) youtube (e) pokec (f) dbpedia

Fig. 5: Relative errors and runtime for CentralSS vs. CentralSU (CentralSU*) at different sampling node ratios ($\epsilon = 0.1$).

TABLE II: Comparison of memory consumption (GB) and runtime (seconds) between CentralSU and CentralSU*.

Metric	Method	WV	EP	WS	YT	PK
Memory	CentralSU	0.06	1.20	2.27	17.66	22.60
	CentralSU*	**0.01**	**0.02**	**0.03**	**0.12**	**0.55**
Runtime	CentralSU	0.45	12.09	14.88	83.74	175.51
	CentralSU*	**0.03**	**0.26**	**0.16**	**1.04**	**9.10**



(a) youtube (b) Barabási-Albert (c) Erdős-Rényi

Fig. 6: Comparison of CentralSS, CentralSU and CentralSU* runtimes for different sparsity levels and degree distributions.

sparsity, since its time complexity depends on both the number of edges and the maximum degree. We further evaluate performance on synthetic graphs with different degree distributions. As shown in Figure 6(b) and (c), CentralSU significantly outperforms other methods. In addition, CentralSS and CentralSU exhibit stable runtimes across degree distributions. Conversely, CentralSU* is more sensitive to the degree distribution, with a noticeable preference for power-law distributed graphs.

Local Model. To evaluate the effectiveness of LocalTwoSU, we compare it with LocalOne and LocalTwoGS. We vary the privacy cost from 1.0 to 5.0, with results illustrated in Figure 7. Note that LocalOne is only tested on the wiki-vote dataset, the smallest dataset in our study, since its $\mathcal{O}(n^3)$ time complexity makes it impractical for other datasets [20], [22]. In summary, LocalTwoSU consistently outperforms all competitors on all datasets. While LocalOne satisfies a stricter privacy notion, it injects too much noise, which reduces the estimation accuracy.

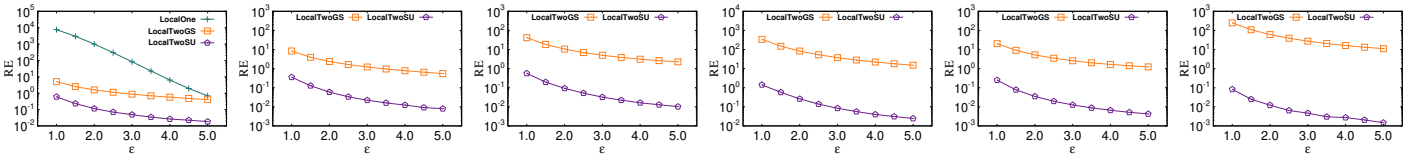
Figure 8 presents the execution time of LocalTwoSU on the wiki-vote and youtube datasets under different sampled node ratios. The primary computational cost arises from the GRR perturbation mechanism and local counting at the node side, with negligible server-side time. The runtime of GRR and local computation both increases with the sampled node ratio. On small datasets, the difference between GRR and local computation is minimal, but on large datasets, the runtime of GRR substantially exceeds that of local computation, which is consistent with the theoretical analysis of the time complexity.

Then, we evaluate the influence of empirical estimation on relative error. Figure 9 illustrates the comparative performance of LocalTwoSU versus the approach without empirical estimation across various privacy cost. It is evident that LocalTwoSU clearly outperforms the method w/o empirical estimation in all cases, which indicates that the unbiased correction does reduce the estimation error.

