Publishing Below-Threshold Triangle Counts under Local Weight Differential Privacy

Kevin Pfisterer
The University of Tokyo
Tokyo, Japan
pfisterer@is.s.u-tokyo.ac.jp

Quentin Hillebrand
University of Copenhagen
Copenhagen, Denmark
quentin.hillebrand@di.ku.dk

Vorapong Suppakitpaisarn
The University of Tokyo
Tokyo, Japan
vorapong@is.s.u-tokyo.ac.jp

Abstract

We propose an algorithm for counting below-threshold triangles in weighted graphs under local weight differential privacy. Although many prior studies have considered the setting in which the graph topology is public and only the edge weights are sensitive, to the best of our knowledge, this is the first work to study this privacy notion in the local model. Building on a two-round protocol for locally differentially private triangle counting, we exploit the public graph topology to design a novel algorithmic framework. This leads to significant improvements in both accuracy and scalability. In particular, when the input graph is planar, our algorithm eliminates the covariance arising from distributed triangle counting at nodes; for graphs with bounded degeneracy, it significantly reduces this covariance. Since covariance is the dominant source of error in the counting task, our method achieves accuracy that closely aligns with the lower bound. We also present an efficient algorithm for the computation of the smooth sensitivity and provide experiments that quantify the trade-off between the biased and unbiased variants of our estimator and demonstrate the effectiveness of the proposed improvements.

1 Introduction

Over recent years, differential privacy [14] has emerged as the de facto standard for formal privacy guarantees. It provides a rigorous mathematical framework under which the inclusion or exclusion of any single individual's data has only a limited effect on the released statistics, thereby preventing an adversary from drawing reliable inferences about any particular data record from the published output. Initially developed for tabular datasets, differential privacy was subsequently adapted to graph-structured data via edge-differential privacy, which protects the presence or absence of individual edges in a network [40, 20]. A key limitation of both the central differential privacy and central edge-differential privacy models is their reliance on a trusted curator to collect and sanitize the data. Local differential privacy [29, 43] and its graph analogue, local edge-differential privacy [43], address this concern by shifting randomization to the user side, thereby eliminating the need for a trusted central server and providing strong privacy guarantees.

A substantial body of work has studied differential privacy for weighted graphs, which model practical settings such as road and telecommunication networks. In such settings, the graph topology is typically public, and hence the presence or absence of edges is not regarded as sensitive. The sensitive information instead lies in the edge weights, which must be protected. Representative results include

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

differentially private shortest-path algorithms [46, 7, 12, 13], mechanisms for releasing privatized versions of weighted graphs [34, 52], and algorithms for privately releasing cut values [1, 11].

In this work, we strengthen the line of study discussed in the previous paragraph by considering its local analogue. Although several works have studied local differential privacy for weighted graphs [49, 8, 51], to the best of our knowledge, all prior work treats both edge existence and edge weights as sensitive attributes. However, hiding the graph topology is overly restrictive and introduces unnecessary error in applications where the underlying infrastructure is already public knowledge.

The local weight differential privacy framework is well suited to several practical applications. For example, consider a web-traffic model in which nodes represent websites, edges represent hyperlinks, and edge weights capture the volume of users navigating along these links. While the hyperlink structure of the network is public knowledge, the exact traffic volume on each link is sensitive. Since each website can observe referral sources and outbound clicks, it has access to the weights of its incident edges. The local weight differential privacy model therefore enables websites to collaboratively compute global graph statistics while each website independently provides strong privacy guarantees for its own users.

The triangle counting problem has been extensively studied in the unweighted setting under local differential privacy [15, 23, 24, 25, 33, 38]. In the unweighted case, triangle counting plays a central role in applications in graph mining, bioinformatics, and social networks [39, 3, 37]. Since the network topology is public information in our setting, counting the number of triangles is straightforward in our privacy notion.

1.1 Our contributions

We study the problem of counting triangles whose total edge weight is smaller than a given threshold under local weight differential privacy.

Motivation. This triangle count and related measures such as the weighted clustering coefficient have been used to identify congested regions, which are central to urban traffic planning [42, 45], and can also be used to assess graph interconnectivity [30]. By designing mechanisms that count triangles whose weight lies below (equivalently, above) a given threshold, our approach directly supports this line of work while providing strong privacy guarantees. Moreover, efficient triangle-counting techniques often extend to more general subgraph counting, as has been shown in the unweighted setting [6, 48].

Prior differentially private work on shortest-path cycles [46, 7, 12, 13] has primarily assumed non-negative edge weights. Below-threshold triangle identification can also be viewed as a basic building block for detecting negative cycles, with negative triangles being the simplest such structures. Since negative-cycle detection is fundamental to shortest-path computation with arbitrary edge weights, our results provide an entry point for studying negative-cycle detection under local differential privacy.

Moreover, below-threshold triangle counts can be used to generate synthetic graphs that preserve key properties of the original graph. A synthetic graph generated only from the marginal distribution of edge weights may reproduce the overall weight histogram while assigning the weights to edges in a way that substantially changes the weights of triangles and other local structures. Weight-aware triangle and clustering statistics can capture this structural organization of weights, which is not determined by either the unweighted topology or the marginal edge-weight distribution alone [41, 45]. When evaluated at several thresholds, below-threshold triangle counts approximate the empirical cumulative distribution of triangle weights and can therefore serve as target statistics for generating or evaluating synthetic edge weights on the known graph structure.

Algorithm. In Section 3, we propose an algorithm for counting the number of triangles whose total weight falls below a given threshold, fully leveraging the flexibility of the differential privacy notion for weighted graphs. The key contributions of our algorithm are twofold:

We design an assignment that minimizes the variance of the triangle count in Section 4. Each triangle is incident to three nodes. It is known that assigning each triangle to a carefully chosen node and letting only that node report it can greatly reduce the variance and thus the error in triangle counting [23, 38]. In the unweighted setting, however, the server does not know the graph structure, so prior work relied on indirect statistics such as noisy degree estimates to guide this assignment. In contrast, under differential privacy for weighted graphs, the server is allowed to know the graph

structure. By exploiting this additional information, we construct an optimal min-cost max-flow algorithm designed to minimize counting variance. For scenarios where this optimal approach is computationally impractical, we also provide an efficient and scalable approximation algorithm.

Furthermore, we prove lower bounds on the precision of any local weight differentially private algorithm operating on planar and degeneracy-bounded graphs. Because these bounds also apply to the local edge differential privacy setting, they establish theoretical lower bounds for the problem studied in [23]. Finally, we demonstrate that with the optimal assignment, a variant of our algorithm achieves asymptotically optimal precision on planar graphs and bounded-degeneracy graphs.

We develop a scalable algorithm to compute the smooth sensitivity in Section 5. Smooth sensitivity [40] is a standard tool for reducing the amount of noise added to released statistics and can yield substantial accuracy improvements in the weighted setting. However, computing a sufficiently tight smooth sensitivity bound is often computationally expensive in practice [22]. Although there are algorithms for computing smooth sensitivity in the unweighted setting, directly applying those bounds to the weighted setting would drastically overestimate the sensitivity and therefore inject far more noise than necessary. By exploiting the knowledge of graph structure, we develop an algorithm that computes the local β-smooth sensitivity in $O(d^2 \log^2 d)$ time for a node of degree d. The key idea is to maintain a sorted array so that we can capture the relevant local changes without recomputing the sensitivity from scratch. We remark that in typical road or communication networks node degrees are generally well below 10^4. This implies that our smooth sensitivity computation can be carried out within practical running time and is therefore scalable in these settings.

In addition to these two main contributions, in Section 3, we adapt the unbiased estimator from [17], originally proposed for privately releasing histograms, so that it can be used to release an unbiased estimate of the number of triangles under our weighted-graph model. When we use the biased estimator, the resulting error scales linearly with the number of triangles in the input graph. In contrast, with the unbiased estimator, the error scales on the order of the square root of the number of triangles. Consequently, for graphs with many triangles, our algorithm achieves a significantly smaller relative error.

Experiments. In Section 6, we evaluate our method on three real-world graphs: a telecommunication network in Milan, Italy [4], a biological network used for integrated transcriptional and metabolic analysis [35], and a Wikipedia page-page network [44]. We evaluate the assignment algorithms on synthetic complete, planar graphs, and a Google web graph. The experiments highlight the trade-off between a single round and two round algorithm and between a biased and unbiased version. Additionally, they show that, for moderate privacy budgets ε, our unbiased version performs the best.

2 Preliminaries

2.1 Problem definition

Let $G = (V, E, w)$ be a weighted undirected graph with vertex set $V = \{1, \ldots, n\}$ and edge set $E \subseteq \binom{V}{2}$. For each edge $\{u, v\} \in E$, let $w_{uv} = w_{vu} \in \mathbb{Z}$ denote its (integer) weight; we restrict to integer weights, which suffice for all applications considered in this paper. For a vertex $v \in V$, write $\mathcal{N}(v) = \{u \in V : \{u, v\} \in E\}$ for its neighborhood and $d_v = |\mathcal{N}(v)|$ for its degree. The maximum degree is denoted by d_{max}. The incident—weight vector at v is $w^v = (w_{vu})_{u \in \mathcal{N}(v)} \in \mathbb{Z}^{d_v}$. Throughout, the graph structure (V, E) is public, whereas each weight w_{uv} is known only to its endpoints u and v; accordingly, w^v constitutes the private data of node v.

We denote by Δ the set of triangles of G, where a triangle $T \in \Delta$ is either represented by its three edges or its three nodes. We use the edge and node representation of a triangle interchangeably and clarify the representation used where necessary. We now present the formal definition of the below-threshold triangle count problem.

Definition 1. *Given a weighted undirected graph $G = (V, E, w)$ and a threshold $\lambda \in \mathbb{Z}$, the goal is to output $f(G) = \sum_{\{v,u,x\} \in \Delta} 1\{w_{vu} + w_{vx} + w_{ux} < \lambda\}$.*

The framework extends directly to directed graphs. When Δ is replaced by any target class of directed triangles (e.g., cyclic triangles $u \to v \to w \to u$), the assignment, the estimators, and their analysis apply to this set unchanged, with the relevant statistics counted with respect to the chosen class.

2.2 Local weight differential privacy

In this section, we introduce the formal notations for our privacy setting together with a discrete and a continuous privacy mechanism. In the setting of local weight differential privacy, it is assumed that the graph topology (nodes and edges) is public and the private information we want to protect is encoded in the edge weights. This assumption is considered in several previous works such as [46, 52], and is natural in applications such as road and telecommunication networks, where the underlying connectivity (streets, intersections, or network links) is typically public, while sensitive usage information is encoded in the edge weights.

Definition 2 (Neighboring weight vectors)**.** *We define two weight vectors $w, w' \in \mathbb{Z}^d$ as neighboring $w \sim w'$ if their l_1 distance is at most one, i.e. $w \sim w' \iff ||w - w'||_1 \leq 1$*

Definition 3 (Local weight differential privacy)**.** *Let $\varepsilon > 0$. A randomized query $\mathcal{R}$ satisfies ε-local weight differential privacy (ε-LWDP) if, for any possible neighboring weight vectors w, w', and any possible outcome set S, $\Pr[\mathcal{R}(w) \in S] \leq e^{\varepsilon} \Pr[\mathcal{R}(w') \in S]$.*

An algorithm $\mathcal{A}$ is said to be ε-LWDP if, for any node v_i, and any sequence of queries $\mathcal{R}_1, \ldots, \mathcal{R}_\kappa$ issued to v_i, where each query $\mathcal{R}_j$ satisfies ε_j-local weight differential privacy (for $1 \leq j \leq \kappa$), the total privacy loss is bounded by $\varepsilon_1 + \cdots + \varepsilon_\kappa \leq \varepsilon$. We call ε the privacy budget *of any ε-LWDP algorithm and query.*

LWDP is a per-node guarantee. When v's incident—weight vector changes by a unit of one, the probability of any report of v changes by a factor of at most e^{ε}. Since each edge weight counts communication or mobility events, this bounds what an adversary can infer about whether a specific event occurred. As an event of edge $\{u, v\}$ belongs to both endpoints u and v, its guarantee is at most 2ε by composition, analogous to edge LDP [24].

Our results can be extended to the relation $\|w - w'\|_1 \leq r$. Setting $r = 2$ already protects a full trip along a simple path, which changes each incident—weight vector by at most two units. The details are deferred to Appendix F.

We now introduce the private queries that we use. The first one is the discrete Laplace query.

Definition 4 (Discrete Laplace query [18])**.** *For $p \in (0, 1)$, $DLap(p)$ denotes the discrete Laplace distribution with probability mass at $i \in \mathbb{Z}$ of $\frac{1-p}{1+p} \cdot p^{|i|}$. For a weight vector $w = (w_1, \ldots, w_d) \in \mathbb{Z}^d$, the discrete Laplace privacy mechanism adds independently drawn noise from $DLap(e^{-\varepsilon})$ to each entry in w, i.e. $\tilde{w}_i = w_i + DLap(e^{-\varepsilon}) \quad \forall i \in [d]$. The discrete Laplace query, also known as geometric query, satisfies ε-LWDP.*

The second query is the Laplace query, whose definition relies on global sensitivity.

Definition 5 (Global sensitivity [14])**.** *For a function $f : \mathbb{Z}^d \to \mathbb{R}$, the global sensitivity $GS(f)$ is defined as $GS(f) := \max_{w \sim w'} ||f(w) - f(w')||_1$.*

Definition 6 (Laplace query [14])**.** *For a function $f : \mathbb{R}^d \to \mathbb{R}$ and $\varepsilon > 0$, the following is defined as the Laplace query $LM_f(w) = f(w) + Y$ where Y is drawn from $Lap(GS(f)/\varepsilon)$. The Laplace query satisfies ε-local weight differential privacy.*

Global sensitivity together with the Laplace query provides a guarantee for all input graphs. Meanwhile, smooth sensitivity allows us to calibrate the noise to the underlying private data.

Definition 7 (Local sensitivity [40])**.** *For a function $f : \mathbb{Z}^d \to \mathbb{R}$ and a weight vector $w \in \mathbb{Z}^d$, the local sensitivity of f at w is $LS_f(w) = \max_{y : y \sim w} |f(w) - f(y)|$*

Local sensitivity depends on the input weight vector $w^v \in \mathbb{Z}^{d_v}$ associated with a vertex $v \in V$ of degree d_v. The β-smooth sensitivity is defined as follows:

Definition 8 (β-Smooth sensitivity [40])**.** *For $\beta > 0$, the β-smooth sensitivity of f at w is defined as $S^*_{f,\beta}(w) = \max_{y \in \mathbb{Z}^d} \left(LS_f(y) e^{-\beta \cdot d(w,y)} \right)$*

We then give the definition of the smooth sensitivity query below.

Definition 9 (β-smooth sensitivity query [40, 50])**.** *For a function f, $\varepsilon > 0$ and $\Gamma > 1$, the following is defined as the β-smooth sensitivity query $SM_f(w) = f(w) + \frac{2(\Gamma-1)^{\frac{\Gamma-1}{\Gamma}}}{\varepsilon} S^*_{f,\varepsilon/(2(\Gamma-1))}(w) \cdot Z$ where Z is drawn from a probability distribution proportional to $1/(1 + |z|^{\Gamma})$. The smooth sensitivity query satisfies ε-LWDP.*

The mechanism remains valid for every $\Gamma > 1$. However, for $\Gamma \leq 3$ we currently do not know how to express $\mathrm{Var}[Z]$ in closed form [22]. In this paper, we fix $\Gamma = 4$, a choice for which $\mathrm{Var}[Z] = 1$.

3 Two-step algorithm for publishing below-threshold triangle counts

In this section, we present our two-step algorithm to release the number of below-threshold triangles under local weight differential privacy. Algorithm 1 proceeds in two rounds of communication, following the two-step paradigm of [24]. In the first step, nodes publish their incident weight vectors so that the central server can construct a noisy graph with this information. In the second step, nodes locally use their private weights together with the noisy graph to accurately estimate the local number of below-threshold triangles. These local counts are published under local weight differential privacy and then aggregated by the central server.

Algorithm 1 TWO-STEP ALGORITHM FOR COUNTING BELOW-THRESHOLD TRIANGLES

Input: weighted graph $G = (V, E, w)$, λ, ε_1, ε_2
Output: below-threshold triangle count $\tilde{k}$
First step: Noisy releases and task assignments
Node $v \in V$:
$\tilde{w}^v \leftarrow w^v + (Y_1, \ldots, Y_{d_v})$ where $Y_i = DLap(e^{-\varepsilon_1})$ for all $i \in [d_v]$. ▷ Discrete Laplace query
Send $\tilde{w}^v$ to the server
Server:
Collect $\tilde{w}^v$ from each $v \in V$
Construct $G' = (V, E, w')$ from $(\tilde{w}^v)_{v \in V}$. ▷ Section 3.1
Compute assignment $\rho : \Delta \mapsto V$ from (V, E). ▷ Section 4
Send Δ_v and $(w'_{ux})_{\{v,u,x\} \in \Delta_v}$ to each node v

Second Step: Local counting
Node $v \in V$:
Let $\Delta_v = \rho^{-1}(v)$. $f'_v(w^v) \leftarrow \sum_{T \in \Delta_v} g_v^T(w^v)$ when $g_v^T(w^v)$ is the estimator declaring if T is below the threshold. ▷ Section 3.2
Compute the sensitivity of f'_v, denoted s_v. ▷ Theorem 1 or Section 5
Compute $\tilde{f}_v(w^v)$, which is the value of $f'_v(w^v)$ released using either the Laplace query (Definition 6) or the smooth-sensitivity query (Definition 9). We set the privacy budget of this release to ε_2.
Send $\tilde{f}_v(w^v)$ to the server
Server:
return $\tilde{k} \leftarrow \sum_{v \in V} \tilde{f}_v(w^v)$

3.1 Noisy releases and task assignments

Each vertex v releases its incident–weight vector using the discrete Laplace query (Definition 4), yielding $\tilde{w}^v = (\tilde{w}_{vu})_{u \in \mathcal{N}(v)}$ for all $v \in \{1, \ldots, n\}$. A central server aggregates the reports and constructs a noisy graph $G' = (V, E, w')$ by enforcing symmetry via an index tie-break: $w'_{uv} = \tilde{w}_{uv}$ if $u < v$, and $w'_{uv} = \tilde{w}_{vu}$ otherwise, for all $\{u, v\} \in E$.

Let Δ denote the set of triangles in G. The server assigns each triangle $T \in \Delta$ to one of its vertices via a mapping $\rho : \Delta \to V$. For each node $v \in V$, let $\Delta_v = \rho^{-1}(v)$ denote the set of triangles for which v is responsible. In Section 4, we define ρ to minimize the expected error. The server then notifies each node v of its assigned set Δ_v. Moreover, for every triangle $T = \{v, u, x\} \in \Delta_v$, the server transmits to v the noisy weight w'_{ux} of the non-incident edge $\{u, x\}$.

3.2 Local counting

Each node v locally counts the number of below-threshold triangles among its assigned set $\Delta_v = \rho^{-1}(v)$. At this stage, v knows the public topology (V, E), its private incident-weight vector w^v, and the noisy weights w'_{ux} for all triangles $\{u, v, x\} \in \Delta_v$ released in Step 1. We consider two estimators for declaring whether T is below the threshold; in Algorithm 1 this is represented by the placeholder

$g_v^T(w^v)$. Node v applies the chosen estimator to each $T \in \Delta_v$ and counts those below the threshold. We first introduce a biased estimator.

Definition 10 (Biased estimator)**.** *Let $v \in V$ be the vertex responsible for a triangle $T = \{v, u, x\} \in \Delta_v$. We define the following* biased *estimator: $B'_T = \mathbf{1}\{ w_{vu} + w_{vx} + w'_{ux} < \lambda \}$, where w_{vu} and w_{vx} are true edge weights incident to v and w'_{ux} is the noisy weight from Step 1.*

We next recall the *unbiased* estimator of [17], originally proposed for releasing anonymous histograms without bias.

Definition 11 (Unbiased estimator)**.** *Let $h : \mathbb{Z} \to \mathbb{R}$ be defined by setting $h(m) = 0$ if $m > \lambda$, $h(m) = -p/(1-p)^2$ if $m = \lambda$, $h(m) = 1 + p/(1-p)^2$ if $m = \lambda - 1$, and $h(m) = 1$ if $m < \lambda - 1$, where $p = e^{-\varepsilon_1}$. Let $v \in V$ be the vertex responsible for counting the triangle $T = \{v, u, x\} \in \Delta_v$. We define the estimator U'_T: $U'_T = h(w_{vu} + w_{vx} + w'_{ux})$.*

Both estimators B'_T and U'_T use the true incident weights w_{vu} and w_{vx} together with the noisy value w'_{ux}. Define f'_v as the local below-threshold count at node v, with g a placeholder for the chosen estimator: $f'_v(w^v) = \sum_{T \in \Delta_v} g_v^T(w^v)$.

We set $g_v^T(w^v) = B'_T$ when using the biased estimator; otherwise, we set $g_v^T(w^v) = U'_T$.