Case Studies. To show the effectiveness of our methods with respect to privacy preservation and to provide useful statistics, we conduct case studies on datasets extracted from wiki-RfA and bitcoin social networks. Figure 10(a) illustrates the relationships of seven nodes in the wiki-RfA dataset, where nodes represent Wikipedia members and edges represent votes. The signs of edges indicate the support or opposition of the votes. For example, if "Cyde" knows the relationships between other five members, he could deduce the specific votes of "Computerjoe", $(\text{Computerjoe}, \text{Alphax}, -)$, $(\text{Computerjoe}, \text{JoshuaZ}, -)$, and $(\text{Computerjoe}, \text{Matt Yeager}, +)$, which violates the privacy of "Computerjoe". This inference is based on the analysis of $T^b = 7$ and $T^u = 5$. Our proposed mechanisms, with privacy cost $\epsilon = 3.5$, output estimated counts 7.50 and 5.77, which are close to the actual counts and prevent such privacy breaches. For the bitcoin dataset, a who-trusts-whom network, similarly, member "bigbitz" can infer the trust relationships of "csm" by the accurate counts 8 and 3, combined with the knowledge of the relationships between other network members. To address privacy concerns, our methods obfuscates these counts to 8.65 and 4.62, which enhances the privacy protection of "csm".

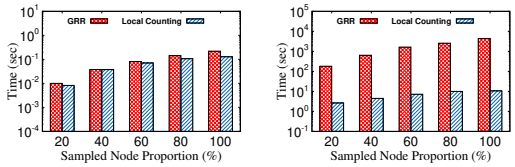
VI. Related Work

Graph Analysis under DP. Centralized DP mechanisms are predicated on the existence of a trusted curator. A substantial portion of graph analysis studies under centralized DP focuses on the estimation of graph statistics. [11] pioneers the concept



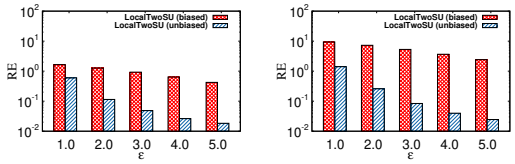
(a) wiki-vote (b) epinions (c) wikisigned (d) youtube (e) pokec (f) dbpedia

Fig. 7: Trade-offs between privacy and relative error of distinct mechanisms under local DP.



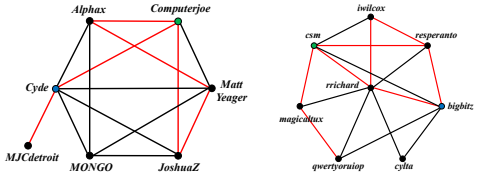
(a) wiki-vote (b) youtube

Fig. 8: Runtime of Generalized Randomized Response and local counting in LocalTwoSU.



(a) wiki-vote (b) youtube

Fig. 9: Relative error comparison between biased and unbiased LocalTwoSU under local DP.



(a) wiki-RfA (b) bitcoin

Fig. 10: Case studies on datasets exacted from wiki-RfA and bitcoin, where black lines represent positive relationships and red lines indicate negative relationships.

of smooth sensitivity and applies it to estimate the costs of minimum spanning trees and triangle counts. This method has been extended to other subgraph count queries, such as k-stars and k-triangles [12]. [13] proposes ladder functions to provide empirical improvements with efficient time complexities. Besides, techniques are developed for other problems [14]–[16].

Apart from graph statistics, researchers have also developed various methods to address other graph-related problems under centralized DP. Some research have explored the problem of releasing specific node subsets, such as the vertex cover [28] and densest subgraphs [29], [30]. Another research direction is the construction of private synthetic graphs [31], [32]. Despite these advancements, there is still an absence of discussion in the literature on the estimation of the number of balanced and unbalanced triangles in signed graphs under centralized DP.

Graph Analysis under LDP. Local DP assumes that the data curator is not trusted. [17] proposes a multi-phase framework for graph generation. [19] introduces the LF-GDPR framework to estimate various graph metrics. The problem of subgraph counting under LDP has attracted attention in research [18], [20]–[22], [33], [34]. [18] estimates the number of subgraphs under the assumption that each user permits his/her friends to view all his/her relationships, but this requirement does not work in many practical scenarios. To address this inadequacy, [34] proposes an enhanced privacy protection scheme named Edge Relationship LDP and develops a federated estimator for triangle counting. [33] applies the randomized response mechanism to the adjacency matrix. However, this perturbation approach introduces a substantial bias to the estimation. [20] estimates the number of k-stars and triangles by multiple rounds of interactions to reduce estimation errors and employs edge sampling techniques to improve communication efficiency [21]. However, none of the current LDP methods are specifically tailored to privately release the number of balanced and unbalanced triangles for signed graphs.

Signed Graph Analysis. The origins of signed graph analysis can be traced back to the realm of social psychology, and the structural balance theory is first presented in [35]. Subsequent research has explored diverse applications such as link prediction [1], [36], [37], balancedness analysis [5], [9], community detection [4], [10], [38], and cohesive subgraph mining [3], [6], [39]. More recently, Arya et al. [7] introduce an efficient algorithm to estimate the number of balanced and unbalanced triangles. Despite these advances, the direct release of graph statistics continues to pose risks to individual privacy, an issue our research endeavors to address.

VII. Conclusion

In this paper, we propose a series of algorithms designed for counting balanced and unbalanced triangles under centralized and local differential privacy, respectively. In the centralized model, our smooth-sensitivity-based method establishes a new benchmark for effectiveness. Meanwhile, our differentially private solution based on smooth upper bound on local sensitivity not only demonstrates its superior efficiency but also provides reliable estimations for most signed graphs. In the local model, our proposed response mechanism introduces less noise to the true results, enhancing data utility. As future work, we intend to extend these methods to count more complex subgraphs in signed networks, such as cliques and 4-cycles.

Acknowledgment

This research is supported by the NSFC Grant U2241211 and the Guangdong Provincial Ordinary Universities' Special Innovation Program (2024KTSCX260). We sincerely thank Prof. Xiaokui Xiao for his insightful and valuable comments. Rong-Hua Li is the corresponding author of this paper.

REFERENCES

[1] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *WWW*, 2010, pp. 641–650.

[2] J. Tang, C. Aggarwal, and H. Liu, "Recommendations in signed social networks," in *WWW*, 2016, pp. 31–40.

[3] R.-H. Li, Q. Dai, L. Qin, G. Wang, X. Xiao, J. X. Yu, and S. Qiao, "Signed clique search in signed networks: concepts and algorithms," *TKDE*, vol. 33, no. 2, pp. 710–727, 2019.

[4] R. Sun, C. Chen, X. Wang, Y. Zhang, and X. Wang, "Stable community detection in signed social networks," *TKDE*, vol. 34, no. 10, pp. 5051–5055, 2020.

[5] P. K. Pandey, B. Adhikari, M. Mazumdar, and N. Ganguly, "Modeling signed networks as 2-layer growing networks," *TKDE*, vol. 34, no. 7, pp. 3377–3390, 2020.

[6] R. Sun, C. Chen, X. Wang, W. Zhang, Y. Zhang, and X. Lin, "Efficient maximum signed biclique identification," in *ICDE*, 2023, pp. 1313–1325.

[7] A. Arya, P. K. Pandey, and A. Saxena, "Balanced and unbalanced triangle count in signed networks," *TKDE*, vol. 35, no. 12, pp. 12 491–12 496, 2023.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.

[9] T. Derr, C. Aggarwal, and J. Tang, "Signed network modeling based on structural balance theory," in *CIKM*, 2018, pp. 557–566.

[10] Y. Kang, W. Lee, Y.-C. Lee, K. Han, and S.-W. Kim, "Adversarial learning of balanced triangles for accurate community detection on signed networks," in *ICDM*, 2021, pp. 1150–1155.

[11] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *STOC*, 2007, pp. 75–84.

[12] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, "Private analysis of graph structure," *TODS*, vol. 39, no. 3, pp. 1–33, 2014.