3.3 Publishing local counts

Because computing the local below-threshold count $f'_v(w^v)$ depends on the private incident-weight vector w^v, we must apply a privacy mechanism to ensure local weight differential privacy before sending $f'_v(w^v)$ to the central server. Note that the vector w'_v from Step 1 is already public and thus treated as a constant here. Since the final output does not need to be an integer, we use the Laplace query of Definition 6 or the smooth-sensitivity query of Definition 9 (Section 5). The proofs of all theorems in this section can be found in Appendix A.

Theorem 1 (Global sensitivity of f'_v)**.** *Let Δ_v denote the set of triangles v is responsible for. The global sensitivity of f'_v is given by $GS(f'_v) = GS(g_v^T) \max_{u:\{v,u\} \in E} |\{T \in \Delta_v : \{v, u\} \in T\}|$ where $GS(g_v^T) = 1$ when using the biased estimator and $GS(g_v^T) = 1 + 2\frac{p}{(1-p)^2}$ when using the unbiased estimator.*

The computation of $GS(f'_v)$ depends only on public data. Therefore, each node $v \in V$ can compute the sensitivity $GS(f'_v)$ locally. For $v \in V$ with degree d, the quantity $\big|\{T \in \Delta_v : e \in T\}\big|$ can be computed in $O(d)$ time by scanning each other edge incident to v and checking whether it closes a triangle. Repeating this for all incident edges yields a total running time of $O(d^2)$.

3.4 Analysis of two-step algorithm

In the following we prove local weight differential privacy and analyze the communication costs and precision of Algorithm 1.

Theorem 2 (Privacy of Algorithm 1)**.** *Algorithm 1 satisfies $(\varepsilon_1 + \varepsilon_2)$-local weight differential privacy.*

Proof. Algorithm 1 issues two queries to each node. The first retrieves the incident weight vector of a node $v \in V$; v responds using the discrete Laplace query, achieving ε_1-LWDP. The second retrieves the local below-threshold count; here the Laplace query or smooth-sensitivity query is used, both achieving ε_2-LWDP. □

Theorem 3 (Communication cost of Algorithm 1)**.** *With probability at least $1 - \frac{1}{m}$, the communication cost of Algorithm 1 is $\tilde{O}(m + |\Delta|)$.*

We now bound the expected squared ℓ_2-error of the unbiased and biased estimator. The error arises from the bias and variance of the estimator g_v^T and the Laplace query for publishing the local counts. The global sensitivity from Theorem 1 is bounded by the degree of a vertex resulting in a dependency on d_{max} in the error bound. For two vertices $x, y \in V$, let $N_{x,y} = \{v \in V : \{v, x, y\} \in \Delta_v\}$ denote the set of vertices that form a triangle with x, y and use the noisy information w'_{xy}. For $v, u \in N_{x,y}$ both use the same noisy information w'_{xy} which results in a covariance between the estimator with $T = \{v, x, y\}$ and with $T' = \{u, x, y\}$. We call this structure a C'_4 instance. Each instance of C'_4,

represented by the pair of triangles T and T', induces covariance between the counts at nodes u and v. This covariance is one of the dominant terms in the counting error. Consequently, the error scales with the number of such instances, denoted by $\#C_4'$.

Theorem 4 (Precision of biased estimator)**.** *The expected squared ℓ_2-error of Algorithm 1 using the biased estimator is bounded by $O\left(\frac{d_{max}m}{\varepsilon_2^2} + \frac{|\Delta|^2}{\varepsilon_1^2} + \frac{\#C_4'}{\varepsilon_1}\right)$.*

Theorem 5 (Precision of unbiased estimator)**.** *The expected squared ℓ_2-error of Algorithm 1 with the unbiased estimator is bounded by $O\left(\frac{d_{max}m}{\varepsilon_1^4\varepsilon_2^2} + \frac{|\Delta|}{\varepsilon_1^3} + \frac{\#C_4'}{\varepsilon_1^3}\right)$.*

Theorems 4 and 5 hold for both choices in Line 9. With $\Gamma = 4$, the noise added by the smooth-sensitivity query has variance $(2 \cdot 3^{3/4}/\varepsilon_2)^2\,\mathbb{E}[(S^*_{f_v'})^2]$, and since $S^*_{f_v',\beta}(w) \leq GS(f_v')$ for all w, this is $O(GS(f_v')^2/\varepsilon_2^2)$, the same order as for the Laplace query (Appendix A.3).

We observe that the unbiased estimator achieves a smaller error when the number of triangles $|\Delta|$ is large, whereas the biased estimator is more accurate when the privacy budgets ε_1 and ε_2 are small.

4 Assignment function ρ

In this section, we provide an algorithm that computes an assignment from triangles to nodes $\rho : \Delta \mapsto V$ to minimize the number of C_4' instances, which is denoted by $\#C_4'$. Let $l(e)$ denote the *load* of an edge $e = \{u, v\} \in E$ representing the number of estimators that use the noisy weight w'_{uv}, i.e. the number of triangles $T \in \Delta$ with $u, v \in T$ such that $\rho(T) \notin \{u, v\}$. We define the assignment costs $c(\rho)$ as the number of C_4' instances generated by ρ: $c(\rho) = \#C_4' = \sum_{e\in E} \binom{l(e)}{2}$. We need to assign every triangle T to one of its edges such that $\#C_4'$ is minimized. Since the sum of loads $\sum_{e\in E} l(e)$ is always equal to the number of triangles $|\Delta|$, minimizing $c(\rho)$ is equivalent to minimizing the sum of squared loads. This is formalized in the following optimization problem. For a triangle $T \in \Delta$ and edge $e \in T$, let $x_{T,e} \in \{0, 1\}$ denote whether T is assigned to e. We calculate $(x_{T,e})_{T\in\Delta, e\in T} \in \{0, 1\}^{|\Delta|\times 3}$ such that $\sum_{e\in T} x_{T,e} = 1$ for all $T \in \Delta$ and $\sum_{e\in E} \left(\sum_{T\in\Delta: e\in T} x_{T,e}\right)^2$ is minimized.

Theorem 6. *Given a graph $G = (V, E)$, the optimal assignment can be computed in $O(|\Delta|^2 + |\Delta|(|\Delta| + m)\log(|\Delta| + m))$ time.*

All proofs for this section are deferred to Appendix B. In dense graphs, the number of triangles $|\Delta|$ scales with $O(n^3)$ and computing the optimal assignment becomes impractical. We therefore introduce a greedy approach that iterates over all triangles and assigns a triangle to the edge with the currently lowest load.

Theorem 7 (Approximation ratio of greedy algorithm)**.** *Let $\#C'_{4,ALG}$ be the number of C_4' instances produced by the greedy algorithm, and $\#C'_{4,OPT}$ be the number of instances in the optimal assignment. The greedy algorithm achieves the following bound $\#C'_{4,ALG} \leq (3 + 2\sqrt{2})\#C'_{4,OPT} + (1 + \sqrt{2})|\Delta|$.*

We can now strengthen Theorems 4 and 5 replacing $\#C_4'$ with $\#C'_{4,OPT}$. This refinement is possible because we operate under local weight differential privacy. In earlier work on local differential privacy, such as [23], such an improvement is not achievable, since the server does not know the graph structure.

Assignment function for planar graphs: We now discuss the assignment when the input graph is planar.

Lemma 1. *Let G be a planar graph. Then the optimal assignment satisfies $\#C'_{4,\mathrm{OPT}} = 0$. Moreover, such an optimal assignment can be found in $O(n^2 \log n)$ time.*

Lemma 2. *There exists a planar graph G with n nodes such that the expected squared ℓ_2-error of any ε-LWDP algorithm is in $\Omega(n^2)$.*

Theorem 8. *Algorithm 1, instantiated with the biased estimator and the assignment function from Lemma 1, achieves asymptotically optimal precision on planar graphs.*

Proof. For planar graphs, $|\Delta| = O(n)$, $d_{max} \leq n$, and $\#C_4' = 0$. Plugging this into Theorem 4 with $\varepsilon_1 = \varepsilon_2 = \varepsilon/2$ yields an upper bound of $O(n^2/\varepsilon^2)$ while Lemma 2 provides the lower bound. □

Assignment function for bounded-degeneracy graphs: In the following, let γ denote the degeneracy of the input graph. Triangle counting under local edge differential privacy (LEDP) on bounded-degeneracy graphs was analyzed in [23]. However, because the structures inducing covariance differ slightly between LEDP and LWDP, the results of [23] do not directly carry over. We therefore provide proofs tailored to the LWDP setting. In addition, we establish two separate lower bounds under LWDP and LEDP, respectively, addressing whether the method of [23] can be improved.

Lemma 3. *Let G be a graph with degeneracy γ, then there exists a triangle-node assignment with $c(\rho) = O(n\gamma^3)$.*

Lemma 4. *For a graph G with degeneracy γ, the expected squared ℓ_2-error of Algorithm 1 with unbiased estimator and privacy budget $\varepsilon_1, \varepsilon_2$ is bounded by $O(\frac{n^2\gamma}{\varepsilon_1^4\varepsilon_2^2} + \frac{n\gamma^3}{\varepsilon_1^3})$.*

Proof. $|\Delta|$ is upper bounded by $O(n\gamma^2)$, $m = O(n\gamma)$, $d_{max} \leq n$, and by Lemma 3 there exists an assignment such that $\#C_4' \in O(n\gamma^3)$. The claim then follows from Theorem 5. □

We now establish lower bounds for the expected squared ℓ_2-error on graphs with degeneracy γ. Our analysis adapts the construction from [15], which proved an $\Omega(n^3/\varepsilon^2)$ lower bound for triangle counting under local edge differential privacy, via a reduction from computing the sum of either γ or $n - \gamma$ values.

Theorem 9. *There exist families of graphs with degeneracy γ such that any ε-LWDP algorithm outputting the number of below-threshold triangles with an additive error of α with probability at least $2/3$ must satisfy $\alpha \in \Omega\left(\frac{1}{\varepsilon}\max\{(n-\gamma)\sqrt{\gamma},\, \gamma\sqrt{n-\gamma}\}\right)$.*

Theorem 9 shows that the error achieved by the unbiased estimator is close to optimal. The $O(n^2\gamma)$ dependency is tight according to Theorem 9 while there remains a γ factor gap between the $O(n\gamma^3)$ term and the $\Omega(n\gamma^2)$ bound of Theorem 9. In Appendix E.4 we provide empirical results on the running times and costs of the assignment algorithms proposed in this section.

5 Efficient smooth sensitivity calculation

In this section, we will use the smooth-sensitivity query from Definition 9 to ensure privacy during the second step of our algorithm. The second step publishes the local below-threshold count of $v \in V$. We saw that the global sensitivity is in $O(d_v)$. This is due to the fact that the increase or decrease of a single edge weight affects at most $d_v - 1$ triangles and, therefore, might change all triangle weights from being less than λ to being at least λ or vice versa in the worst case. However, in most cases the sensitivity is much smaller.

We now present our algorithm to compute the smooth sensitivity of the local count f_v' for both the biased and unbiased estimators. The core idea is to fix the edge whose weight increases or decreases during the local sensitivity computation, and then allocate weights such that this local sensitivity is maximized while balancing the penalty term. We demonstrate that by maintaining a sorted array of partial triangle weights, this weight distribution step, a bottleneck operation that has to be conducted d times, can be carried out in $O(d \log^2 d)$ time. This is a significant improvement over the straightforward $O(d^2)$ approach. Full details are deferred to Appendices C and D.

Theorem 10 (Computation time of smooth sensitivity)**.** *There exists an algorithm that computes the β-smooth sensitivity of f_v' for a node v with degree d in $O(d^2 \log^2 d)$ time.*

This improvement allows us to replace the Laplace query, used in Step 2 to publish local triangle counts, with the β-smooth sensitivity query from Definition 9. As demonstrated in Section 6, this approach yields utility benefits while our proposed algorithm ensures that the overall computation time remains practical.

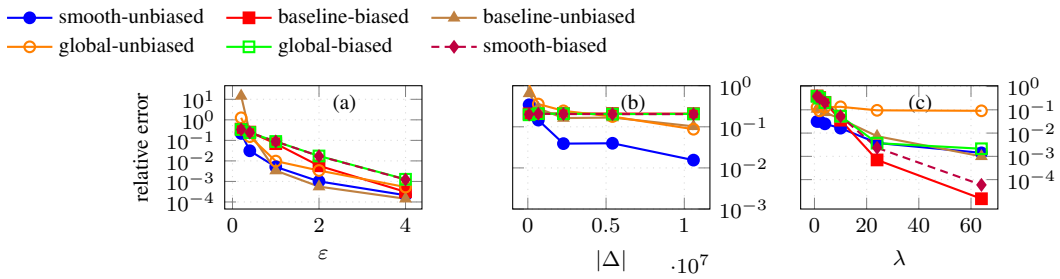


Figure 1: Relative error across all methods under different settings for the input graph ML-TELE-400: (a) as a function of the privacy budget ε for $\lambda = 4$, (b) as a function of the number of triangles $|\Delta|$ for $\lambda = 4$ and $\varepsilon = 0.5$, (c) as a function of the threshold λ for $\varepsilon = 0.5$.

6 Experiments

In this section, we evaluate four variants of Algorithm 1 on a graph (ML-TELE-400) derived from telecommunication data collected in the city of Milan, Italy. The details of the data to graph conversion are given in Appendix E. The implementation used in our experiments is publicly available[1]. All experiments were run on the SHIROKANE supercomputer, provided by the Human Genome Center, Institute of Medical Science, The University of Tokyo, using up to 3 cores and at most 7 GB RAM[2].

We evaluate four variants of Algorithm 1: *global-biased*, *global-unbiased*, *smooth-biased*, and *smooth-unbiased*. The *biased* variants apply Algorithm 1 with the biased estimator B'_T to compute the local below-threshold triangle counts, while the *unbiased* variants use the unbiased estimator U'_T. For Step 2, the *global* variants apply the Laplace query from Definition 6, and the *smooth* variants apply the smooth-sensitivity query from Definition 9. In all cases, we use the approximation algorithm (Theorem 7) to compute the assignment function $\rho : \Delta \mapsto V$. We also compare these methods against two non-interactive baselines. In both approaches, the central server requests the incident weight vector of each node v using the discrete Laplace mechanism, and then estimates the below-threshold triangle counts solely from the resulting noisy weights. For *baseline-biased*, a triangle is estimated by $1\{\tilde{w}_{uv} + \tilde{w}_{vx} + \tilde{w}_{ux} < \lambda\}$, while *baseline-unbiased* uses the post-processed unbiased estimator of [21].

We fix $\lambda = 4$ and we use a total privacy budget of $\varepsilon = 0.5$ for both the baseline and our method; in our method we split this as $\varepsilon_1 = \varepsilon_2 = 0.25$. In our experiments, we executed each algorithmic variant $N = 10$ times and report the relative root-mean-square error $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - r)^2} \,/\, r$, where r is the true number of triangles below the threshold and p_i is the number of triangles reported by the algorithm in the i-th run. We selected 10 repetitions because this was sufficient to obtain stable estimates across all experimental settings we evaluated.

Figure 1 shows the results on ML-TELE-400. The smooth-unbiased variant attains a relative error below 0.05 in almost all settings with $\varepsilon \geq 0.4$. Despite the fact that the graph has very high degree and all computations are performed centrally on the server rather than being distributed across nodes, the total runtime remains below five minutes even for the smooth-sensitivity mechanism. This demonstrates that the smooth-sensitivity approach introduced in Section 5 is both scalable and practical for large graphs.

Figure 1(a) shows the existence of three distinct regimes. For moderate values of ε ($\varepsilon \approx 0.5$), the *smooth-unbiased* variant performs best. We observe that, for $\varepsilon = 4$, *smooth-unbiased*, *baseline-unbiased* and *baseline-biased* achieve comparable relative error. This is because a privacy budget of this size allows all methods to produce highly accurate estimates. The unbiased variants are only worse than the biased estimator when $\varepsilon \ll 1$, which matches the theoretical dependence. Moreover, Figure 1(a) shows that *baseline-unbiased* performs several orders of magnitude worse than *smooth-unbiased* on smaller ε.

[1] Source code: `https://github.com/Crightub/private-below-threshold-triangle-counting`

[2] For hardware specifications, see `https://supcom.hgc.jp/english/sys_const/system-main.html`.

Figure 1(b) reports the results as we vary the number of triangles $|\Delta|$ in the input graph. We uniformly sample vertex subsets of various sizes and run each algorithm on the induced subgraph. Under this sampling procedure, the true below-threshold triangle count grows linearly in $|\Delta|$. The results are consistent with Theorem 4, which states that the biased estimator's squared ℓ_2-error grows quadratically in $|\Delta|$, resulting in a constant relative ℓ_2-error as depicted. Theorem 5 shows that the squared ℓ_2-error of the unbiased estimator grows only linearly in $|\Delta|$. As a result, the relative error of the unbiased estimator is decreasing for growing $|\Delta|$.

We now evaluate performance as we vary the threshold λ, as summarized in Figure 1(c). Consistent with the trends in Figures 1(a) and 1(b), we observe roughly an order-of-magnitude improvement from the unbiased estimator, and an additional substantial improvement when using smooth sensitivity. For large λ, the single round baselines outperform *smooth-unbiased*. This occurs because most triangle weights are very small, so even after adding discrete Laplace noise to the edge weights, only a few triangles exceed a large threshold. In such regimes, the two-step algorithms still incur error from the second step, where Laplace noise is added to the below-threshold counts. This introduces a roughly constant error term, which becomes dominant when the true count above the threshold is very small.

In addition to the ML-TELE-400 graph, we have also conducted two more experiments on sparser graphs GMWCS produced by GAM, a web service for integrated transcriptional metabolic network analysis [35, 47], and MUSAE-WIKI, a dataset representing a page-page wikipedia network on articles related to crocodiles [44]. On these graphs, the single-round baselines are preferable in most settings, highlighting the limitations of a two-step approach on smaller graphs. The results are shown in Appendix E.

Discussion. The experiments show a regime-dependent trade-off. The error of the single-round baseline is dominated by the covariance that arises because every triangle estimate uses the noisy weights of all three of its edges, which grows with the number of triangles an edge participates in. The two-step algorithm uses a single noisy weight per triangle, but incurs an additional error from releasing the local counts in Step 2. On ML-TELE-400, every edge lies in many triangles and the first effect dominates, while on GMWCS the second effect dominates.

7 Conclusion

In this work, we investigate the problem of counting below-threshold triangles in weighted graphs under local weight differential privacy, and we introduce a two-step algorithm to address this challenge. We develop both biased and unbiased estimators: the biased estimator incurs error that scales linearly with the number of triangles, whereas the unbiased estimator substantially improves accuracy at the expense of a stronger dependence on the privacy budget. To further improve performance, we incorporate a preprocessing procedure that balances the triangle-counting workload across nodes and can be computed in time linear in the number of triangles. In addition, we present an efficient algorithm for computing smooth sensitivity, running in $O(d^2 \log^2 d)$ time for a node of degree d, enabling more practical noise calibration.

Our experiments show that the two-step approach yields its largest gains on graphs in which edges participate in many triangles, where the covariance of the single-round baseline dominates its error. On very sparse graphs with few triangles per edge, the error added in the second step dominates and the single-round baseline is preferable. An unbiased single-round estimator is competitive for large budgets, but its variance grows much faster as ε decreases, making the two-step unbiased estimator the more robust choice for moderate privacy budgets. These results provide a foundation for accurate subgraph counting in weighted graphs under local differential privacy and suggest natural extensions to more general subgraph statistics.

Acknowledgments

Kevin Pfisterer and Quentin Hillebrand are supported by JST SPRING, Grant Number JPMJSP2108. Vorapong Suppakitpaisarn is supported by JST NEXUS Grant Number Y2024L0906031 and KAKENHI Grant JP25K00369. Quentin Hillebrand is part of BARC, Basic Algorithms Research Copenhagen, supported by the VILLUM Foundation grant 54451. Quentin Hillebrand was also supported by a Data Science Distinguished Investigator grant from Novo Nordisk Fonden.

References

[1] Anders Aamand, Justin Y Chen, Mina Dalirrooyfard, Slobodan Mitrović, Yuriy Nevmyvaka, Sandeep Silwal, and Yinzhan Xu. Differentially private Gomory-Hu trees. In *NeurIPS 2025*, 2025.

[2] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows. 1988.

[3] Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *ITCS 2019*, pages 6:1–6:20, 2019.

[4] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of Milan and the province of Trentino. *Scientific Data*, 2(1):1–15, 2015.

[5] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Annual International Cryptology Conference*, pages 451–468. Springer, 2008.

[6] Louis Betzer, Vorapong Suppakitpaisarn, and Quentin Hillebrand. Publishing number of walks and katz centrality under local differential privacy. In *UAI 2024*, 2024.

[7] Greg Bodwin, Chengyuan Deng, Jie Gao, Gary Hoppenworth, Jalaj Upadhyay, and Chen Wang. The discrepancy of shortest paths. In *ICALP 2024*, pages 27:1–27:20, 2024.