[13] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Private release of graph statistics using ladder functions," in *SIGMOD*, 2015, pp. 731–745.

[14] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, "Analyzing graphs with node differential privacy," in *TCC*, 2013, pp. 457–476.

[15] W.-Y. Day, N. Li, and M. Lyu, "Publishing graph degree distribution with node differential privacy," in *SIGMOD*, 2016, pp. 123–138.

[16] X. Ding, S. Sheng, H. Zhou, X. Zhang, Z. Bao, P. Zhou, and H. Jin, "Differentially private triangle counting in large graphs," *TKDE*, 2021.

[17] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *CCS*, 2017, pp. 425–438.

[18] H. Sun, X. Xiao, I. Khalil, Y. Yang, Z. Qin, H. Wang, and T. Yu, "Analyzing subgraph statistics from extended local views with decentralized differential privacy," in *CCS*, 2019, pp. 703–717.

[19] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Lf-gdpr: A framework for estimating graph metrics with local differential privacy," *TKDE*, vol. 34, no. 10, pp. 4905–4920, 2020.

[20] J. Imola, T. Murakami, and K. Chaudhuri, "Locally differentially private analysis of graph statistics," in *USENIX Security*, 2021, pp. 983–1000.

[21] J. Imola, T. Murakami, and K. Chaudhuri, "Communication-efficient triangle counting under local differential privacy," in *USENIX Security*, 2022, pp. 537–554.

[22] J. Imola, T. Murakami, and K. Chaudhuri, "Differentially private triangle and 4-cycle counting in the shuffle model," in *CCS*, 2022, pp. 1505–1519.

[23] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*, 2013, pp. 429–438.

[24] V. A. Farias, F. T. Brito, C. Flynn, J. C. Machado, S. Majumdar, and D. Srivastava, "Local dampening: Differential privacy for non-numeric queries via local sensitivity," *The VLDB Journal*, pp. 1–24, 2023.

[25] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," *NeurIPS*, vol. 27, 2014.

[26] Z. Li, R.-H. Li, F. Jin, and G. Wang, "Privacy-preserving triangle counting in signed graphs," 2024. [Online]. Available: https://github.com/Zening-Li/TC-SG/blob/main/FullVersion.pdf

[27] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *USENIX Security*, 2017, pp. 729–745.

[28] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar, "Differentially private combinatorial optimization," in *SODA*, 2010, pp. 1106–1125.

[29] D. Nguyen and A. Vullikanti, "Differentially private densest subgraph detection," in *ICML*, 2021, pp. 8140–8151.

[30] L. Dhulipala, Q. C. Liu, S. Raskhodnikova, J. Shi, J. Shun, and S. Yu, "Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs," in *FOCS*, 2022, pp. 754–765.

[31] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *KDD*, 2014, pp. 911–920.

[32] Z. Jorgensen, T. Yu, and G. Cormode, "Publishing attributed social graphs with formal privacy guarantees," in *SIGMOD*, 2016, pp. 107–122.

[33] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *ICDE*, 2020, pp. 1922–1925.

[34] Y. Liu, T. Wang, Y. Liu, H. Chen, and C. Li, "Edge-protected triangle count estimation under relationship local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[35] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory." *Psychological review*, vol. 63, no. 5, p. 277, 1956.

[36] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *WWW*, 2013, pp. 1477–1488.

[37] X. Li, H. Fang, and J. Zhang, "Rethinking the link prediction problem in signed social networks," in *AAAI*, vol. 31, no. 1, 2017.

[38] J. Zhao, R. Sun, Q. Zhu, X. Wang, and C. Chen, "Community identification in signed networks: a k-truss based model," in *CIKM*, 2020, pp. 2321–2324.

[39] R. Sun, Q. Zhu, C. Chen, X. Wang, Y. Zhang, and X. Wang, "Discovering cliques in signed networks based on balance theory," in *DASFAA*. Springer, 2020, pp. 666–674.