[8] Felipe T Brito, Victor AE Farias, Cheryl Flynn, Subhabrata Majumdar, Javam C Machado, and Divesh Srivastava. Global and local differentially private release of count-weighted graphs. *PACMMOD 2023*, 1(2):1–25, 2023.

[9] Ioannis Caragiannis. Better bounds for online load balancing on unrelated machines. In *SODA 2008*, pages 972–981, 2008.

[10] TH Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *European Symposium on Algorithms*, pages 277–288. Springer, 2012.

[11] Rishi Chandra, Michael Dinitz, Chenglin Fan, and Zongrui Zou. Differentially private algorithms for graph cuts: A shifting mechanism approach and more. In *SODA 2026*, 2026.

[12] Justin Y Chen, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Shyam Narayanan, Jelani Nelson, and Yinzhan Xu. Differentially private all-pairs shortest path distances: Improved algorithms and lower bounds. In *SODA 2023*, pages 5040–5067, 2023.

[13] Michael Dinitz, Chenglin Fan, Jingcheng Liu, Jalaj Upadhyay, and Zongrui Zou. A generalized binary tree mechanism for differentially private approximation of all-pair distances. *arXiv preprint arXiv:2504.03354*, 2025.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC 2006*, pages 265–284, 2006.

[15] Talya Eden, Quanquan C Liu, Sofya Raskhodnikova, and Adam Smith. Triangle counting with local edge differential privacy. *Random Structures & Algorithms*, 66(4):e70002, 2025.

[16] Michael R Garey, David S Johnson, Franco P Preparata, and Robert E Tarjan. Triangulating a simple polygon. *Information Processing Letters*, 7(4):175–179, 1978.

[17] Badih Ghazi, Pritish Kamath, Ravi Kumar, and Pasin Manurangsi. Anonymized histograms in intermediate privacy models. *Advances in Neural Information Processing Systems*, 35:8456–8468, 2022.

[18] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC 2009*, pages 351–360, 2009.

[19] Seifollah Louis Hakimi and Edward F Schmeichel. On the number of cycles of length k in a maximal planar graph. *Journal of Graph Theory*, 3(1):69–86, 1979.

[20] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM 2009*, pages 169–178, 2009.

[21] Quentin Hillebrand, Jacob Imola, Rasmus Pagh, and Sia Sejer. Privacy by postprocessing the discrete laplace mechanism. *arXiv preprint arXiv:2605.06502*, 2026.

[22] Quentin Hillebrand, Vorapong Suppakitpaisarn, and Tetsuo Shibuya. Communication cost reduction for subgraph counting under local differential privacy via hash functions. *Transactions on Machine Learning Research*, 2025.

[23] Quentin Hillebrand, Vorapong Suppakitpaisarn, and Tetsuo Shibuya. Cycle counting under local differential privacy for degeneracy-bounded graphs. In *STACS 2025*, pages 49:1–49:22, 2025.

[24] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. In *USENIX Security 2021*, pages 983–1000, 2021.

[25] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Communication-efficient triangle counting under local differential privacy. In *USENIX Security 2022*, pages 537–554, 2022.

[26] Telecom Italia. Telecommunications - MI to MI, 2015.

[27] Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The role of interactivity in local differential privacy. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 94–105. IEEE, 2019.

[28] Chaogui Kang, Yu Liu, Xiujun Ma, and Lun Wu. Towards estimating urban population distributions from mobile call data. *Journal of Urban Technology*, 19(4):3–21, 2012.

[29] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[30] Silvio Lattanzi and Stefano Leonardi. Efficient computation of the weighted clustering coefficient. *Internet Mathematics*, 12(6):381–401, 2016.

[31] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[32] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[33] Zening Li, Rong-Hua Li, and Fusheng Jin. Triangle counting over signed graphs with differential privacy. In *ICDE 2025*, pages 2094–2106, 2025.

[34] Jingcheng Liu, Jalaj Upadhyay, and Zongrui Zou. Optimal bounds on private graph approximation. In *SODA 2024*, pages 1019–1049, 2024.

[35] Alexander A Loboda, Maxim N Artyomov, and Alexey A Sergushichev. Solving generalized maximum-weight connected subgraph problem for network enrichment analysis. In *WABI 2016*, pages 210–221, 2016.

[36] David W Matula and Leland L Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983.

[37] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[38] Pranay Mundra, Charalampos Papamanthou, Julian Shun, and Quanquan C. Liu. Practical and accurate local edge differentially private graph algorithms. *Proc. VLDB Endow.*, 18(11):4199–4213, 2025.

[39] Mark EJ Newman. Random graphs with clustering. *Physical Review Letters*, 103(5):058701, 2009.

[40] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC 2007*, pages 75–84, 2007.

[41] Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. Intensity and coherence of motifs in weighted complex networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 71(6):065103, 2005.

[42] Jiayu Qin, Gang Mei, and Lei Xiao. Building the traffic flow network with taxi gps trajectories and its application to identify urban congestion areas for traffic planning. *Sustainability*, 13(1):266, 2020.

[43] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *CCS 2017*, pages 425–438, 2017.

[44] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[45] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. Generalizations of the clustering coefficient to weighted complex networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 75(2):027105, 2007.

[46] Adam Sealfon. Shortest paths and distances with differential privacy. In *PODS 2016*, pages 29–41, 2016.

[47] Alexey A Sergushichev, Alexander A Loboda, Abhishek K Jha, Emma E Vincent, Edward M Driggers, Russell G Jones, Edward J Pearce, and Maxim N Artyomov. GAM: A web-service for integrated transcriptional and metabolic network analysis. *Nucleic acids research*, 44(W1):W194–W200, 2016.

[48] Vorapong Suppakitpaisarn, Donlapark Ponnoprat, Nicha Hirankarn, and Quentin Hillebrand. Counting graphlets of size k under local differential privacy. In *AISTATS 2025*, pages 5005–5013, 2025.

[49] Wen Xu, Pengpeng Qiao, Shang Liu, Zhirun Zheng, Yang Cao, and Zhetao Li. Continuous publication of weighted graphs with local differential privacy. *Proc. VLDB Endow.*, 18(11):4214–4226, 2025.

[50] Akito Yamamoto and Tetsuo Shibuya. Differentially private selection using smooth sensitivity. In *IPCCC 2024*, pages 1–10. IEEE, 2024.

[51] Jiani Zhu, Xi Lin, Yuxin Qi, Gaolei Li, Chaofeng Zhang, and Jianhua Li. Local differential private spatio-temporal dynamic graph learning for wireless social networks. In *WCNC 2024*, pages 1–6. IEEE, 2024.

[52] Zongrui Zou, Jingcheng Liu, and Jalaj Upadhyay. Almost linear time differentially private release of synthetic graphs. In *AISTATS 2025*, pages 289–297, 2025.

A Analysis of two-step algorithm

In the following we provide the missing proofs for statements made regarding the analysis of Algorithm 1 of Section 3.

Proof for Theorem 1. By Definition 5, we need to find $w \sim w'$ such that $|f'_v(w) - f'_v(w')|$ is maximized. Since w and w' are neighboring, there exists exactly one incident edge to v whose weight differs by 1 between w and w'. Let $\{v, u\}$ denote this edge. Then, a change of w_{vu} affects all triangles that contain $\{v, u\}$ and v is responsible for counting: $S = \{T \in \Delta_v : \{v, u\} \in T\}$. In the worst case for each triangle T in S, the estimator changes by $GS(g_v^T)$ where $GS(g_v^T)$ is the maximum change of the respective estimator. It is straightforward to show $GS(g_v^T)$ for the biased estimator and unbiased estimator. □

Proof of Theorem 3. The communication incurred by each node consists of three components: (i) uploading the noisy incident-weight vector, (ii) downloading the noisy weights of its assigned triangles, and (iii) uploading its local triangle-count statistics.

Let W denote the maximum edge weight in the graph. Using tail bounds for the discrete and continuous Laplace distributions together with a union bound over all released noise variables, we obtain the following results.

The total communication cost for uploading the noisy weight vectors is bounded by

$$O\left(m \log W + m \log\left(\frac{1}{\varepsilon_1} \log\left(\frac{m}{\gamma}\right)\right)\right)$$

with probability at least $1-\gamma$. During redistribution, only weights corresponding to assigned triangles are transmitted. Since each triangle is assigned to a single node, at most one noisy weight is sent per triangle. The total cost for downloading noisy triangle weights is therefore bounded by

$$O\left(|\Delta| \log W + |\Delta| \log\left(\frac{1}{\varepsilon_1} \log\left(\frac{m}{\gamma}\right)\right)\right)$$

with probability at least $1-\gamma$. Finally, publishing the noisy local triangle counts incurs a communication cost of

$$O\left(|\Delta|\left(\log\left(\frac{1}{\varepsilon_2} \log \frac{|\Delta|}{\gamma}\right) + 1\right)\right)$$

with probability at least $1-\gamma$. We obtain the lemma statement by setting $\gamma = 1/(3m)$. □

A.1 Precision of the biased estimator

We now provide the analysis of the expected squared ℓ_2-error of Algorithm 1 under the biased estimator B'_T proving Theorem 4. Randomness arises in two places: (i) Step 1, where incident weight vectors are released via the discrete Laplace query, and (ii) Step 2, where local below-threshold counts are released via the Laplace query.

Lemma 5. *For a triangle $T \in \Delta$ with triangle weight w_T the expected value of the estimator B'_T is given by*

$$\mathbb{E}[B'_T] = \begin{cases} 1 - \frac{p^{\lambda - w_T}}{1+p} & \text{if } w_T < \lambda \\ \frac{p^{w_T - \lambda + 1}}{1+p} & \text{if } w_T \geq \lambda \end{cases}$$

Proof. Let $T \in \Delta$ with triangle weight w_T. Denote $w_{vu} + w_{vx} + w'_{ux}$ by w'_T. Notice that $w'_T = w_{vu} + w_{vx} + w_{ux} + Z = w_T + Z$ where $Z \sim DLap(p)$. When $w_T < \lambda$,

$$\begin{aligned} \mathbb{E}[B'_T] = \Pr[Z < \lambda - w_T] &= \frac{1-p}{1+p} \sum_{i=-\infty}^{\lambda - w_T - 1} p^{|i|} \\ &= \frac{1-p}{1+p}\left(\sum_{i=0}^{\infty} p^i - 1 + \sum_{i=0}^{\lambda - w_T - 1} p^i\right). \end{aligned}$$

Since $p = e^{-\varepsilon_1}$, it holds that $|p| < 1$ and we can apply the closed form solution for the geometric series.

$$\Pr[Z < \lambda - w_T] = 1 - \frac{p^{\lambda - w_T}}{1+p}$$

When $w_T \geq \lambda$, we apply the same techniques to obtain that

$$\mathbb{E}[B'_T] = \Pr[Z < \lambda - w_T] = \frac{p^{w_T - \lambda + 1}}{1+p}.$$

□

We can infer from Lemma 5 that the expected value for the triangle $T \in \Delta$ is independent from the node that is responsible for counting it.

Lemma 6. *The bias for a triangle $T \in \Delta$ is given by*

$$\mathbb{E}[B'_T] - 1\{w_T < \lambda\} = \begin{cases} -\frac{p^{\lambda - w_T}}{1+p} & \text{if } w_T < \lambda \\ \frac{p^{w_T - \lambda + 1}}{1+p} & \text{if } w_T \geq \lambda \end{cases}$$

Proof. The statement directly follows from Lemma 5. □

We derive the following two corollaries from Lemma 6.

Corollary 1. *The bias for a triangle $T \in \Delta$ is maximized for triangle weights at the boundary cases: $w_T = \lambda$, $w_T = \lambda - 1$; This results in the following bounds for the bias of the estimator.*

$$-\frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}} \leq \mathbb{E}[B'_T] - 1\{w_T < \lambda\} \leq \frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}}.$$

Corollary 2. *For the triangle $T \in \Delta$, the absolute bias decreases exponentially in $|\lambda - w_T|$:*

$$|\mathbb{E}[B'_T] - 1\{w_T < \lambda\}| \leq e^{-\varepsilon_1 |\lambda - w_T|}.$$

Recall that the output of Algorithm 1 is $\tilde{k}$ and let $f(G)$ be the actual count of below-threshold triangles. We decompose the expected squared ℓ_2-error into the squared bias plus the variance:

$$(\mathbb{E}[\tilde{k}] - f(G))^2 + \mathrm{Var}[f'(G, \varepsilon_1, \varepsilon_2)]$$

Lemma 7. *The squared bias of Algorithm 1 with biased estimator B'_T is bounded by*

$$|\Delta|^2 \left(\frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}}\right)^2$$

Proof. The expectation is taken over the randomness used when releasing the weight vectors and when releasing the local below-threshold counts.

$$\mathbb{E}[\tilde{k}] = \sum_{v \in V} \mathbb{E}[f'_v(w^v) + Lap(GS(f'_v)/\varepsilon_2)]$$

The Laplace query is unbiased. Hence, the bias of the algorithm arises solely from f'_v.

$$\mathbb{E}[\tilde{k}] - f(G) = \sum_{T \in \Delta} (\mathbb{E}[B'_T] - 1\{w_T < \lambda\})$$

From Corollary 1 the bound on the squared bias follows. □

We observe from the previous lemma that the bias scales the true value by a factor of $\frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}} < 1$. In other words, the relative error introduced by this biased estimator is this constant factor.

We now analyze the variance of $\tilde{k}$. Figure 2 visualizes such a C'_4 instance as introduced in Section 3.4. We use the C'_4 definition to separate the total variance $\mathrm{Var}[\sum_{v \in V} f'_v(w^v)]$ as follows:

$$\sum_{v \in V} \mathrm{Var}[f'_v(w^v)] + \sum_{x,y \in V} \sum_{v,u \in N_{x,y}} \mathrm{Cov}[f'_v(w^v), f'_u(w^u)].$$

We first analyze the variance of the estimated local below-threshold count.

Lemma 8. *For a node $v \in V$, the variance $\mathrm{Var}[f'_v(G)]$ is bounded by*

$$\frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}} \left(1 - \frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}}\right) |\Delta_v| = O\left(\frac{|\Delta_v|}{\varepsilon_1}\right)$$

Proof. For every triangle $T \in \Delta_v$ the noise used in the estimator B'_T is independent from the noise in the other triangles in Δ_v. We obtain that:

$$\mathrm{Var}[f'_v(w^v)] = \sum_{T \in \Delta_v} \mathrm{Var}[B'_T]$$

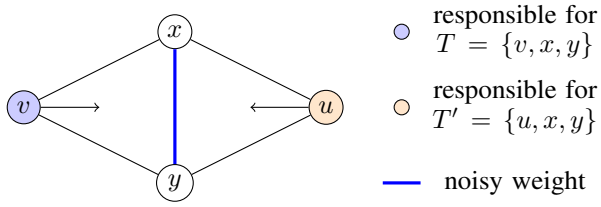


Figure 2: Visualization of C_4' instance introducing covariance

From the fact that B'_T is an indicator variable and from the probabilities shown in Lemma 5, for $w_T < \lambda$, we obtain that

$$\mathrm{Var}[B'_T] = \left(1 - \frac{p^{\lambda - w_T}}{1+p}\right) \frac{p^{\lambda - w_T}}{1+p}.$$

For $w_T \geq \lambda$, we obtain that

$$\mathrm{Var}[B'_T] = \left(1 - \frac{p^{w_T - \lambda + 1}}{1+p}\right) \frac{p^{w_T - \lambda + 1}}{1+p}.$$

Similar to the bias, the variance is maximized in the boundary cases, which proves the claim. □

Lastly, we bound the covariance between B'_T and $B'_{T'}$, which rely on the same noisy weight information.

Lemma 9. *For the triangles $T \in \Delta_v$ and $T' \in \Delta_{v'}$ that use the same noisy weight information the covariance of the estimators B'_T and $B'_{T'}$ is bounded by $2\frac{e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}}$.*

Proof. Let $T, T' \in \Delta$ denote triangles that share the same noisy weight information, $B'_T, B'_{T'}$ be the respective biased estimators and $Z \sim DLap(p)$ be the noise added to the weight information.

Both estimators are 1 at the same time when $\max(w_T, w_{T'}) + Z < \lambda$. This results in the following formula for the covariance:

$$\Pr[Z < \lambda - \max(w_T, w_{T'})] - \Pr[Z < \lambda - w_T] \Pr[Z < \lambda - w_{T'}].$$

We make a case distinction on the values of w_T and $w_{T'}$ and use the probabilities from Lemma 5 to bound the covariance in each case.

Case $w_T < \lambda$ and $w_{T'} < \lambda$: Using the same techniques as in Lemma 5 gives:

$$\Pr[Z < \lambda - \max(w_T, w_{T'})] = 1 - \frac{p^{\lambda - \max(w_T, w_{T'})}}{1+p},$$

which results in a covariance of

$$\frac{1}{1+p} \cdot \left(p^{\lambda - w_T} + p^{\lambda - w_{T'}} - p^{\lambda - \max(w_T, w_{T'})} - \frac{p^{2\lambda - w_T - w_{T'}}}{1+p}\right).$$

This covariance is maximized when $w_T = \lambda - 1$ and $w_{T'} = \lambda - 1$. In this case, the covariance of B'_T and $B'_{T'}$ is not larger than $\frac{2p}{1+p}$.

For the remaining cases, we provide only the resulting covariance and the corresponding upper bound obtained at the boundary of each case.

Case $w_T < \lambda$ and $w_{T'} \geq \lambda$: $\frac{p^{w_{T'} - w_T + 1}}{(1+p)^2} \leq \frac{p}{1+p}$.

Case $w_T \geq \lambda$ and $w_{T'} < \lambda$: $\frac{p^{w_T - w_{T'} + 1}}{(1+p)^2} \leq \frac{p}{1+p}$.

Case $w_T \geq \lambda$ and $w_{T'} \geq \lambda$:

$$1/(1+p)\left(p^{\max(w_T, w_{T'}) - \lambda + 1} - \frac{p^{w_T + w_{T'} - 2\lambda + 2}}{1+p}\right) \leq \frac{p}{1+p}.$$

□

This lets us bound the total variance of the biased estimators.

Lemma 10. *Let $\#C_4'$ denote the number of C_4' instances in our assignment ρ. The variance* $\mathrm{Var}\left[\sum_{v\in V} f_v'(w^v)\right]$ *is bounded by* $O\left(\frac{|\Delta|}{\varepsilon_1} + \frac{\#C_4'}{\varepsilon_1}\right)$.

Proof. The result follows from the bounds introduced in Lemmas 8 and 9.

$$\sum_{v\in V} \mathrm{Var}[f_v'(w^v)] + \sum_{u,x\in V} \sum_{v,v'\in N_{u,x}} \mathrm{Cov}[f_v'(w^v), f_{v'}'(w^{v'})]$$
$$\leq O\left(\frac{1}{\varepsilon_1}\right) \sum_{v\in V} |\Delta_v| + \#C_4' \frac{2e^{-\varepsilon_1}}{1+e^{-\varepsilon_1}} = O\left(\frac{|\Delta|}{\varepsilon_1} + \frac{\#C_4'}{\varepsilon_1}\right).$$

□

Lastly, we bound the variance introduced by publishing the local triangle counts in the final step.

Lemma 11. *The variance of the Laplace query in the second step is bounded by* $O\left(\frac{d_{max} m}{\varepsilon_2^2}\right)$.

Proof. The variances of publishing the local counts are independent from each other and the variance of the Laplace query $Lap(b)$ is equal to $2b^2$. Hence, the Laplace query for publishing the local counts in Step 2 is bounded by $2\sum_{v\in V} \frac{GS(f_v')^2}{\varepsilon_2^2} \leq 4\frac{d_{max}}{\varepsilon_2^2} m$. The inequality follows from the fact that a single incident edge of v can be in at most $d_v - 1$ triangles. The sensitivity $GS(f_v')$ is therefore bounded by the degree d_v. This then also results in $\sum_{v\in V} GS(f_v') \leq \sum_{v\in V} d_v = 2m$. □

We are now ready to prove the bound on the expected squared ℓ_2-error of Theorem 4.

Proof of Theorem 4. This follows from Lemmas 7, 10 and 11. □

We note that in the high-privacy regime ($\varepsilon \to 0$) the terms depending on ε_1 in the covariance and variance are bounded by some constant. As explained in Section 3.4, the lower dependency on the privacy budget is the main advantage of the biased variant compared to the unbiased one.

A.2 Precision of the unbiased estimator

In this section, we analyze Algorithm 1 when the unbiased estimator U_T' is employed at Line 7. The analysis follows the steps of the previous section.

Lemma 12. *For a triangle $T \in \Delta$, the estimator U_T' is unbiased, i.e.* $\mathbb{E}[U_T'] = 1\{w_T < \lambda\}$.

Proof. The claim follows from Lemma 10 in [17]. Let f be the function defined in that lemma. Then,

$$\mathbb{E}[h(w_T')] = 1 - \mathbb{E}[f(w_T' - \lambda)] = 1 - [w_T \geq \lambda] = \mathbf{1}[w_T < \lambda].$$

We also note that the same proof applies for all $w_T, \lambda \in \mathbb{Z}$, and is not limited to $\mathbb{N} \cup \{0\}$ as originally stated in [17]. □

Hence, the expected squared ℓ_2-error consists only of the variance introduced by the estimator and the variance of the general Laplace query for publishing the local below-threshold counts.

Lemma 13. *For a triangle $T \in \Delta$, the variance of the estimator U_T' is given by* $4p^{|w_T - \lambda|+1}\left(\frac{p}{(1-p)^3} + 1 - p\right)$.

Proof. This follows from Theorem 10 in [17]. □

Lemma 14. *For a node $v \in V$, the variance* $\mathrm{Var}[f_v'(w^v)]$ *is bounded by* $O\left(\frac{|\Delta_v|}{\varepsilon_1^3}\right)$.

Proof. The variance of an individual estimator U'_T with $T \in \Delta_v$ is maximized in the boundary case $|w_T - \lambda| = 0$. Moreover, each triangle counted by v uses an independent noisy weight, so there is no covariance between different estimators. Hence,

$$\mathrm{Var}[f'_v(w^v)] \leq |\Delta_v| 4p \left(\frac{p}{(1-p)^3} + 1 - p \right) = O\left(\frac{|\Delta_v|}{\varepsilon_1^3} \right).$$

□

As discussed in Appendix A.1, any two estimators U'_T and $U'_{T'}$ that rely on the same noisy weight share a nonzero covariance. To bound the covariance contribution of each C'_4 instance, we apply the Cauchy–Schwarz inequality.

Lemma 15. *For two estimators $U'_T, U'_{T'}$ that use the same noisy weight information, the covariance is bounded by*

$$4p \left(\frac{p}{(1-p)^3} + 1 - p \right) = O\left(\frac{1}{\varepsilon_1^3} \right).$$

Proof. We bound the covariance of the estimators U'_T and $U'_{T'}$ using the Cauchy–Schwarz inequality:

$$4 \left(\frac{p}{(1-p)^3} + (1-p) \right) \sqrt{p^{\,|w_T - \lambda| + |w_{T'} - \lambda| + 2}}.$$

This bound is maximized when both triangle weights are exactly at the threshold, i.e., $w_T = \lambda$ and $w_{T'} = \lambda$. □

We then obtain the variance introduced by the unbiased estimator in the following theorem.

Lemma 16. *The variance introduced by using the unbiased estimator with noisy edge weights is bounded by $O\left(\frac{|\Delta|}{\varepsilon_1^3} + \frac{\#C'_4}{\varepsilon_1^3} \right)$*

Proof. This follows from Lemmas 14 and 15. □

Before presenting the final theorem on the expected ℓ_2-error of the algorithm with the unbiased estimator, we first bound the variance arising from releasing the local counts in Step 2. We note that the unbiased estimator depends on the privacy budget ε_1. As a consequence, both its sensitivity and the parameter of the Laplace query used in Step 2 also depend on ε_1.

Lemma 17. *The variance of the Laplace query in the second step of algorithm 1 with unbiased estimator is $O\left(\frac{d_{max}}{\varepsilon_1^4 \varepsilon_2^2} m \right)$.*

Proof. We obtain that

$$2 \sum_{v \in V} \frac{GS(f'_v)^2}{\varepsilon_2^2} \leq \frac{4}{\varepsilon_2^2} \left(1 + 2 \frac{p}{(1-p)^2} \right)^2 d_{max} m = O\left(\frac{d_{max}}{\varepsilon_1^4 \varepsilon_2^2} m \right).$$

□

Proof of Theorem 5. This follows from Lemmas 16 and 17. □

A.3 Variance of smooth sensitivity query

We now show that Theorem 4 and 5 also hold when the smooth-sensitivity query is used in Line 9 of Algorithm 1.

Lemma 18. *If Line 9 uses the smooth-sensitivity query with $\Gamma = 4$, the variance of the Step 2 release is bounded by $O(d_{\max} m / \varepsilon_2^2)$ for the biased estimator and by $O(d_{\max} m / (\varepsilon_1^4 \varepsilon_2^2))$ for the unbiased estimator. The bias is unchanged.*

Proof. Let $c = 2(\Gamma-1)^{(\Gamma-1)/\Gamma} = 2 \cdot 3^{3/4}$. Then $\tilde{f}_v = f'_v + c\, S^*_{f,\varepsilon_2/2(\Gamma-1)}(w^v) Z_v/\varepsilon_2$, where $S^*_{f,\varepsilon_2/2(\Gamma-1)}(w^v)$ is the smooth sensitivity computed by v and Z_v is drawn independently of all other randomness, with $\mathbb{E}[Z_v] = 0$ and $\mathrm{Var}[Z_v] = 1$. Hence the noise term has mean zero, all cross terms with f'_v and between different nodes vanish, and $\mathrm{Var}[cS^*_{f,\varepsilon_2/2(\Gamma-1)}(w^v)Z_v/\varepsilon_2] = c^2\,\mathbb{E}[(S^*_{f,\varepsilon_2/2(\Gamma-1)}(w^v)^2]/\varepsilon_2^2$. Since $LS_{f'_v}(y) \le GS(f'_v)$ for every y and $e^{-\beta d(w,y)} \le 1$, we have $S^*_{f,\varepsilon_2/2(\Gamma-1)}(w^v) \le GS(f'_v)$ deterministically. Summing over v and bounding $\sum_v GS(f'_v)^2$ as in Lemmas 11 and 17 gives the claim. □

The proof of Theorem 4 and 5 for the smooth-sensitivity query then follows from Lemma 18 and the Lemmas 10 and 16.

B Assignment function

In this section we provide the algorithm for computing the optimal assignment and the proofs for the lemmas and theorems of Section 4.

B.1 Computation of optimal assignment

Given a graph $G = (V, E)$ with the set of triangles Δ, we construct a min-cost max-flow network to compute the optimal assignment function. In the following, $N = (V_N, E_N)$ denotes the min-cost max-flow network.

B.1.1 Network construction

In addition to a source s and target t, we introduce a node v_T for every triangle $T \in \Delta$ and a node v_e for edge $e \in E$:

$$V_N = \{s, t\} \cup \{v_T : T \in \Delta\} \cup \{v_e : e \in E\}$$

For an edge $(u, v) \in E_N$, $a(u, v) > 0$ denotes the capacity and $b(u, v)$ denotes the cost for sending a unit of flow along the edge. We represent the edge including the capacity and cost directly as a tuple of the form $(u, v, a(u, v), b(u, v))$. We first introduce edges linking the source s to the triangle nodes v_T with capacity 1 and costs 0.

$$\{(s, v_T, 1, 0) : \forall T \in \Delta\}$$

Secondly, a triangle node v_T is connected to all edge nodes v_e where e is contained in the triangle T with capacity 1 and cost 0.

$$\{(v_T, v_e, 1, 0) : \forall e \in T, T \in \Delta\}$$

Let $d_{in}(v_e)$ denote the incoming degree of an edge node v_e in N. This represents the number of triangles the edge e is contained in. We now add $d_{in}(v_e)$ edges from the edge node v_e to the target t with capacity 1 and costs in $\{0, ..., d_{in}(v_e) - 1\}$.

$$\{(v_e, t, 1, b) : b \in \{0, ..., d_{in}(v_e) - 1\}, \forall e \in E\}$$

B.1.2 Assignment extraction

We now describe how to extract the assignment ρ alongside the costs of the assignment $c(\tau)$ for a max-flow min-cost solution of the network N. Let $f(u, v)$ denote the flow of an edge $(u, v) \in E_N$ in the network. For a triangle $T = \{u, v, w\} \in \Delta$, let $(v_T, v_e) \in E_N$ be a network edge with flow one: $f(v_T, v_e) = 1$. We then assign T to the node that is not included in the edge $e \in E$. For $e = \{u, v\}$, this results in $\rho(T) = w$.

We first remind the reader of the min-cost max-flow integrality property.

Lemma 19 ([2])**.** *Any minimum cost network flow problem instance whose demands, supplies and capacities are all integers has an optimal solution with integer flow on each edge. Such an optimal solution can be found in $O\big(F \cdot (|E_N| + |V_N| \log |V_N|)\big)$ time for an input network with $|V_N|$ nodes, $|E_N|$ edges, and maximum flow value F.*

We observe that, in the flow obtained in Lemma 19, for each $T \in \Delta$, exactly one edge in $\{(v_T, v_e, 1, 0) : e \in T\}$ has flow value one. We define the assignment ρ_f by setting $\rho_f(T) = e$ whenever the edge $(v_T, v_e, 1, 0)$ carries one unit of flow. In what follows, we assume that f is an optimal integer-flow solution.

Lemma 20. *If the flow solution f has cost C, then $c(\rho_f) = C$.*

Proof. We first show that for each triangle $T = \{e_1, e_2, e_3\} \in \Delta$ there exist exactly one edge $(v_T, v_{e_i}) \in E_N$ with $i \in \{1, 2, 3\}$ and flow 1. Given that the max-flow value is $|\Delta|$ and each source edge (s, v_T) has capacity 1, every such edge (s, v_T) is saturated with flow 1. By flow conservation at v_T and flow integrality, exactly one outgoing edge (v_T, v_{e_i}) has a flow of 1.

We now prove that the costs of the flow solution is equivalent to the assignment costs. The network has only costs on the edges of the form (v_e, t). Let v_e be an edge node with incoming flow $f_{in}(v_e)$. The costs are minimized by satisfying the outgoing edges in increasing order of costs. The total costs by flow traveling through v_e is then

$$\sum_{i=0}^{f_{in}(v_e)-1} i = \binom{f_{in}(v_e)}{2}.$$

Since the edges from v_e to t and $v_{e'}$ to t are distinct, the total costs is given by $\sum_{e \in E} \binom{f_{in}(v_e)}{2}$. An edge $e \in E$ is assigned to T if $f(v_T, v_e) = 1$, so $f_{in}(v_e)$ represents exactly the load $l(e)$. □

Lemma 21. *Let ρ denote an assignment for graph G with cost C, then there exists a min-cost max-flow solution f for the network N with flow $|\Delta|$ and costs C.*

Proof. Let ρ be an assignment for the graph G with cost C. We show how to construct a solution f. Firstly, for each source-triangle edge (s, v_T) set the flow to 1. For $T = \{u, v, w\}$ with $\rho(T) = v$, set $f(v_T, v_{\{u,w\}}) = 1$. Let $f_{in}(v_e)$ denote the incoming flow of v_e, choose $f_{in}(v_e)$ outgoing edges to t in order of increasing costs.

We first show that the constructed solution is valid. The capacity constraint is satisfied since the set flow is bounded by 1 and the capacity is 1 for every edge in N. Flow conservation is given on triangle nodes since each has incoming flow of 1 and also outgoing flow of 1 by the fact that each triangle is assigned to exactly 1 edge in the assignment ρ. For edge nodes $v_e \in V_N$ flow conservation is satisfied by the construction of the solution. Lastly, the outgoing flow of s is $|\Delta|$ and the incoming flow of t is also $|\Delta|$. The cost of the flow is then given by

$$\sum_{e \in E} \binom{f_{in}(v_e)}{2} = \sum_{e \in E_G} \binom{l(e)}{2}$$

□

Theorem 11. *Let f denote the optimal min-cost max-flow solution to the network N, then the extracted assignment ρ_f is an optimal assignment.*

Proof. Follows from Lemma 20 and Lemma 21. □

B.1.3 Running time analysis

The constructed network allows us to apply any min-cost max-flow algorithm to compute the optimal flow solution. In the following, we show the size of the flow-network N which provides the bounds on the running time. We assume that the network was converted to a simple graph without multi-edges by introducing stub nodes.

Lemma 22. *The number of nodes $|V_N|$ is $2 + 4|\Delta| + |E|$.*

Proof. The number of nodes without stubs follows directly from the definition.

$$|\{s, t\} \cup \{v_T : T \in \Delta\} \cup \{v_e : e \in E\}| = 2 + |\Delta| + |E|$$

A stub is generated for the multi-edges (v_e, t). By the construction $d_{in}(v_e)$ multi-edges between v_e and t exist. Since the total number of incoming edges on the edge-node level is equal to the number of outgoing edges on the triangle-node level and each triangle-node v_T has 3 outgoing edges this results in the following total amount for the number of stub nodes: $\sum_{e \in E} d_{in}(v_e) = 3|\Delta|$ □

Lemma 23. *The number of edges $|E_N|$ is $10|\Delta|$.*

Proof. There exist $|\Delta|$ edges between the source s and the triangle-nodes, $3|\Delta|$ edges between the triangle-nodes and edge-nodes, and $6|\Delta|$ edges between edge-nodes and the target. The latter follows from the fact that there exist $3|\Delta|$ stubs and each is incident to exactly two edges. □

We are now able to provide the proof of Theorem 6.

Proof of Theorem 6. To compute the min-cost max-flow, we utilize the Successive Shortest Path algorithm. For a network with a known maximum flow F, node set V_N, and edge set E_N, each augmentation step requires a shortest-path computation which takes $O(|E_N| + |V_N| \log |V_N|)$ time. The total running time is therefore bounded by $O(F \cdot (|E_N| + |V_N| \log |V_N|))$.

From Lemma 22 and Lemma 23, the network size is bounded by $|V_N| = O(|\Delta| + |E|)$ and $|E_N| = O(|\Delta|)$. By construction, the maximum flow is exactly $F = |\Delta|$. Substituting these values into the algorithmic complexity, we obtain:

$$O(|\Delta| \cdot (|\Delta| + (|\Delta| + |E|) \log(|\Delta| + |E|)))$$

Expanding this expression yields the final theoretical bound on the running time:

$$O(|\Delta|^2 + |\Delta|(|\Delta| + |E|) \log(|\Delta| + |E|))$$

□

B.2 Greedy algorithm for the computation of the optimal assignment

Algorithm 2 presents the greedy approach described in Section 4, whose running time is linear in $|\Delta|$.

Algorithm 2 GREEDY ALGORITHM FOR COMPUTING ASSIGNMENT ρ

```
Initialize l(e) = 0   ∀e ∈ E
for T ∈ Δ do
    e = arg min_{e∈T} l(e)
    Let v be the only node in T/e. Assign ρ(T) = v
    l(e) = l(e) + 1
end for
```

We restate the optimization problem introduced in Section 4. As previously shown, solving the optimization problem is equivalent to finding the assignment with minimal costs.

For a triangle $T \in \Delta$ and edge $e \in T$, let $x_{T,e} \in \{0, 1\}$ denote if T is assigned to e.

$$\begin{aligned} \min \quad & \sum_{e \in E} \Big(\sum_{T \in \Delta} x_{T,e} \Big)^2 & & \\ \text{s.t.} \quad & \sum_{e \in T} x_{T,e} = 1 & & \forall T \in \Delta \\ & x_{T,e} \in \{0, 1\} & & \forall T \in \Delta, e \in T \end{aligned} \tag{1}$$

Lemma 24. *The optimization problem (1) reduces to minimizing the ℓ_2 norm in the classic offline load-balancing problem, where unit-size jobs must be assigned to machines under restricted assignment constraints.*

Proof. Let a triangle $T \in \Delta$ represent job J_T and let edge $e \in E$ represent machines M_e. A job J_T can then be scheduled to exactly one of the three machines $M_{e_i}, M_{e_j}, M_{e_k}$ such that $e_i, e_j, e_k \in T$. Each job increases the load on its assigned machine by 1. The squared ℓ_2-norm of the machine loads represents the objective value in problem (1). □

Lemma 25. *Algorithm 2 has an approximation ratio of $3 + 2\sqrt{2}$ for the optimization problem (1).*

Proof. In [9], it was shown that the greedy algorithm for the offline load-balancing problem attains a $1+\sqrt{2}$ approximation ratio for the ℓ_2 norm. This implies an approximation ratio of $3+2\sqrt{2}$ for the squared ℓ_2 norm. □

We are now able to prove the approximation ratio as stated in Theorem 7.

Proof of Theorem 7. Let $l(e)$ denote the loads in the greedy assignment, and $l^*(e)$ denote the loads in the optimal assignment. From the definition of the assignment function, the number of triangles is conserved, meaning $\sum_{e\in E} l(e) = \sum_{e\in E} l^*(e) = |\Delta|$.

The cost of the optimal assignment is: $\#C'_{4,OPT} = \frac{1}{2}\sum_{e\in E}(l^*(e))^2 - \frac{1}{2}|\Delta|$. Rearranging this equation to isolate the sum of squared loads gives:

$$\sum_{e\in E}(l^*(e))^2 = 2\#C'_{4,OPT} + |\Delta|$$

Similarly, for the greedy algorithm, we have:

$$\sum_{e\in E}(l(e))^2 = 2\#C'_{4,ALG} + |\Delta|$$

By Lemma 25, we know the greedy algorithm achieves a $3+2\sqrt{2}$ approximation for the sum of squared loads:

$$\sum_{e\in E}(l(e))^2 \leq (3+2\sqrt{2})\sum_{e\in E}(l^*(e))^2$$

Substituting the sum of squared loads and rearranging the inequality yields the statement. □

B.3 Bounds for planar graphs

We now present the proofs for the upper and lower bounds of the precision on planar graphs of Section 4. We denote the number of nodes by n and the number of edges by m in this section. We first state the required definitions for this section.

Definition 12 (Planar graph)**.** *A graph $G = (V, E)$ is* planar *if it admits a drawing in the plane such that edges intersect only at their common endpoints.*

Definition 13 (Maximal planar graph)**.** *A planar graph is* maximal planar *(or a* triangulation*) if no edge can be added without violating planarity, or equivalently, if every face of any planar embedding, including the outer face, is a triangle.*

Lemma 26. *For any maximal planar graph G with $|\Delta| \geq 1$, $|\Delta| \leq m - 2$.*

Proof. From [19], it is known that any maximal planar graph with $n \geq 4$ has $|\Delta| \leq 3n-8$. Moreover, the number of edges is $m = 3n-6$ in maximal planar graphs. This shows the $|\Delta| \leq m-2$ statement for $n \geq 4$. For $n = 3$, there exists only the graph consisting of a single triangle for which the statement holds trivially. □

We now generalize Lemma 26 for general planar graphs.

Lemma 27. *For any planar graph with $|\Delta| \geq 1$, $|\Delta| \leq m - 2$.*

Proof. Let $G = (V, E)$ be a planar graph with $|\Delta| \geq 1$. If the graph is maximal planar, then the statement follows from Lemma 26.

In the following, assume that G is not maximal planar. We generate a maximal planar graph $G' = (V, E')$ by adding edges to G without violating planarity.

Since G is not maximal planar, there exists at least one face f that is bounded by $k \geq 4$ vertices. Treating the boundary of f as a simple polygon, we can triangulate its interior. As proven by [16], any simple polygon of k vertices can be fully partitioned into exactly $k-2$ triangles by adding exactly $k-3$ non-intersecting internal chords.

Let $\mathsf{c_f}$ denote the number of edges (chords) added to triangulate face f, and $\mathsf{d_f}$ denote the number of new triangles created. Based on [16]: $\mathsf{c_f} = k - 3$ and $\mathsf{d_f} \geq k - 2$. We note that additional triangles might be generated outside the face resulting an inequality for $\mathsf{d_f}$. It is guaranteed that for every face triangulated, the number of new triangles generated is strictly greater than the number of edges added, $\mathsf{d_f} > \mathsf{c_f}$.

We repeat this triangulation process for all non-triangular faces until the resulting graph G' is maximal planar. Let m', Δ' denote the number of edges and the set of triangles after the process, and let c denote the total number of edges introduced during the entire procedure ($\mathsf{c} = m' - m$), and let d denote the total number of new triangles introduced ($\mathsf{d} = |\Delta'| - |\Delta|$).

Because G was not maximal planar, we must have triangulated at least one face, meaning $\mathsf{d_f} > \mathsf{c_f}$ occurred at least once. Therefore, summing the newly added edges and triangles across the entire graph yields: $\mathsf{d} \geq \mathsf{c} + 1$.

Since the resulting graph G' is maximal planar, we apply Lemma 26 to get the following inequality:

$$|\Delta'| \leq m' - 2$$

Substituting $|\Delta'| = |\Delta| + \mathsf{d}$ and $m' = m + \mathsf{c}$ into the inequality yields:

$$|\Delta| + \mathsf{d} \leq m + \mathsf{c} - 2$$

Rearranging the terms to isolate $|\Delta|$ gives:

$$|\Delta| \leq m - 2 - (\mathsf{d} - \mathsf{c})$$

Because we have established that $\mathsf{d} - \mathsf{c} \geq 0$, the term $-(\mathsf{d} - \mathsf{c})$ is less than or equal to zero. Therefore, $|\Delta| \leq m - 2$. □

We are now ready to prove the existence of an assignment ρ with $c(\rho) = 0$ for any planar graph.

Proof of Lemma 1. Given a planar graph $G = (V, E)$, we construct the bipartite graph $H = (A \cup B, E_H)$ with $A = \{v_T : T \in \Delta\}, B = \{v_e : e \in E\}$, and $E_H = \{\{v_T, v_e\} : \forall e \in T, T \in \Delta\}$. Let $\mathcal{N}(v_T) = \{v_e : \{v_T, v_e\} \in E_H\}$ denote the neighborhood of a triangle node $v_T \in A$. Let $\mathcal{N}(S) = \bigcup_{v_T \in S} \mathcal{N}(v_T)$ be the neighborhood of the set $S \subseteq A$. By Hall's theorem there exists a perfect matching on the triangles if for every subset $S \subseteq A$, $|S| \leq |\mathcal{N}(S)|$. Let $G(S) = (V, E_S)$ be the subgraph restricted to edges contained in a triangle in S:

$$E_S = \{e \in E : \exists v_T \in S \; s.t. \; e \in T\}$$

Let $X_1, ..., X_j$ denote the connected components of $G(S)$ and let $S(X_i)$ denote the set of triangles in X_i. By construction every triangle in S is in exactly one connected component X_i: $\sum_{X_i} |S(X_i)| = |S|$. Moreover, each connected component contains at least one triangle: $|S(X_i)| \geq 1$.

We can now apply Lemma 27 to each component: $|S(X_i)| \leq |E_{X_i}| - 2$. Summing over all connected components, we get:

$$|E_S| = \sum_{X_i} |E_{X_i}| \geq \sum_{X_i} (|S(X_i)| + 2) = |S| + 2j \geq |S|$$

Since $|\mathcal{N}(S)| = |E_S|$, by Hall's theorem, there exist a perfect triangle-side matching and therefore a triangle assignment ρ with $c(\rho) = 0$. For planar graphs $|\Delta| = O(n)$ and $m = O(n)$, the $O(n^2 \log n)$ running time follows from Theorem 6. □

In the following we provide the construction of the $\Omega(n^2)$ lower bound on the expected squared ℓ_2-error of any ε-LWDP algorithm.

Proof of Lemma 2. Let $G = (V, E, w)$ be a graph with $V = \{v_1, \ldots, v_n\}$ consisting of $n - 2$ triangles that share the common edge $\{v_1, v_2\}$. That is,

$$E = \{\{v_1, v_2\}\} \cup \{\{v_1, v_j\}, \{v_2, v_j\} : v_j \in V \setminus \{v_1, v_2\}\}.$$

Let the weights of all edges except $\{v_1, v_2\}$ be 0, and let the threshold be $\lambda = 1$. We further assume that the weights of all edges except $\{v_1, v_2\}$ are public information, while the weight of $\{v_1, v_2\}$ belongs to $\{0, 1\}$.

Suppose, for contradiction, that for every constant $c(\varepsilon)$ depending on ε such that $c(\varepsilon) > 0$, there exists an ε-LWDP algorithm $\mathcal{M}$ whose expected squared ℓ_2-error is smaller than $c(\varepsilon)n^2$.

Let $\mathcal{G}$ be an algorithm that guesses the weight of $\{v_1, v_2\}$ from the output of $\mathcal{M}$. Specifically, $\mathcal{G}$ outputs 0 if the output of $\mathcal{M}$ is smaller than $(n-2)/2$, and outputs 1 otherwise. By the post-processing property of ε-LWDP, the algorithm $\mathcal{G}$ is also ε-LWDP.

The two graphs only differ in $w_{v_1v_2}$, which changes the incident vector of v_1 and v_2 by at most 1. By the per-node guarantee of LWDP and composition over v_1 and v_2, the output distribution of G changes by a factor of at most $e^{2\varepsilon}$. Therefore, $\mathcal{G}$ must guess the weight incorrectly with probability at least $\frac{1}{1+e^{2\varepsilon}}$. Whenever $\mathcal{G}$ guesses incorrectly, the squared ℓ_2-error of $\mathcal{M}$ is at least $\frac{(n-2)^2}{4}$. Hence, the expected squared ℓ_2-error of $\mathcal{M}$ is at least $\frac{1}{1+e^{2\varepsilon}} \cdot \frac{(n-2)^2}{4}$. This contradicts the assumption that the expected squared ℓ_2-error is smaller than $c(\varepsilon)n^2$ for every $c(\varepsilon) > 0$. Therefore, no such ε-LWDP algorithm $\mathcal{M}$ exists. □

B.4 Bounds for degeneracy-bounded graphs

In the following, we prove the statements regarding graphs with degeneracy γ.

Definition 14 (Degeneracy)**.** *For a graph G the degeneracy $\gamma(G)$ is the smallest number, such that in any subgraph of G there exists a node with induced degree at most $\gamma(G)$.*

For a graph $G = (V, E)$, let $\prec$ be an ordering of the nodes. We say v precedes u if $v \prec u$. Let $d_v^{\prec}$ denote the forward degree of $v \in V$:

$$d_v^{\prec} = |\{\{u, v\} \in E : v \prec u\}|.$$

Lemma 28. *Let $G = (V, E)$ be a graph with ordering $\prec$. Then there exists a triangle-node assignment ρ with $c(\rho) = \sum_{v \in V} (d_v^{\prec})^3$.*

Proof. For a triangle $T = \{u, v, w\} \in \Delta$, assign the triangle to the edge whose nodes occur first in the ordering. $\rho(T) = v$ if and only if $u \prec v$, and $w \prec v$. The load of an edge $\{u, w\} \in E$ is therefore upper bounded by $\min\{d_u^{\prec}, d_w^{\prec}\}$.

$$\#C_4' \leq \sum_{\{u,w\} \in E} \binom{\min\{d_u^{\prec}, d_w^{\prec}\}}{2}$$

The statement then follows:

$$\sum_{\{u,w\} \in E} \binom{\min\{d_u^{\prec}, d_w^{\prec}\}}{2} \leq \sum_{v \in V} d_v^{\prec} \binom{d_v^{\prec}}{2} = O(\sum_{v \in V} (d_v^{\prec})^3).$$

□

Proof of Lemma 3. Let $G = (V, E)$ be a graph with degeneracy bounded by γ. Let $\prec$ denote the ordering resulting by repeatedly removing the node with the smallest degree. The forward degree is then bounded by γ and we get $\#C_4' = O(n\gamma^3)$ by Lemma 28. □

We now present the proof of Theorem 9. We present a proof for the more general framework of (ε, δ)-LWDP.

Definition 15 ((ε, δ)-Local weight differential privacy)**.** *Let $\varepsilon > 0$ and $\delta \in (0, 1)$. A randomized query $\mathcal{R}$ satisfies (ε, δ)-local weight differential privacy ((ε, δ)-LWDP) if, for any neighboring weight vectors w, w', and any measurable set of outputs S,*

$$\Pr[\mathcal{R}(w) \in S] \leq e^{\varepsilon} \Pr[\mathcal{R}(w') \in S] + \delta.$$

An algorithm $\mathcal{A}$ is said to be (ε, δ)-LWDP if, for any node v_i, and any (possibly adaptive) sequence of queries $\mathcal{R}_1, \ldots, \mathcal{R}_\kappa$ posed to v_i, where each query $\mathcal{R}_j$ satisfies $(\varepsilon_j, \delta_j)$-LWDP for $1 \leq j \leq \kappa$, it holds that $\sum_{j=1}^{\kappa} \varepsilon_j \leq \varepsilon$ and $\sum_{j=1}^{\kappa} \delta_j \leq \delta$. For $\delta = 0$, (ε, δ)-LWDP is equivalent to ε-LWDP.

As described in Section 4 the proof is based on a reduction from the problem of computing the sum of n values under local differential privacy. This proof follows the approach proposed in [15]. They analyzed the setting for general graphs while we provide the analysis for graphs with degeneracy γ.

Definition 16 (*SUM$_n$* [15])**.** *Given $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, we define SUM$_n$ as*

$$\mathit{SUM}_n = \sum_{i=1}^{n} x_i.$$

Lemma 29 ([10], [5], [27], [15])**.** *There exists a constant $c > 0$ such that for every $\varepsilon \in (0,1), n \in \mathbb{N}, \alpha_0 \in (0,n]$ and $\delta \in [0, \frac{1}{10^5} \cdot \frac{\varepsilon^3 \alpha_0^2}{n^3 \ln(n^2/\varepsilon\alpha_0)}]$, if $\mathcal{A}$ is an (ε,δ)-LDP algorithm where each user i has the private bit x_i and $\mathcal{A}$ estimates SUM$_n$ up to an additive error α_0 with probability at least $2/3$, then $\alpha_0 \geq c\sqrt{n}/\varepsilon$.*

In the following, we provide two separate lower bounds that then prove Theorem 9.

Lemma 30. *There exists a family of graphs with degeneracy γ and a constant $c > 0$ such that for every $\varepsilon \in (0, 1/2)$, $n \in \mathbb{N}$, $\alpha \in (0, k(n-\gamma)]$, and*

$$\delta \in \left[0,\ \frac{4}{10^5} \cdot \frac{\varepsilon^3 \alpha^2}{(n-\gamma)^2 k^3 \ln\bigl(k^2(n-\gamma)/(2\varepsilon\alpha)\bigr)}\right], \quad \textit{where } k = \lfloor \gamma/2 \rfloor,$$

every (ε,δ)-LWDP algorithm that, given a graph with n nodes and degeneracy γ, outputs an estimate of the number of below-threshold triangles with additive error at most α with probability at least $2/3$, must satisfy $\alpha \geq c\,(n-\gamma)\sqrt{\lfloor \gamma/2 \rfloor}/\varepsilon$.

Proof. Let $k = \lfloor \gamma/2 \rfloor$. We reduce from an instance of SUM_k. Let $\vec{x} = (x_1, \ldots, x_k) \in \{0,1\}^k$ denote the private bits of the k users.

We construct a weighted, undirected graph $G = (V, E, w)$ with n nodes as follows. For each $i \in \{1, \ldots, k\}$, introduce two nodes u_{2i} and u_{2i+1}, together with an edge $\{u_{2i}, u_{2i+1}\}$ of weight $-x_i$. Let

$$S = \{u_i : i \in \{2, \ldots, 2k+1\}\}$$

denote the set of these nodes. If γ is odd, add one additional node to S that is not paired with any other node. Let H be the set of the remaining $n - \gamma$ hub nodes. For every pair consisting of one node in S and one node in H, add an edge of weight 0.

We first show that G has degeneracy γ. Let $A \subseteq V$ and consider the induced subgraph $G(A)$. If $A \cap H \neq \emptyset$, then any node in $A \cap H$ has degree at most γ in $G(A)$, since it is only connected to nodes in S. If $A \cap H = \emptyset$, then $A \subseteq S$, and every node in S has degree at most 1 in $G(A)$. Hence, every induced subgraph contains a node of degree at most γ, implying that the degeneracy of G is γ.

Each pair of nodes u_{2i}, u_{2i+1} is associated with user i, while nodes in H can be simulated by any user since all incident edge weights are public. Changing x_i changes only the weight of $\{u_{2i}, u_{2i+1}\}$, by one unit. Hence, $w^{u_{2i}}$ and $w^{u_{2i+1}}$ each change by one unit in ℓ_1-norm, while all other incident-weight vectors remain unchanged. The resulting protocol for SUM_k is $(2\varepsilon, 2\delta)$-LDP.

Let $\mathcal{A}$ be an (ε,δ)-LWDP algorithm that estimates the number of below-threshold triangles with additive error α. Fix the threshold $\lambda = 0$, and let o denote the output of $\mathcal{A}$.

Since there are no edges within H and only disjoint edges within S, every triangle must consist of two nodes from a pair in S and one node from H. Thus, all triangles are of the form $\{u_{2i}, u_{2i+1}, h\}$ for $i \in \{1, \ldots, k\}$ and $h \in H$. The additional node (if γ is odd) does not participate in any triangle.

If $x_i = 1$, then the edge $\{u_{2i}, u_{2i+1}\}$ has weight -1, and each triangle $\{u_{2i}, u_{2i+1}, h\}$ has total weight -1, which is below the threshold. Hence, user i contributes exactly $n - \gamma$ below-threshold triangles. If $x_i = 0$, no such triangles are below the threshold.

Therefore,

$$s = \frac{o}{n-\gamma}$$

yields an estimate for SUM_k with additive error $\alpha_0 = \frac{\alpha}{n-\gamma}$.

We apply Lemma 29 with privacy parameters $(2\varepsilon, 2\delta)$, k users, and $\alpha_0 = \alpha/(n-\gamma)$. Its premises are $2\varepsilon \in (0,1)$, $\alpha_0 \in (0,k]$, and $2\delta \le \frac{1}{10^5} \cdot \frac{(2\varepsilon)^3 \alpha_0^2}{k^3 \ln(k^2/(2\varepsilon\alpha_0))}$, which hold by the assumptions on ε, α and δ. It yields $\alpha/(n-\gamma) \ge c_0\sqrt{k}/(2\varepsilon)$, where c_0 is the constant of Lemma 29. Setting $c = c_0/2$ completes the proof. □

Lemma 31. *There exists a family of graphs with degeneracy γ and a constant $c > 0$ such that for every $\varepsilon \in (0, 1/2)$, $n \in \mathbb{N}$, $\alpha \in (0, (\gamma-1)\lfloor (n-\gamma)/2 \rfloor]$, and*

$$\delta \in \left[0, \frac{4}{10^5} \cdot \frac{\varepsilon^3 \alpha^2}{(\gamma-1)^2 k^3 \ln(k^2(\gamma-1)/(2\varepsilon\alpha))}\right], \quad \textit{where } k = \left\lfloor \frac{n-\gamma}{2} \right\rfloor,$$

every (ε, δ)-local weight differential privacy algorithm that, given a graph with n nodes and degeneracy γ, outputs an estimate of the number of below-threshold triangles with additive error at most α, with probability at least $2/3$, must satisfy

$$\alpha \ge c \frac{(\gamma-1)\sqrt{k}}{\varepsilon}.$$

Proof. Let $k = \lfloor (n-\gamma)/2 \rfloor$, and consider an instance of SUM_k with private bits $\vec{x} = (x_1, \ldots, x_k) \in \{0,1\}^k$.

Construct a weighted undirected graph $G = (V, E, w)$ as follows. For each $i \in \{1, \ldots, k\}$, introduce two nodes u_{2i} and u_{2i+1} together with an edge $\{u_{2i}, u_{2i+1}\}$ of weight $-x_i$. Let P be the set of these $2k$ nodes. If $n-\gamma$ is odd, there is one additional node in P that is not part of any pair; it has no effect on the construction below. Let H be a set of $\gamma - 1$ hub nodes. Add edges of weight 0 between every node in P and every node in H. Finally, add one isolated node so that the total number of nodes is n.

We verify that the degeneracy of G is γ. Consider any subset $A \subseteq V$ and the induced subgraph $G(A)$. If A contains a node from P, then that node has degree at most γ in $G(A)$. If A contains only nodes from H and possibly the isolated node, then every node has degree 0. Hence every induced subgraph has a node of degree at most γ, and since nodes of degree γ exist, the degeneracy is exactly γ.

Fix threshold $\lambda = 0$, and let o be the output of an (ε, δ)-LWDP algorithm $\mathcal{A}$ that estimates the number of below-threshold triangles with additive error α. Every triangle consists of two nodes from a pair in P and one node in H. If $x_i = 1$, then the edge $\{u_{2i}, u_{2i+1}\}$ has weight -1, and each triangle $\{u_{2i}, u_{2i+1}, h\}$ has total weight -1. Hence, user i contributes exactly $|H| = \gamma - 1$ below-threshold triangles. If $x_i = 0$, no such triangle is below the threshold.

Therefore,$s = \frac{o}{\gamma-1}$ is a $2\varepsilon, 2\delta$-local differentially private estimate of SUM_k with additive error $\alpha_0 = \frac{\alpha}{\gamma-1}$.

Substituting $\alpha_0 = \alpha/(\gamma-1)$ and k in Lemma 29 gives

$$\delta \in \left[0, \frac{1}{10^5} \cdot \frac{\varepsilon^2 \alpha^2}{(\gamma-1)^2 k^3 \ln\left(\frac{k^2(\gamma-1)}{\varepsilon\alpha}\right)}\right]$$

and

$$\alpha \ge c \frac{(\gamma-1)\sqrt{k}}{\varepsilon},$$

which proves the claim. □

We are now able to provide the complete proof of Theorem 9.

Proof of Theorem 9. The claim follows from Lemma 30 and Lemma 31. □

C Efficient smooth sensitivity computation for biased estimator

In this section we provide the complete algorithm for the computation of the smooth sensitivity of the function $f'_v(w^v)$ using the biased estimator B'_T introduced in Section 5.

C.1 Properties of smooth sensitivity for f'_v

Let $E(v) = \{e \in E : v \in e\}$ denote the incident edges of v. We slightly adapt the notation for β-smooth sensitivity. Instead of maximizing over $y \in \mathbb{Z}^d$, we reparameterize using the difference $z = y - w$ to obtain that:

$$S^*_{f'_v,\beta}(w) = \max_{z\in\mathbb{Z}^d}(LS_f(w+z)e^{-\beta\cdot|z|})$$

Let $B := \{b \in \{-1,0,1\}^d : |b|_1 = 1\}$. We have that $w \sim w'$ if there exists $b \in B$ such that $w' = w + b$. Hence,

$$S^*_{f'_v,\beta}(w) = \max_{b\in B}\max_{z\in\mathbb{Z}^d}|f'_v(w+z+b) - f'_v(w+z)|e^{-\beta\cdot|z|}$$

Assume i is the unique index with $b_i \neq 0$. If the weight of edge i increases (i.e., $b_i = 1$), then the local sensitivity is the number of triangles that contain edge i and whose total weight is $\lambda - 1$ in $w + z$. In $w + z + b$, each such triangle now has total weight λ and is therefore flipped to being above the threshold λ. Similarly, if the weight of edge i decreases (i.e., $b_i = -1$), then the local sensitivity is the number of triangles that contain edge i and whose total weight is λ in $w + z$.

In the following, for a node v, let Δ_v denote the set of triangles assigned to v, and for each $T \in \Delta_v$, let $T \subseteq E$ denote the set of three edges forming that triangle. We analyze the case $b_i = 1$; the case $b_i = -1$ follows by a symmetric argument. Define $\Delta_v(i) = \{\, j \in E(v) \mid \exists\, \{i,j,k\} \in \Delta_v \,\}$ as the set of incident edges that form a triangle together with $i \in E(v)$. For an edge e, let $u(e)$ denote the endpoint of e distinct from v. In this setting, the local sensitivity is given by the following expression:

$$\max_{i\in E(v)}\max_{z\in\mathbb{Z}^d}\sum_{j\in\Delta_v(i)} 1\{w_j + z_j + w'_{u(i)u(j)} = \lambda - 1 - w_i - z_i\}e^{-\beta|z|}$$

We obtain this since $|f'_v(w+z+b) - f'_v(w+z)|$ equals the number of triangles that cross the threshold when the weight vector changes from $w + z$ to $w + z + b$, i.e., triangles that are below the threshold under $w + z$ but are no less than it under $w + z + b$. Equivalently, it is the number of triangles incident to edge i whose weight sum under $w + z$ is exactly $\lambda - 1$, so that adding b increases the sum to λ.

Fix the index i. For all $j \in \Delta_v(i)$, let $c_j = w_j + w'_{u(i)u(j)}$. The smooth sensitivity for a fixed i and b becomes,

$$\max_{z\in\mathbb{Z}^d}\sum_{j\in\Delta_v(i)} 1\{c_j + z_j = \lambda - 1 - w_i - z_i\}e^{-\beta|z|} \tag{2}$$

We aim to solve optimization problem (2) by choosing z so that many of the values c_j are shifted to the *target* of $t = \lambda - 1 - w_i - z_i$. Figure 3 visualizes this setting.

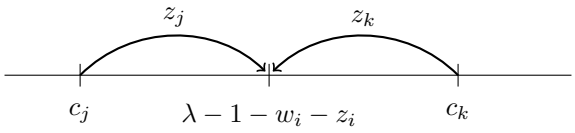


Figure 3: Representation of smooth sensitivity as a problem of shifting values to a target

Lemma 32. *There is an optimal solution z to optimization problem 2 in which z_i satisfies $z_i = \lambda - 1 - w_i - c_j$ for some $j \in \Delta_v(i)$, or $z_i = 0$. Equivalently, there is an optimal solution whose target t is either $t = \lambda - 1 - w_i$ or $t = c_j$ for some $j \in \Delta_v(i)$.*

Proof. Assume z^* to be an optimal solution to optimization problem 2 such that $z^*_i \neq 0$ and $z^*_i \neq \lambda - 1 - w_i - c_j$ for all $j \in \Delta_v(i)$. Let $t^* = \lambda - 1 - w_i - z^*_i$ be the target in this optimal assignment, let the objective value be $k^* e^{-\beta|z^*|}$, and let $s = |\Delta_v(i)|$. This means we have k^* elements of c_j shifted to $\lambda - 1 - w_i - z_i$.

We now either show that all optimal solutions satisfy the claimed property or construct another optimal solution z satisfying it. We do so by shifting the target $t = \lambda - 1 - w_i - z_i$ and adjusting the values z_j so that the indicators $1\{c_j + z_j = \lambda - 1 - w_i - z_i\}$ remain unchanged for all $j \in \Delta_v(i)$.

Let $S_L^{t^*} = \{j \in [s] \mid c_j \leq t^* \wedge c_j + z_j^* = t^*\}$ be the set of values that are shifted from the left side of the target t^*, and let $l = |S_L^{t^*}|$ denote its size: Similarly, define $S_R^{t^*} = \{j \in [s] \mid c_j > t^* \wedge c_j + z_j^* = t^*\}$ to be the set of values shifted from the right side of the target t^*, and let $r = |S_R^{t^*}|$ be its size.

Consider the case where $l = r$. Without loss of generality, we assume $t^* < \lambda - 1 - w_i$. If there is j such that $t^* < c_j \leq \lambda - 1 - w_i$, we set t to smallest c_j in that range. Otherwise, we set t to $\lambda - 1 - w_i$. We then shift all k^* elements in the sets $S_L^{t^*}$ and $S_R^{t^*}$ to the target t. Let $\delta = t - t^*$ be the distance of the target shift. We obtain that:

$$\begin{aligned}
|z| &= \sum_{c_j \in S_L^t} |z_j| + \sum_{c_j \in S_R^t} |z_j| + |z_i| \\
&\leq \sum_{c_j \in S_L^{t^*}} (|z_j^*| + \delta) + \sum_{c_j \in S_R^{t^*}} (|z_j^*| - \delta) + |z_i| \\
&< |z^*| + (l - r)\delta = |z^*|.
\end{aligned}$$

For the second inequality, we use the fact that $|z_i| < |z_i^*|$. From this, we have $k^* e^{-\beta|z|} > k^* e^{-\beta|z^*|}$. This contradiction shows that for $l = r$, no such optimal solution z^* can exist and all optimal solutions must satisfy the property.

For $l > r$, we move the target t to the first c_j to the left of t^*. Since $l > r$, such a value must exist. We obtain that:

$$\begin{aligned}
|z| &= \sum_{c_j \in S_L^t} |z_j| + \sum_{c_j \in S_R^t} |z_j| + |z_i| \\
&\leq \sum_{c_j \in S_L^{t^*}} (|z_j^*| - \delta) + \sum_{c_j \in S_R^{t^*}} (|z_j^*| + \delta) + |z_i| \\
&\leq (|z^*| + \delta) + (r - l)\delta \leq |z^*|
\end{aligned}$$

Since z connects the same values as z^* and satisfies $|z| \leq |z^*|$, it achieves an objective value that is at least as good as that of z^*. In the case where $r < l$, we shift the target t to the first c_j located to the right of t^*. Applying the same reasoning as in the previous case confirms that z is indeed an optimal solution. □

From Lemma 32, it is sufficient to iterate over $t \in c \cup \{\lambda - 1 - w_i\}$ and find the optimal value of the optimization 2 in order to compute the global optimum. We note that, for a fixed t and fixed number $k = \sum_{j=1}^{s} 1\{c_j + z_j = \lambda - 1 - w_i - z_i\}$, it is optimal to shift the k-closest values in c to the target t. Let $d_t(j)$ be the j-th smallest distance from any value in c to target t and $s = |\Delta_v(i)|$. The optimal number of connected values k^* for target t is then given by

$$k^* = \arg\max_{k \in [s]} k e^{-\beta \sum_{j=1}^{k} d_t(j)}.$$

The idea of our algorithm is to iterate over every possible target t in non-decreasing order and compute k^* together with its objective value to compute the global maximum. The straightforward approach of iterating over every possible target t and possible k results in a running time of $O(d^2)$ for a fixed edge i. From now, we show how this running time can be reduced to $O(d \log^2 d)$. For simplicity, we denote $d_t(j)$ by $d(j)$ when the target t is fixed and clear from context.

Lemma 33. *Let $Obj(k) = k e^{-\beta \sum_{j=1}^{k} d(j)}$ for $k \in [s]$. We have that $Obj(k+1) > Obj(k)$ if and only if $k/(k+1) < e^{-\beta d(k+1)}$.*

Proof. When $Obj(k+1) > Obj(k)$, we have that

$$k e^{-\beta \sum_{j=1}^{k} d(j)} < (k+1) e^{-\beta \sum_{j=1}^{k+1} d(j)}.$$

This implies $k/(k+1) < e^{-\beta d(k+1)}$. □

Since $e^{-\beta d(k+1)}$ is decreasing and $k/(k+1)$ is increasing for growing k we can use binary search to find the largest k for which this inequality is satisfied.

The evaluation of $k/(k+1) < e^{-\beta d(k+1)}$ requires the access to $d(k+1)$ so the $(k+1)$-th closest distance to the current target. In the next section, we describe how to efficiently retrieve the distance.

C.2 Efficient distance retrieval for single target

The computation of the optimal k^* requires access to the k-th smallest distance to the target point t. Moreover, evaluating the objective function for k^* requires the sum of the k smallest distances to the target point. In the following, we introduce a single-target data structure that supports these operations efficiently for a target point t.

Let $p = (p_0, \ldots, p_{n-1})$ be a sorted array of integers with $p_i \leq p_{i+1}$ for all i. For a target $t \in \mathbb{Z}$, define the single-target distance

$$d_t(x) := |x - t|.$$

The data structure $S(p, t)$ supports: (i) retrieval of the k-th smallest distance, (ii) retrieval of the sum of the k smallest distances, and (iii) target updates.

C.2.1 Initialization

We partition p into two regions according to its relation to the value t. Define the index

$$j = \min\{i \mid p_i \geq t\}.$$

This index partitions p into two disjoint regions: elements strictly to the left of the target ($p_i < t$) and elements to the right or equal to the target ($p_i \geq t$). Additionally, we compute the prefix-sum array $a = (a_1, \ldots, a_n)$ with $a_i := \sum_{v=0}^{i-1} p_v$.

C.2.2 Retrieving k-th closest Distance

The KthDistance(k) operation selects k elements by choosing l elements from the left region and $k - l$ elements from the right region. The maximum selected distance is given by

$$h(l) := \max\big(t - p_{j-l},\ p_{j+(k-l)-1} - t\big).$$

The left term of $h(l)$ increases with l, while the right term decreases with l. Thus, the minimizing value of $h(l)$ (corresponding to the k-th smallest distance) can be found via a binary search over the number of elements l taken from the left.

C.2.3 Sum of k Closest Distances

The sum of the k smallest distances is computed using the same split point l obtained during the execution of KthDistance(k). By maintaining the prefix sum array a, the sum of the k-th closest distances can be computed efficiently.

The l closest values to t from the left region are stored in the index interval $(j - l, j - 1)$. The sum of their distances to t is therefore

$$\sum_{i=j-l}^{j-1} (t - p_i) = l \cdot t - \sum_{i=j-l}^{j-1} p_i = l \cdot t - (a[j] - a[j - l]).$$

The remaining $r = k - l$ elements are taken from the right region. These values are stored in the index interval $(j, j + r - 1)$. Their contribution to the distance sum is given by

$$\sum_{i=j}^{j+r-1} (p_i - t) = \sum_{i=j}^{j+r-1} p_i - r \cdot t = (a[j + r] - a[j]) - r \cdot t.$$

Combining both contributions yields the total sum of the k closest distances to t:

$$l \cdot t - (a[j] - a[j - l]) + (a[j + k - l] - a[j]) - (k - l) \cdot t.$$

C.2.4 Target Updates

For an update of the target to a new value t', recomputing the index j via binary search restores all invariants.

Lemma 34. *Initialization takes $O(n)$ time. Each of KthDistance, SumKDistances, and UpdateTarget runs in $O(\log n)$ time.*

Proof. Initialization requires a scan of p to compute the prefix-sum array a, giving a complexity of $O(n)$. For KthDistance, SumKDistances, and UpdateTarget, each operation requires a binary search that dominates the computational costs. Therefore, for each operation, the running time is bounded by $O(\log n)$. □

We are now able to provide the running time for finding the optimal number of shifted values k^*.

Lemma 35. *Finding the optimal k^* for a fixed target t takes $O(\log^2 d)$ time.*

Proof. The computation consists of two binary searches. The outer one is over the number of shifted values k as described in Lemma 33. The number of iteration is therefore bounded by $O(\log d)$ where each iteration computes the $(k+1)$-th closest distance in $O(\log d)$ time with the single-target data structure S. □

C.3 Efficient computation of smooth-sensitivity

We now present Algorithm 3 to summarize the discussion in this section. We note that for better readability Algorithm 3 only handles the case that the weight of the edge $e \in E(v)$ is increased. For the case that w_e decreases by 1, the target value is set to $\lambda - w_e$ instead of $\lambda - w_e - 1$. We

Algorithm 3 EFFICIENT COMPUTATION OF SMOOTH SENSITIVITY

Input: $G = (V, E, w)$, $v \in V$, Δ_v, $\beta > 0$, threshold λ
Output: β-smooth sensitivity of f'_v

1: $s \leftarrow 0$ ▷ Initialization
2: **for** $e \in E(v)$ **do**
3: $c \leftarrow \{w_j + w'_{ij} : j \in \Delta_v(e)\}$
4: $\mathcal{T} \leftarrow c \cup \{\lambda - 1 - w_e\}$ ▷ Target set
5: Let $t_0 \leq \cdots \leq t_d$ be the elements of $\mathcal{T}$.
6: We initialize $S(c, t_0)$ as the single target distance datastructure
7: **for** $\mathsf{i} \in \{0, \ldots, d\}$ **do**
8: Find optimal z^* with k^* for target t_i. ▷ Appendix C.1
9: $s \leftarrow \max\left\{s, k^* e^{-\beta|z^*|}\right\}$.
10: If $\mathsf{i} \neq d$, UpdateTarget(t_{i+1}).
11: **end for**
12: **end for**
13: **return** s

iterate over the targets $t \in \mathcal{T}$ in nondecreasing order. For each target t, we apply the algorithm from Appendix C.1 to determine the optimal number of values k^* that should be connected to t. If this yields a larger objective value, we update s. Finally, as we move to the next target, we call UpdateTarget for the single-target data structure S.

Lemma 36 (Computation time of Algorithm 3)**.** *Algorithm 3 computes the β-smooth sensitivity of f'_v using the biased estimator B'_T for a node v with degree d in $O(d^2 \log^2 d)$ time.*

Proof. The node v has d incident edges and therefore the outer for loop is executed d times. Initializing the single-target data structure $S(c, t_0)$ takes $O(d)$ time. By Lemma 35, Line 8 runs in $O(\log^2 d)$ time and Line 9 takes $O(\log^2 d)$ by using SumKDistance for the computation of $|z^*|$. Lastly, updating S takes $O(\log d)$ time. The inner loop is executed $O(d)$ where each iteration runs in $O(\log^2 d)$. Hence, the total running time is in $O(d^2 \log^2 d)$. □

D Efficient smooth sensitivity computation for unbiased estimator

We now propose the algorithm for the smooth sensitivity of $f'_v(w^v)$ using the unbiased estimator U'_T. It follows the same idea of shifting values to a target as in Appendix C but requires a more detailed

analysis since U'_T can take four distinct values compared to the two of B'_T. We recall the following notations. For a node $v \in V$, let $E(v) = \{v \in e : e \in E\}$ denote the set of incident edges. For each $i \in E(v)$, let $\Delta_v(i) = \{j \in E(v) : \exists\{i, j, k\} \in \Delta_v\}$ denote the set of incident edge indices that form a triangle together with edge i and a non-incident edge of v. Moreover, let $u(i)$ and $u(j)$ denote the other endpoints of edges i and j, respectively.

In this section, we provide the smooth sensitivity algorithm from Section 5 for the unbiased estimator U'_T. We first recall the reparameterized definition of the smooth sensitivity for the count of local below-threshold triangles f'_v.

$$S^*_{f'_v,\beta}(w) = \max_{b\in B} \max_{z\in\mathbb{Z}^d} |f'_v(w + z + b) - f'_v(w + z)|\, e^{-\beta\cdot|z|}$$

Here, $b = (b_j)_{j\in E(v)}$ is a vector with exactly one nonzero entry, where $b_j \in \{-1, 0, 1\}$. It specifies which edge weight is increased or decreased by one. Appendix D.5 iterates over all $2d$ possible choices of b. We then consider the case when b is fixed. Suppose that i is the index of b such that $b_i \neq 0$.

With these definitions, we expand the formulation of the smooth sensitivity explicitly for the unbiased estimator and a fixed i:

$$\max_{z\in\mathbb{Z}^d} \left| \sum_{j\in\Delta_v(i)} U'_T(w_T + z_j + z_i + b_i) - U'_T(w_T + z_j + z_i) \right| e^{-\beta\cdot|z|}$$

We introduce $y(j) = U'_T(w_T + z_j + z_i + b_i) - U'_T(w_T + z_j + z_i)$, representing the contribution of $j \in \Delta_v(i)$ to the sum. This yields

$$\max_{z\in\mathbb{Z}^d} \left| \sum_{j\in\Delta_v(i)} y(j) \right| e^{-\beta\cdot|z|} \tag{3}$$

We compute the maximum corresponding to a positive sum and a negative sum separately and take the maximum of the two in order to evaluate the absolute value:

$$\max \left\{ \max_{z\in\mathbb{Z}^d} \sum_{j\in\Delta_v(i)} y(j) e^{-\beta\cdot|z|}, \max_{z\in\mathbb{Z}^d} - \sum_{j\in\Delta_v(i)} y(j) e^{-\beta\cdot|z|} \right\}$$

We refer to these as the objective functions with positive and negative sum contribution, respectively. In the following we fix $b_i = 1$, the same arguments and algorithm can be applied to the case $b_i = -1$.

D.1 Optimal solution for positive sum contribution

In this section, we analyze the case where the sum contributes positively to the objective function:

$$\max_{z\in\mathbb{Z}^d} \sum_{j\in\Delta_v(i)} y(j) e^{-\beta\cdot|z|} \tag{4}$$

In the biased setting, the difference between the two estimators $y(j)$ for each $j \in \Delta_v(i)$ takes values in $\{0, 1\}$, corresponding to whether a triangle flips or remains unchanged. In contrast, in the unbiased setting, $y(j)$ may attain three distinct values. For $b_i = 1$, the possible values are 0, $\frac{p}{(1-p)^2}$, and $-1 - \frac{2p}{(1-p)^2}$. We introduce the shorthand $x = \frac{p}{(1-p)^2}$ to simplify notation. Let $c = \left(w_j + w_{u(i)u(j)}\right)_{j\in\Delta_v(i)}$ denote the partial triangle weights, and define $t_i = \lambda - 1 - w_i - z_i$ as the target value. We partition the index set $\Delta_v(i)$ into three disjoint subsets M, N, O. Specifically, M denotes the set of values adjacent to the target:

$$M = \{j \in \Delta_v(i) : c_j = t_i - 1 \vee c_j = t_i + 1\}$$

The set N consists of values exactly on the target:

$$N = \{j \in \Delta_v(i) : c_j = t_i\}.$$

Finally, O contains all remaining values:

$$O = \{j \in \Delta_v(i) : c_j < t_i - 1 \vee c_j > t_i + 1\}.$$

By construction, the sets M, N, and O form a partition of $\Delta_v(i)$.

Lemma 37. *For a solution z to optimization problem* (4) *with $j \in \Delta_v(i)$ and $z_j = 0$, the contribution $y(j)$ satisfies*

$$y(j) = \begin{cases} x & \text{if } j \in M \\ -1 - 2x & \text{if } j \in N \\ 0 & \text{if } j \in O \end{cases}$$

Proof. Under the assumption $z_j = 0$, the contribution of index j simplifies to

$$y(j) = U'_T(c_j + w_i + z_i + 1) - U'_T(c_j + w_i + z_i).$$

Consider first $j \in M$ with $c_j = t_i - 1$. In this case, we obtain

$$U'_T(t_i - 1 + w_i + z_i + 1) - U'_T(t_i - 1 + w_i + z_i).$$

Substituting the definition of t_i yields

$$U'_T(\lambda - 1) - U'_T(\lambda - 2) = x.$$

For $j \in N$, the contribution evaluates to $y(j) = U'_T(\lambda) - U'_T(\lambda - 1) = -1 - 2x$. Finally, for $j \in O$, both $U'_T(c_j + w_i + z_i + 1)$ and $U'_T(c_j + w_i + z_i)$ are equal, implying $y(j) = 0$. □

D.2 Finding optimal shift sets

We now determine how to set the values z_j for each of the sets M, N, and O. Let k_a, for $a \in \{M, N, O\}$, denote the number of values shifted from set a. We say that a value c_j is shifted if $z_j > 0$.

D.2.1 Shifting values from M

We first show that no value in M is shifted.

Lemma 38. *In any optimal solution z^*, it holds that, for all $j \in M$, $z^*_j = 0$ and $y(j) = x$.*

Proof. Let $j \in M$ and consider $z_j = 0$. By Lemma 37, the contribution of j is $y(j) = x$. Moreover, this is the maximum possible contribution of a single value. Increasing z_j strictly decreases the objective value due to the multiplicative factor $e^{-\beta|z|}$. Hence, in any optimal solution, $z_j = 0$ for all $j \in M$ must hold. □

Consequently, it is always optimal to set $z_j = 0$ for all $j \in M$, and we have $k_M = |M|$.

D.2.2 Shifting values from N

Next, we show that it is always optimal to shift all values in N before shifting any value from O.

Lemma 39. *Let z^* be an optimal solution. If there exists $j \in N$ such that $|z^*_j| \neq 1$, then, for all $o \in O$, $z^*_o = 0$.*

Proof. Let $o \in O$ such that $z^*_o \neq 0$, and assume there exists $n \in N$ with $|z^*_n| \neq 1$. We first consider the case $z^*_n = 0$. Then, n contributes $-1 - 2x$ to the sum, while o contributes x, yielding

$$\sum_{j \in \Delta_i(v)} y(j) = \sum_{j \in \Delta_i(v) \setminus \{o,n\}} y(j) + x - 1 - 2x.$$

We construct a new solution z' by setting $z'_j = z^*_j$ for all $j \in \Delta_i(v) \setminus \{o, n\}$, $z'_o = 0$, and $z'_n = 1$. Clearly, $|z'| \leq |z^*|$. Moreover, o now contributes 0, while n contributes x, giving

$$\sum_{j=1}^{s} y(j) = \sum_{j \in \Delta_i(v) \setminus \{o,n\}} y(j) + x,$$

which strictly improves the objective value, contradicting the optimality of z^*. If instead $|z^*_n| > 1$, then n contributes 0 to the sum. Decreasing $|z^*_n|$ to 1 causes n to contribute x, which again improves the objective value. □

Lemma 39 implies that all values in N must be shifted before any value in O is shifted. We now determine the optimal number of values shifted from N.

We begin with the solution z such that $z_j = 0$ for all $j \in \Delta_i(v)$. In this configuration, values in M contribute x, values in N contribute $-1-2x$, and values in O contribute 0. Shifting a value from N changes its contribution from $-1-2x$ to x. Thus, the objective function as a function of k_N is

$$((|M| + k_N)x + (|N| - k_N)(-1-2x))e^{-\beta(k_N+|z_i|)}$$

Since $|z|$ is fixed when optimizing over k_N, the factor $e^{-\beta|z_i|}$ is a positive constant independent of k_N and therefore does not affect the maximizer of the objective function. Hence, rewriting this expression yields

$$obj(k_N) = \big(|M|x + (-1-2x)|N| + (1+3x)k_N\big)e^{-\beta k_N}.$$

Let $A = |M|x + (-1-2x)|N|$ and $B = 1+3x$, and define $obj(k_N) = (A + Bk_N)e^{-\beta k_N}$. Differentiating with respect to k_N gives

$$\frac{d}{dk_N} obj(k_N) = \big(B - \beta(A + Bk_N)\big)e^{-\beta k_N}.$$

Since $e^{-\beta k_N} > 0$ for all k_N, the unique root satisfies $B - \beta(A + Bk_N) = 0$, yielding

$$k_N = \frac{1}{\beta} - \frac{A}{B}.$$

Substituting A and B gives

$$k_N^* = \frac{1}{\beta} - \frac{|M|x + (-1-2x)|N|}{1+3x},$$

which identifies the point beyond which shifting additional values from N is no longer beneficial. Since k_N may be fractional, we select either $\lfloor k_N \rfloor$ or $\lceil k_N \rceil$, depending on which yields a higher objective value. If $k_N < |N|$, we found the optimal solution, otherwise, we proceed to shift values from O.

D.2.3 Shifting values from O

We first show that any shifted value from O is always shifted to the closer of $t_i - 1$ or $t_i + 1$.

Lemma 40. *Let z^* be an optimal solution and let $j \in O$ with $z_j^* \neq 0$. Then*

$$|z_j^*| = \min\{|c_j - t_i - 1|,\ |c_j - t_i + 1|\}.$$

Proof. If $z_j^* \neq t_i - 1 - c_j$ and $z_j^* \neq t_i + 1 - c_j$, then $y(j) \leq 0$ implies that setting $z_j = 0$ results in a strictly higher objective value. For both choices $z_j^* = t_i - 1 - c_j$ and $z_j^* = t_i + 1 - c_j$, the contribution is $y(j) = x$. Among these, the choice minimizing $|z_j^*|$ maximizes the objective, proving the claim. □

We now describe how to determine the optimal number of values shifted from O. We note that it is optimal to shift the values from O in increasing order of their distance to $t_i - 1$ or $t_i + 1$. At this stage, the sets M and N are fixed, and all values in $M \cup N$ contribute x. Shifting k_O values from O yields the objective value

$$obj(k_O) = x(k_O + |N| + |M|)e^{-\beta(|N|+\sum_j^{k_O} d_t(j))}$$

where $d_t(j)$ denotes the j-th closest distance from a value of O to $t-1$ or $t+1$ and we drop the constant factor of $e^{-\beta|z_i|}$.

Lemma 41. *Let k denote the number of values shifted from O. The $(k+1)$-st value is shifted if and only if*

$$\frac{k + |M| + |N|}{k + |M| + |N| + 1} < e^{-\beta d_t(k+1)}.$$

Proof. The condition $obj(k+1) > obj(k)$ is equivalent to

$$x(k+1+|M|+|N|)e^{-\beta(\sum_{j=1}^{k+1} d_t(j))} > x(k+|M|+|N|)e^{-\beta(\sum_{j=1}^{k} d_t(j))},$$

which simplifies to the stated inequality. □

Since $|M|$ and $|N|$ are fixed, the left-hand side of the inequality is monotonically decreasing in k, while the right-hand side is monotonically non-increasing. Thus, we can use binary search to find the largest value of k for which the inequality holds and shift $k+1$ values from O in the optimal solution.

Finally, we show that it suffices to consider only a linear number of target values.

Lemma 42. *There exists an optimal solution z such that $z_i = \lambda - w_i - c_j$, $z_i = \lambda - 1 - w_i - c_j$, or $z_i = \lambda - 2 - w_i - c_j$ for some $j \in [s]$, or $z_i = 0$. Equivalently, there exists an optimal solution with target t such that $t = c_j$, $t = c_j - 1$, or $t = c_j + 1$ for some $j \in [s]$, or $t = \lambda - 1 - w_i$.*

Proof. Let z^* be an optimal solution that does not satisfy the stated property, and let t^* denote its target. By construction, $|t^* - c_j| \geq 2$ for all $j \in [s]$, implying that the sets M and N are empty. In this case, the same arguments as in Lemma 32 apply here, and we are able to generate another optimal solution z' from z^* that satisfies the claim. □

The preceding lemma implies that it suffices to consider $O(d)$ distinct values for z_i. Thus, we can iterate over all candidate values while maintaining an efficient overall algorithm.

D.3 Optimal solution for negative sum contribution

We now consider the case in which the sum contributes negatively to the objective function:

$$\max_{z \in \mathbb{Z}^d} \sum_{j \in \Delta_v(i)} -y(j)e^{-\beta|z|}. \tag{5}$$

We reuse the definitions of the sets M, N, and O introduced in the analysis of the positive contribution case. By Lemma 37, for a solution z with $z_j = 0$ for all $j \in \Delta_v(i)$, the contributions satisfy $-y(j) = -x$ for $j \in M$, $-y(j) = 1 + 2x$ for $j \in N$, and $-y(j) = 0$ for $j \in O$. Consequently, in the negative contribution case, the roles of the sets M and N are effectively reversed.

Lemma 43. *In any optimal solution z^*, it holds that for all $j \in N$, $z_j^* = 0$ and $-y(j) = 1 + 2x$.*

Proof. Let $j \in N$. Setting $z_j^* = 0$ yields the maximum possible contribution of $1 + 2x$ to the sum. Any choice with $z_j^* \neq 0$ decreases the objective value in (5) due to $e^{-\beta \cdot |z|}$. □

An argument analogous to that of Lemma 39 applies to the sets M and O in this setting. Specifically, values from M must be shifted before any value from O is shifted. Let $0 \leq k_M \leq |M|$ denote the number of values shifted from M. Values from N and shifted values of M contribute $1 + 2x$, while the remaining unshifted values in M contribute $-x$. Shifting k_M values from M results in $|z| = k_M$. The objective function, without constant factor $e^{-\beta|z_i|}$, as a function of k_M is therefore given by

$$obj(k_M) = \big((1+2x)(|N| + k_M) - x(|M| - k_M)\big)e^{-\beta k_M}.$$

Maximizing this expression yields an optimal number of shifted values

$$k_M^* = \frac{1}{\beta} - \frac{(1+2x)|N| - x|M|}{1+3x}.$$

Evaluating the objective function at the neighboring integer values of k_M^* identifies the optimal number of values shifted from M. Finally, we consider shifting values from O. Let k_O denote the number of values shifted from O. The objective function can be written as

$$obj(k_O) = (1+2x)(k_O + |N| + |M|)e^{-\beta(|M| + \sum_{j=1}^{k_O} d_t(j))},$$

where $d_t(j)$ denotes the j-th smallest distance from the target t. In contrast to the positive contribution case, where distances were measured relative to $t-1$ and $t+1$, distances here are taken with respect to t.

Using the same argument as in Lemma 41, we apply binary search to find the largest value k satisfying

$$\frac{k+|M|+|N|}{k+|M|+|N|+1} < e^{-\beta d_t(k+1)}.$$

The optimal solution then shifts $k_O = k+1$ values from O. Here, we are able to reuse the single-target data structure from Appendix C.2 to compute $d_t(k+1)$.

D.4 Efficient distance retrieval for double target

The computation of k_O^* requires access to the k-th smallest distance to $t-1$ or $t+1$ for problem (4). Moreover, evaluating the objective function for k_O^* requires the sum of the k smallest distances to the respective target points. In the following, we introduce a data structure that supports these operations efficiently for the two target points $t-1$ and $t+1$.

Let $p = (p_0, \ldots, p_{n-1})$ be a sorted array of integers with $p_i \leq p_{i+1}$ for all i. For a target $t \in \mathbb{Z}$, define the double-target distance

$$d_t(x) := \min\big(|x-(t-1)|,\ |x-(t+1)|\big).$$

The double-target data structure $D(p,t)$ supports: (i) retrieval of the k-th smallest distance, (ii) retrieval of the sum of the k smallest distances, and (iii) target updates.

D.4.1 Initialization

We partition p into regions according to its relation to the values $t-1$, t, and $t+1$. Define the indices

$$\ell_0 = \min\{i \mid p_i \geq t-1\}, \qquad \ell_1 = \min\{i \mid p_i > t-1\},$$
$$r_0 = \min\{i \mid p_i \geq t+1\}, \qquad r_1 = \min\{i \mid p_i > t+1\}.$$

These indices partition p into five disjoint regions: $p_i < t-1$, $p_i = t-1$, $p_i = t$, $p_i = t+1$, and $p_i > t+1$. Additionally, we compute the prefix-sum array $a = (a_1, \ldots, a_n)$ with $a_i := \sum_{j=0}^{i-1} p_j$.

D.4.2 Retrieving k-th closest Distance

Let $z := (\ell_1 - \ell_0) + (r_1 - r_0)$ define the number of points on $t-1$ or $t+1$, and let $m := r_0 - \ell_1$ define the number of points on t. The KthDistance(k) operation proceeds in three phases:

1. If $k \leq z$, return 0.
2. Else if $k \leq z + m$, return 1.
3. Otherwise, let $k' := k - z - m$ and consider only indices $i < \ell_0$ and $i \geq r_1$.

The remaining distances form two sorted sequences. We select k' elements by choosing l elements from the left region and $k'-l$ from the right region. The maximum selected distance is

$$h(l) := \max\big((t-1) - p_{\ell_0 - l},\ p_{r_1 + (k'-l) - 1} - (t+1)\big).$$

The left term of $h(l)$ increases with l, while the right term decreases with l. Thus, the minimizing value of $h(l)$ (corresponding to the k-th smallest distance) can be found via binary search.

D.4.3 Sum of k Closest Distances

The sum of the k smallest distances is computed using the same split point l obtained during the execution of KthDistance(k). Equivalently to the KthDistance function, let $z := (\ell_1 - \ell_0) + (r_1 - r_0)$ define the number of points on $t-1$ or $t+1$, and let $m := r_0 - \ell_1$ define the number of points on t. Elements at distance zero contribute nothing to the sum, while m elements at distance one contribute exactly their count. The remaining contributions are computed using prefix sums.

Let l denote the number of selected elements from the left region. The l closest values to $t-1$ among those satisfying $p_i < t-1$ are stored in the index interval $(\ell_0 - l, \ell_0 - 1)$. The sum of their distances to $t-1$ is therefore

$$\sum_{i=\ell_0-l}^{\ell_0-1} ((t-1) - p_i) = l \cdot (t-1) - \sum_{i=\ell_0-l}^{\ell_0-1} p_i = l \cdot (t-1) - (a[\ell_0] - a[\ell_0 - l]).$$

The remaining $r = k - l - m - z$ elements are taken from the right region. The r closest values to $t+1$ among those satisfying $p_i > t+1$ are stored in the index interval $(r_1, r_1 + r - 1)$. Their contribution to the distance sum is given by

$$\sum_{i=r_1}^{r_1+r-1} (p_i - (t+1)) = \sum_{i=r_1}^{r_1+r-1} p_i - r \cdot (t+1) = (a[r_1 + r] - a[r_1]) - r \cdot (t+1)$$

Combining all contributions yields the sum of the k closest distances to either $t-1$ or $t+1$:

$$m + l \cdot (t-1) - (a[\ell_0] - a[\ell_0 - l]) + (a[r_1 + r] - a[r_1]) - r \cdot (t+1).$$

D.4.4 Target Updates

For an update of the target to t', recomputing the indices ℓ_0, ℓ_1, r_0, r_1 via binary search restores all invariants.

Lemma 44. *Initialization takes $O(n)$ time. Each of KthDistance, SumKDistances, and UpdateTarget runs in $O(\log n)$ time.*

Proof. Initialization requires a scan of p to compute the prefix-sum array a, giving a complexity of $O(n)$. For KthDistance, SumKDistances, and UpdateTarget each requires a binary search dominating the costs. Therefore, for each operation the running time is bounded by $O(\log n)$. □

D.5 Algorithm

In this subsection, we present Algorithm 4, which computes the β-smooth sensitivity of f'_v for the unbiased estimator U'_T. The algorithm explicitly handles the case $b_i = 1$; the case $b_i = -1$ is treated analogously by using the initial target $\lambda - w_e$ instead of $\lambda - 1 - w_e$.

For each incident edge $e \in E(v)$, the algorithm computes the partial triangle weights c and constructs the set of candidate target values $\mathcal{T}$ which is processed in sorted order. For each target, the algorithm computes the optimal solution to the optimization problems (4) and (5). In the positive contribution case, a double-target distance data structure is used to determine the optimal number of shifted values from N, and, if necessary, from O via binary search. The negative contribution case is handled analogously using a single-target distance data structure from Appendix C.2, first optimizing over M and then over O. In both cases, targets are updated efficiently between iterations. The algorithm returns the maximum objective value over all incident edges and candidate targets. We now analyze the running time of the proposed algorithm by first bounding the cost of computing the optimal solution for a fixed incident edge and a fixed contribution sign.

Lemma 45. *The optimal solutions to optimization problems* (4) *and* (5) *can be computed in $O(d \log^2 d)$ time.*

Proof. We analyze the running time for a fixed incident edge and a fixed sign of the contribution.

The computation begins by initializing the corresponding target-distance data structure, which requires $O(d)$ time. The algorithm then iterates over $O(d)$ candidate target values. In each iteration, the optimal number of shifted values from N (or M in the negative case) is computed in constant time. If necessary, the optimal number of shifted values from O is determined via a binary search over an interval of size $O(d)$. Each step of the binary search performs a distance query on the data structure, which takes $O(\log d)$ time. Consequently, determining the optimal number of shifted values from O requires $O(\log^2 d)$ time per target. Finally, updating the target in the data structure between consecutive iterations requires $O(\log d)$ time. Combining these bounds, each iteration costs $O(\log^2 d)$ time, and over $O(d)$ target values the total running time is $O(d \log^2 d)$. □

Algorithm 4 CALCULATION OF SMOOTH SENSITIVITY WITH UNBIASED ESTIMATOR

Input: $G = (V, E, w)$, $v \in V$, Δ_v, $\beta > 0$
Output: β-smooth sensitivity of f'_v

$s \leftarrow 0$ ▷ Initialization
for $e \in E(v)$ **do**
  $c \leftarrow \{w_j + w'_{ij} : j \in \Delta_v(e)\}$
  $\mathcal{T} \leftarrow \{c_j + \delta : c_j \in c, \delta \in \{-1, 0, 1\}\} \cup \{\lambda - 1 - w_e\}$ ▷ Target Set
  Let $t_0 \leq \cdots \leq t_d$ be the elements of $\mathcal{T}$.
  Solve Problem (4):
  Initialize double target data structure $D(c, t_0)$.
  for $\mathsf{i} \in \{0, \ldots, |\mathcal{T}| - 1\}$ **do**
    Find optimal number of values shifted from N.
    If $k^*_N < |N|$, update s if necessary, advance target in D to t_{i+1}, and continue.
    Find optimal k^*_O with binary search.
    Update global maximum s if necessary.
    Advance target in D to t_{i+1}.
  end for
  Solve Problem (5):
  Initialize single-target data structure $S(c, t_0)$.
  for $\mathsf{i} \in \{0, \ldots, |\mathcal{T}| - 1\}$ **do**
    Find optimal number of values shifted from M.
    If $k^*_M < |M|$, update s if necessary, advance target in S to t_{i+1}, and continue.
    Find optimal k^*_O with binary search.
    Update global maximum s if necessary.
    Advance target in S to t_{i+1}.
  end for
end for
return s

Using this bound, we next derive the overall running time of Algorithm 4.

Lemma 46 (Computation time of Algorithm 4)**.** *Algorithm 4 computes the β-smooth sensitivity of f'_v using the unbiased estimator U'_T for a node $v \in V$ of degree d in $O(d^2 \log^2 d)$ time.*

Proof. Algorithm 4 iterates over all incident edges $e \in E(v)$, of which there are at most d. For each such edge, the algorithm performs an initialization step followed by the computation of both the positive and negative sum contributions.

For a fixed edge e, the initialization requires enumerating all triangles containing e and computing the corresponding partial triangle weights. Since $|\Delta_v(e)| \leq d$, this step requires $O(d)$ time to collect the values and $O(d \log d)$ time to sort them.

By Lemma 45, the computation of the optimal solution for the positive contribution and for the negative contribution each requires $O(d \log^2 d)$ time. Thus, the total time per incident edge is $O(d \log^2 d)$.

For d incident edges, this results in an overall running time of $O(d^2 \log^2 d)$. □

Proof of Theorem 10. Follows from Lemma 36 and Lemma 46. □

E Experiments

In this section we give a detailed description on how we constructed the graph ML-TELE-400 for the experiments of Section 6 and provide additional results on two more graphs.

Table 1: Experiment Dataset Overview

Property	ML-Tele-400	musae-wiki	GMWCS
n	400	11631	1618
m	79800	170773	1847
$\lvert\Delta\rvert$	10586800	623433	132
$\max w_T$	194	764590	-255
$\min w_T$	0	0	-522

E.1 Details for the ML-TELE-400 graph construction

In 2014, Telecom Italia, in collaboration with several research institutions, organized the "Telecom Italia Big Data Challenge"[3], releasing an anonymized telecommunications dataset for the city of Milan [4]. The dataset ("Telecommunications - MI to MI" [26]) records aggregated communication intensity between geographic areas of Milan, which are partitioned into a 100×100 grid of equally sized cells. For every pair of areas, it reports the volume of telecommunication activity in 10-minute intervals.

In our analysis, we consider the 10-minute interval on Sunday, November 3, 2013, from 18:50 to 19:00 (local time). We assume that each base station serves a 5×5 block of cells, which yields a total of 400 base stations across the city. From this, we construct an undirected weighted graph in which each node corresponds to a radio base station, each edge represents a connection between two stations, and each edge weight is the number of calls between them. Under local weight differential privacy, the influence of any single call on the released statistics is bounded. The published count provides only limited, probabilistically bounded evidence about whether a specific call took place between two areas.

Let I_e denote the interaction intensity associated with an edge $e \in E$ obtained from the dataset, and let L be the total number of calls that occurred during the selected 10-minute interval. We set

$$w_e \;=\; L \cdot \frac{I_e}{\sum_{e' \in E} I_{e'}}$$

for all $e \in E$. The total number of calls L is not explicitly reported in the challenge data. Assuming an average of two calls per person per day [28] and a population of approximately 1.3 million in Milan [4], we estimate $L \approx 20{,}000$ for a 10-minute interval. This estimate indicates that the number of edges m is much larger than L, so most edges obtain weight zero. We summarize the resulting graph, which we refer to as ML-TELE-400, in Table 1.

E.2 Experiment on Wikipedia network

We further evaluate our methods on MUSAE-WIKI, a page-page network of English Wikipedia articles related to crocodiles [44]. Nodes represent articles, edges represent mutual links between them, and each node is annotated with the average monthly traffic of the article. Since the traffic on individual links is not reported, we estimate the weight of an edge by splitting the traffic of both endpoints among their neighbors in proportion to the neighbors' traffic. The properties of the resulting graph are summarized in Table 1. Compared to ML-TELE-400, the graph contains far fewer triangles relative to its number of edges and has nodes of very high degree. We choose the thresholds as percentiles of the triangle weights.

Figure 4(a) shows that the privacy budget at which the unbiased variants overtake the biased ones is shifted towards larger values of ε. While on ML-TELE-400 *smooth-unbiased* already outperforms all biased variants for small budgets, on MUSAE-WIKI the biased variants remain preferable up to moderate budgets, and the unbiased variants only become more accurate for larger ε. This shift aligns with Theorems 4 and 5. The relative error of the biased estimator is dominated by its bias and is therefore largely independent of the graph size, whereas the error of the unbiased estimator depends strongly on the privacy budget and has to be small compared to the true count. Since MUSAE-WIKI has a much smaller true count but a larger maximum degree than ML-TELE-400, a larger privacy budget is required before the unbiased estimator becomes more accurate than the biased one. Figure 4(b)

[3] from BigDataChallenge contest

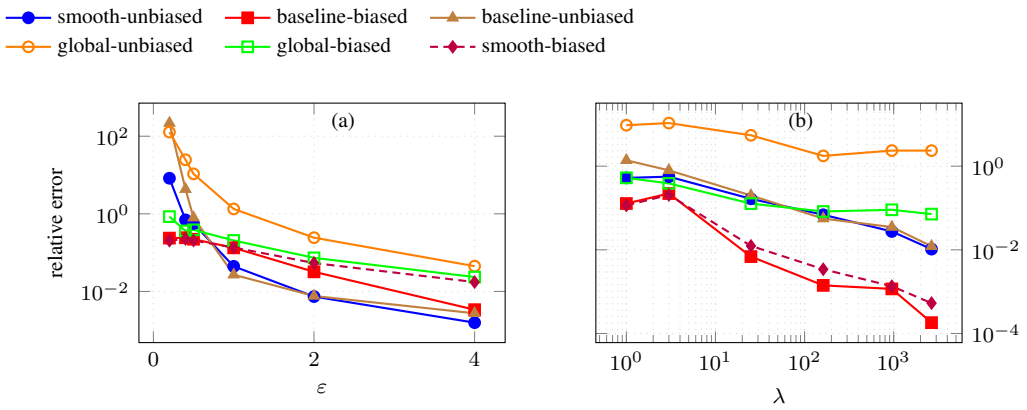


Figure 4: Relative error across all methods for the input graph MUSAE-WIKI: (a) as a function of the privacy budget ε for $\lambda = 3$, (b) as a function of the threshold λ for $\varepsilon = 0.5$.

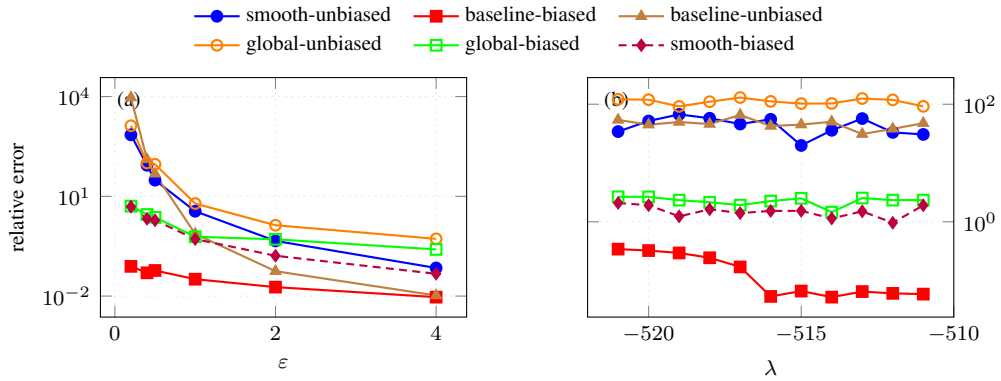


Figure 5: Relative error of all methods on the input graph GMWCS: (a) as a function of the privacy budget ε for $\lambda = -511$, (b) as a function of the threshold λ for $\varepsilon = 0.5$.

confirms this for a fixed budget of $\varepsilon = 0.5$, which lies in the regime favoring the biased estimator on this graph, across all thresholds *baseline-biased* and *smooth-biased* attain the lowest errors.

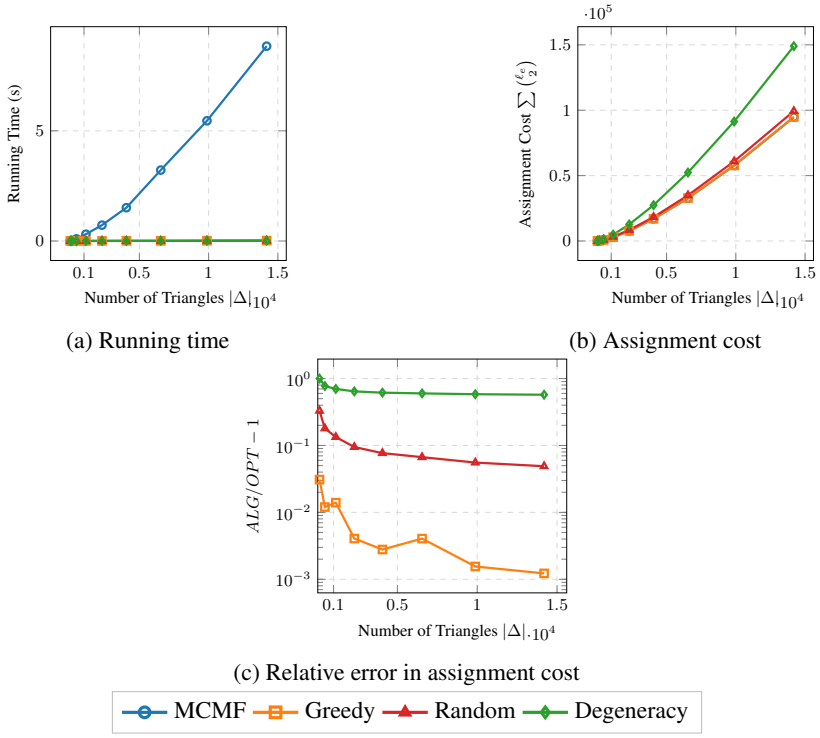
E.3 Experiment on GMWCS

We next evaluate our methods on GMWCS, the largest instance from the benchmark set of [35], which was also included in the *Generalized Maximum-Weight Connected Subgraph* (GMWCS) track of the *11th DIMACS Implementation Challenge in Collaboration with ICERM: Steiner Tree Problems*. The properties of this instance are summarized in Table 1. Although this graph has more nodes than ML-TELE-400, it is extremely sparse, with an average degree only slightly above two and only 132 triangles.

Figure 5 shows the same effect as on MUSAE-WIKI, but even more pronounced. Since the true below-threshold count is very small while every node adds noise in Step 2, the biased variants are preferable for almost all privacy budgets, and the unbiased variants only become accurate for the largest values of ε. In particular, the single-round *baseline-biased* attains the lowest error across all privacy budgets and thresholds, which shows that a single-round approach is preferable on small and sparse graphs.

E.4 Experiments on assignment functions

In this section, we present empirical results to compare the assignment strategies we proposed in Section 4.

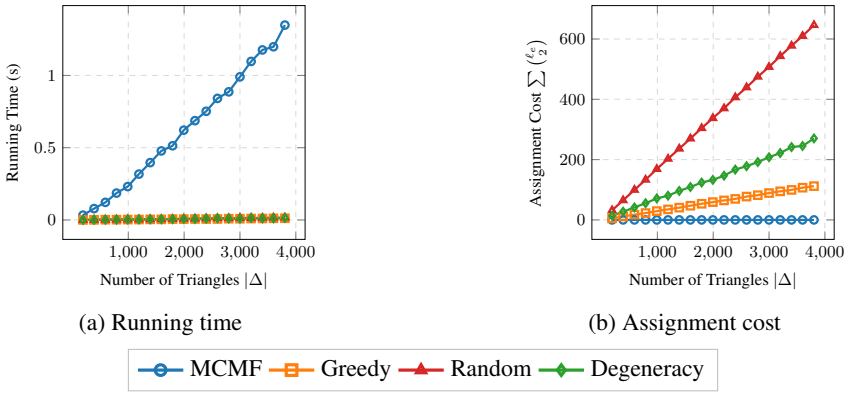


(a) Running time

(b) Assignment cost

(c) Relative error in assignment cost

Figure 6: Experimental results of assignment algorithms on complete graphs



(a) Running time

(b) Assignment cost

Figure 7: Experimental results of assignment algorithms on graphs generated via Delaunay triangulation.

Experimental Setup. We evaluate the assignment cost and running time of four strategies: *Min-Cost Max-Flow* (*MCMF*), *Greedy*, *Degeneracy*, and a *Random* baseline. The *Random* baseline assigns each triangle independently and uniformly at random to one of its three edges. The *MCMF* algorithm follows the graph construction and the proof of Theorem 6. The *Degeneracy* algorithm follows the construction in the proof of Lemma 3, using the linear-time $O(n + m)$ algorithm of [36] to compute the degeneracy ordering. We evaluate all algorithms on three graph families: complete graphs K_n, planar graphs generated via Delaunay triangulation, and subgraphs of a real-world Google web graph [32], which was released in 2002 by Google as part of a programming contest. The web graph is part of the SNAP network datasets [31]. Since the *Greedy* and *Random* strategies depend on the order in which triangles are processed, we execute each algorithm 20 times for planar and complete graphs and 5 times on the web graph on independently and uniformly shuffled triangle lists and report the mean. Across all graph types and sizes, the standard deviation of both running time and assignment cost over these trials remained negligibly small, confirming that the chosen repetitions

are sufficient to obtain stable estimates. All experiments were conducted on a 2024 MacBook Pro with an Apple M4 Pro chip and 48 GB of memory.

Complete Graphs. Figure 6 reports assignment cost, running time, and relative cost with respect to *MCMF* on complete graphs K_n. *Greedy*, *Random*, and *Degeneracy* all run in under 0.02 seconds across all tested sizes, whereas *MCMF* already exceeds 5 seconds at $n = 40$ nodes ($|\Delta| = 9{,}880$ triangles), confirming that computing the optimal solution is intractable at scale. Figure 6c shows the relative error of the assignment cost, defined as $ALG/OPT - 1$, where ALG denotes the cost obtained by each algorithm and OPT denotes the optimal cost obtained by *MCMF*. *Greedy* achieves roughly one order of magnitude lower relative error than *Random* across all sizes, with gaps of approximately 0.15% versus 5.5% at $n = 40$. The *Degeneracy* ordering performs poorly on complete graphs. Nodes that appear early in the ordering are assigned edges connecting them to almost all other nodes, resulting in highly skewed loads. In this unstructured setting, both *Greedy* and *Random* distribute load far more evenly and therefore achieve more competitive costs.

Planar Graphs via Delaunay Triangulation. Figure 7 reports results on planar graphs generated by Delaunay triangulation. As established in Lemma 1, an optimal assignment of cost zero always exists for planar graphs, and therefore *MCMF* achieves it in all instances. The running time profile mirrors that of complete graphs: *MCMF* scales poorly while the three heuristics remain fast, with *MCMF* reaching 1.1 seconds at $|\Delta| \approx 3{,}200$ triangles compared to under 0.015 seconds for all heuristics. In terms of assignment cost, the structural ordering of *Degeneracy* now proves beneficial. It outperforms *Random* by a factor of approximately $3\times$ across all sizes, as the degeneracy ordering aligns well with the local triangle structure of planar graphs. *Greedy* retains the best cost among all non-optimal strategies, achieving costs roughly $6\times$ lower than *Random* and $2\times$ lower than *Degeneracy* at $|\Delta| \approx 3{,}200$.

Google Web Graph. Figure 8 reports results on subgraphs of the Google web graph, a real-world instance with bounded degeneracy of 44. *MCMF* is omitted as it is no longer tractable at this scale. To preserve the power-law degree distribution characteristic of web graphs, we subsample by retaining each edge independently with probability $p \in \{0.05, 0.1, 0.25, 0.5, 0.7, 0.8, 0.9, 1\}$, rather than sampling nodes. A node of degree d retains expected degree pd under edge sampling, preserving the relative ordering of degrees and thus the shape of the degree distribution. For every value of p, we generated 5 random subgraphs and we averaged the resulting costs and running time of all algorithms. The results reveal a pronounced and consistent separation between strategies as the graph density increases. At a sampling fraction of $p = 0.2$ ($|\Delta| \approx 107{,}380$), *Greedy* achieves a cost of 969, compared to $6{,}519$ for *Degeneracy* and $23{,}342$ for *Random*. At the largest fraction $p = 1.0$, corresponding to over 13 million triangles, *Greedy* achieves a cost of $26{,}427{,}913$ versus $41{,}021{,}836$ for *Degeneracy* ($1.6\times$ higher) and $69{,}055{,}393$ for *Random* ($2.6\times$ higher). Running times at this scale are 61 seconds for *Greedy*, 59 seconds for *Degeneracy*, and 47 seconds for *Random*, all within the same order of magnitude.

F Generalized neighboring relation

Definition 2 protects a change of a single unit in the incident-weight vector of a node. In this section, we show that all our mechanisms and guarantees extend to neighboring weight vectors that differ by up to r units in ℓ_1-norm, for any fixed $r \in \mathbb{N}$.

F.1 Definition and interpretation

Definition 17 (r-neighboring weight vectors)**.** *For $r \in \mathbb{N}$, two weight vectors $w, w' \in \mathbb{Z}^d$ are r-neighboring, denoted $w \sim_r w'$, if $\|w - w'\|_1 \leq r$. A randomized query $\mathcal{R}$ satisfies ε-LWDP under $\sim_r$ if $\Pr[\mathcal{R}(w) \in S] \leq e^{\varepsilon} \Pr[\mathcal{R}(w') \in S]$ for all $w \sim_r w'$ and all outcome sets S. Privacy budgets compose over the queries issued to a node as in Definition 3.*

For $r = 1$, we recover Definition 2, which is an event-level guarantee that protects a single communication or mobility event on any edge incident to a node. Larger values of r protect larger units of data. A trip along a simple path enters and leaves each intermediate node through two distinct incident edges and touches each endpoint through a single edge. Hence, a single trip changes the

incident-weight vector of every node by at most two units, and $r = 2$ protects the trip at every node on the path. More generally, if a user contributes at most t such trips, $r = 2t$ protects all of the user's contributions at each node.

We denote by $GS_r(f) := \max_{w \sim_r w'} |f(w) - f(w')|$ and $LS_{r,f}(w) := \max_{y \sim_r w} |f(w) - f(y)|$ the global and local sensitivity under $\sim_r$. All results below follow from a simple decomposition: if $\|w - w'\|_1 = j$, there exist $y_0 = w, y_1, \ldots, y_j = w'$ such that $\|y_t - y_{t-1}\|_1 = 1$ and $\|y_t - w\|_1 = t$ for all $t \in [j]$.

F.2 Step 1: releasing incident weights

Lemma 47. *Adding independent noise drawn from $DLap(e^{-\varepsilon_1/r})$ to each entry of the incident-weight vector satisfies ε_1-LWDP under $\sim_r$.*

Proof. Let $p = e^{-\varepsilon_1/r}$ and $w \sim_r w'$. For any output $x \in \mathbb{Z}^d$,

$$\frac{\Pr[\tilde{w} = x \mid w]}{\Pr[\tilde{w} = x \mid w']} = \prod_{i=1}^{d} p^{|x_i - w_i| - |x_i - w'_i|} \leq p^{-\|w - w'\|_1} = e^{\varepsilon_1 \|w - w'\|_1 / r} \leq e^{\varepsilon_1},$$

where the first inequality follows from the triangle inequality. □

F.3 Assignment and estimators

The assignment ρ depends only on the public topology and is unchanged. The biased estimator B'_T does not depend on the noise distribution and is unchanged. The unbiased estimator U'_T is used with $p = e^{-\varepsilon_1/r}$ in Definition 11. Its unbiasedness (Appendix A.2) only requires the noise on w'_{ux} to follow $DLap(p)$ and therefore continues to hold.

F.4 Step 2 with global sensitivity

Lemma 48. *For every node $v \in V$, $GS_r(f'_v) \leq r \cdot GS(f'_v)$, where $GS(f'_v)$ is given by Theorem 1 with $p = e^{-\varepsilon_1/r}$. In particular, $GS_r(f'_v)$ can be bounded in $O(d_v^2)$ time.*

Proof. Let $w \sim_r w'$ with $\|w - w'\|_1 = j \leq r$, and let $y_0, \ldots, y_j$ be the decomposition above. Then $|f'_v(w) - f'_v(w')| \leq \sum_{t=1}^{j} |f'_v(y_t) - f'_v(y_{t-1})| \leq j \cdot GS(f'_v) \leq r \cdot GS(f'_v)$. The running time follows from Theorem 1. □

A tighter bound is available at a higher computational cost. Let $\Delta_v(S) = \{T \in \Delta_v : T \cap S \neq \emptyset\}$ for a set S of incident edges of v. If $w \sim_r w'$, the vector $w - w'$ is supported on a set S of at most r incident edges, and only the estimators of triangles in $\Delta_v(S)$ can change. Each of them changes by at most $GS(g_v^T)$, since $GS(g_v^T)$ equals the range of the estimator ($\{0, 1\}$ for B'_T and $[-p/(1-p)^2,\ 1 + p/(1-p)^2]$ for U'_T). Hence, $GS_r(f'_v) \leq GS(g_v^T) \max_{|S| \leq r} |\Delta_v(S)|$, which can be computed in $O(d_v^{r+1})$ time by enumerating all sets of at most r incident edges.

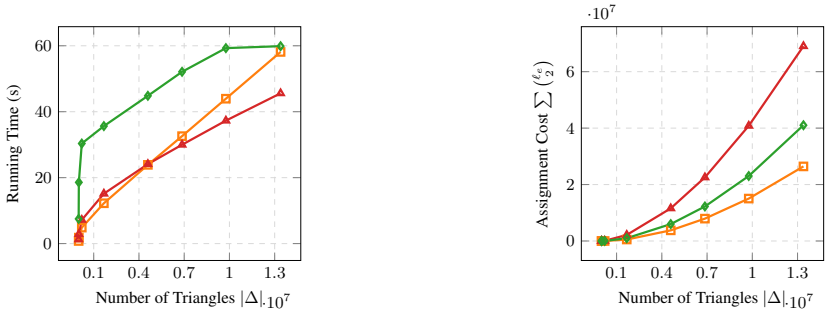
F.5 Step 2 with smooth sensitivity

The smooth-sensitivity query remains ε_2-LWDP if $S^*_{f,\beta}$ in Definition 9 is replaced by any β-smooth upper bound on the local sensitivity, i.e., any function U with $U(w) \geq LS_{r,f}(w)$ for all w and $U(w) \leq e^{\beta} U(w')$ for all $w \sim_r w'$ [40]. We show that such a bound can be computed with the algorithms of Appendices C and D without any change to their running time.

Lemma 49. *Let $\beta > 0$, $\beta' = \beta/r$, and let $S^*_{f'_v,\beta'}$ denote the β'-smooth sensitivity of f'_v under the unit relation, as computed in Appendices C and D. Then*

$$U(w) := r\, e^{\beta'(r-1)}\, S^*_{f'_v,\beta'}(w)$$

is a β-smooth upper bound on LS_{r,f'_v} under $\sim_r$. Consequently, it can be computed in $O(d^2 \log^2 d)$ time for a node of degree d.



(a) Running time.

(b) Assignment cost.

Figure 8: Experimental results of assignment algorithms on graphs sub-sampled from a Google web graph.

Proof. Upper bound. Let $y \sim_r w$ with $\|y - w\|_1 = j \le r$, and let $y_0 = w, \dots, y_j = y$ be the decomposition above. Since $\|y_{t-1} - w\|_1 \le r - 1$ for all $t \in [j]$, the definition of the smooth sensitivity gives $LS_{f'_v}(y_{t-1}) \le e^{\beta'(r-1)} S^*_{f'_v,\beta'}(w)$. Therefore,

$$|f'_v(y) - f'_v(w)| \le \sum_{t=1}^{j} LS_{f'_v}(y_{t-1}) \le r\, e^{\beta'(r-1)} S^*_{f'_v,\beta'}(w) = U(w).$$

Maximizing over $y \sim_r w$ yields $LS_{r,f'_v}(w) \le U(w)$.

Smoothness. Let $w \sim_r w'$. For every y, $\|w - y\|_1 \le \|w' - y\|_1 + r$, hence

$$S^*_{f'_v,\beta'}(w) = \max_y LS_{f'_v}(y) e^{-\beta' \|w-y\|_1} \ge e^{-\beta' r} \max_y LS_{f'_v}(y) e^{-\beta' \|w'-y\|_1} = e^{-\beta} S^*_{f'_v,\beta'}(w').$$

Multiplying by $re^{\beta'(r-1)}$ gives $U(w') \le e^{\beta} U(w)$. The running time follows from Theorem 10, applied with β' instead of β. □

F.6 Privacy and precision

Theorem 12. *Algorithm 1, with Step 1 noise drawn from $DLap(e^{-\varepsilon_1/r})$, the unbiased estimator instantiated with $p = e^{-\varepsilon_1/r}$, and Step 2 calibrated to $r \cdot GS(f'_v)$ (Laplace query) or to U from Lemma 49 (smooth-sensitivity query), satisfies $(\varepsilon_1 + \varepsilon_2)$-LWDP under $\sim_r$.*

Proof. Step 1 is ε_1-LWDP under $\sim_r$ by Lemma 47. The assignment depends only on public data. Step 2 is ε_2-LWDP under $\sim_r$ for any fixed outcome of Step 1, by Lemma 48 for the Laplace query and by Lemma 49 for the smooth-sensitivity query. The claim follows by sequential composition. □

Theorem 13. *Under the modifications of Theorem 12, the expected squared ℓ_2-error of Algorithm 1 is bounded by the expressions of Theorems 4 and 5 with ε_1 replaced by ε_1/r and the first term ($d_{\max} m$) multiplied by r^2 for the Laplace query and by $r^2 e^{\varepsilon_2/3}$ for the smooth-sensitivity query with $\Gamma = 4$. For fixed r and $\varepsilon_2 = O(1)$, the bounds are thus of the same order as in the case $r = 1$ with privacy budget ε_1/r in Step 1.*

Proof. The bias, variance, and covariance bounds of Appendices A.1 and A.2 depend on the Step 1 noise only through p, so they hold with $p = e^{-\varepsilon_1/r}$, i.e., with ε_1 replaced by ε_1/r. The Step 2 noise remains unbiased and independent of all other randomness. For the Laplace query, its variance is $2\sum_v (r\, GS(f'_v))^2/\varepsilon_2^2$, i.e., r^2 times the bounds of Lemmas 11 and 17. For the smooth-sensitivity query with $\Gamma = 4$, we have $\beta = \varepsilon_2/6$, and since $S^*_{f'_v,\beta'} \le GS(f'_v)$, it holds that $U \le re^{\beta(r-1)/r} GS(f'_v) \le re^{\varepsilon_2/6} GS(f'_v)$. The argument of Lemma 18 then yields the bound of the Laplace case multiplied by $e^{\varepsilon_2/3}$. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction correctly state the algorithmic contributions (variance reducing assignment, efficient smooth sensitivity computation), the key theoretical results (bounds for planar and degeneracy-bounded graphs), and the empirical results of Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses its limitations both through its underlying assumptions (e.g., the requirement of a publicly known graph topology) and through the observed performance characteristics of the proposed algorithms. In particular, the limitations of the unbiased and biased variants are explicitly reflected in the theoretical bounds and are further validated empirically in experiments comparing dense and sparse graph settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions of the setting are defined in Section 2. For all theoretical results the complete proofs are provided either directly in the main section or the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experimental settings, including parameters and datasets, are clearly specified. The code and data are provided in the supplementary material and via a link to a public repository.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is supplied in the supplemental material. The code repository includes the data and clear instructions for running the experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All parameters (e.g. privacy budget, threshold) used are stated together with a reason for their choice for each experiment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The results were stable across all test settings, so statistical significance analysis was not provided.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Section 6 the resource is introduced as an HPC cluster (CPU) with details about the core and memory usage of the experiments. The resources for the experiments on the assignment function are given in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the Code of Ethics. All experiments are based on publicly available anonymized data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The paper focuses on theoretical contributions to privacy-preserving algorithms and does not target a specific application domain. The authors do not expect any immediate negative societal impacts of the work performed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The research does not release any data or models with high risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The sources of all datasets (SNAP Google Web Graph, ML-Tele-400, musae-wiki, GMWCS) used are properly cited in Section 6 and Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper provides a repository that includes the code and derived artifacts used in the experiments, including the constructed graph data (ML-Tele-400). The repository contains documentation describing how the graph is generated from the underlying public dataset, along with instructions for reproducing the preprocessing steps. The license is included alongside the graph and in accordance with the underlying used data.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not introduce any crowdsourcing experiments with human subjects. All datasets are public and any data referring to humans was anonymized according to research standards.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not involve as an important, original, or non-standard component of this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.