Locally Differentially Private Analysis of Graph Statistics

Jacob Imola*
UC San Diego

Takao Murakami*
AIST

Kamalika Chaudhuri
UC San Diego

Abstract

Differentially private analysis of graphs is widely used for releasing statistics from sensitive graphs while still preserving user privacy. Most existing algorithms however are in a centralized privacy model, where a trusted data curator holds the entire graph. As this model raises a number of privacy and security issues – such as, the trustworthiness of the curator and the possibility of data breaches, it is desirable to consider algorithms in a more decentralized local model where no server holds the entire graph.

In this work, we consider a local model, and present algorithms for counting subgraphs – a fundamental task for analyzing the connection patterns in a graph – with LDP (Local Differential Privacy). For triangle counts, we present algorithms that use one and two rounds of interaction, and show that an additional round can significantly improve the utility. For k-star counts, we present an algorithm that achieves an order optimal estimation error in the non-interactive local model. We provide new lower-bounds on the estimation error for general graph statistics including triangle counts and k-star counts. Finally, we perform extensive experiments on two real datasets, and show that it is indeed possible to accurately estimate subgraph counts in the local differential privacy model.

1 Introduction

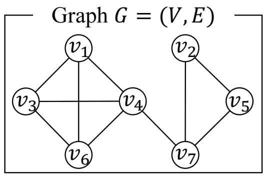
Analysis of network statistics is a useful tool for finding meaningful patterns in graph data, such as social, e-mail, citation and epidemiological networks. For example, the average *degree* (i.e., number of edges connected to a node) in a social graph can reveal the average connectivity. *Subgraph counts* (e.g., the number of triangles, stars, or cliques) can be used to measure centrality properties such as the *clustering coefficient*, which represents the probability that two friends of an individual will also be friends of one another [41]. However, the vast majority of graph analytics is carried out on sensitive data, which could be leaked through the results of graph analysis. Thus, there is a need to develop solutions that can analyze these graph properties while still preserving the privacy of individuals in the network.

The standard way to analyze graphs with privacy is through differentially private graph analysis [22, 23, 49]. Differential privacy provides individual privacy against adversaries with arbitrary background knowledge, and has currently emerged as the gold standard for private analytics. However, a vast majority of differentially private graph analysis algorithms are in the *centralized (or global) model* [13, 15, 16, 27, 34, 36, 42, 48, 49, 52, 58, 59], where a single trusted data curator holds the entire graph and releases sanitized versions of the statistics. By assuming a trusted party that can access the entire graph, it is possible to release accurate graph statistics (e.g., subgraph counts [34, 36, 52], degree distribution [16, 27, 48], spectra [59]) and synthetic graphs [15, 58].

In many applications however, a single trusted curator may not be practicable due to security or logistical reasons. A centralized data holder is amenable to security issues such as data breaches and leaks – a growing threat in recent years [39, 51]. Additionally, *decentralized social networks* [43, 50] (e.g., Diaspora [5]) have no central server that contains an entire social graph, and use instead many servers all over the world, each containing the data of users who have chosen to register there. Finally, a centralized solution is also not applicable to fully decentralized applications, where the server does not automatically hold information connecting users. An example of this is a mobile application that asks each user how many of her friends she has seen today, and sends noisy counts to a central server. In this application, the server does not hold any individual edge, but can still aggregate the responses to determine the average mobility in an area.

The standard privacy solution that does not assume a trusted third party is LDP (Local Differential Privacy) [20, 35]. This is a special case of DP (Differential Privacy) in the *local model*, where each user obfuscates her personal data by herself and sends the obfuscated data to a (possibly malicious) data collector. Since the data collector does not hold the original

*The first and second authors made equal contributions.



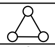
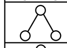
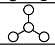
Shape	Name	Count
	Triangle	5
	2-star	20
	3-star	8

Figure 1: Example of subgraph counts.

personal data, it does not suffer from data leakage issues. Therefore, LDP has recently attracted attention from both academia [8,10,11,24,32,33,40,45,57,62] as well as industry [17,55,56]. However, the use of LDP has mostly been in the context of tabular data where each row corresponds to an individual, and little attention has been paid to LDP for more complex data such as graphs (see Section 2 for details).

In this paper, we consider LDP for graph data, and provide algorithms and theoretical performance guarantees for calculating graph statistics in this model. In particular, we focus on counting triangles and k-stars – the most basic and useful subgraphs. A triangle is a set of three nodes with three edges (we exclude automorphisms; i.e., #closed triplets $= 3\times$ #triangles). A k-star consists of a central node connected to k other nodes. Figure 1 shows an example of triangles and k-stars. Counting them is a fundamental task of analyzing the connection patterns in a graph, as the clustering coefficient can be calculated from triangle and 2-star counts as: $\frac{3\times\#\text{triangles}}{\#\text{2-stars}}$ (in Figure 1, $\frac{3\times 5}{20} = 0.75$).

When we look to protect privacy of relationship information modeled by edges in a graph, we need to pay attention to the fact that some relationship information could be leaked from subgraph counts. For example, suppose that user (node) v_2 in Figure 1 knows all edges connected to v_2 and all edges between $v_3, \dots, v_7$ as background knowledge, and that v_2 wants to know who are friends with v_1. Then "#2-stars = 20" reveals the fact that v_1 has three friends, and "#triangles = 5" reveals the fact that the three friends of v_1 are v_3, v_4, and v_6. Moreover, a central server that holds all friendship information (i.e., all edges) may face data breaches, as explained above. Therefore, a private algorithm for counting subgraphs in the local model is highly beneficial to individual privacy.

The main challenge in counting subgraphs in the local model is that existing techniques and their analysis do not directly apply. The existing work on LDP for tabular data assumes that each person's data is independently and identically drawn from an underlying distribution. In graphs, this is no longer the case; e.g., each triangle is not independent, because multiple triangles can involve the same edge; each k-star is not independent for the same reason. Moreover, complex inter-dependencies involving multiple people are possible in graphs. For example, each user cannot count triangles involving herself, because she cannot see edges between other users; e.g., user v_1 cannot see an edge between v_3 and v_4 in Figure 1.

We show that although these complex dependency among users introduces challenges, it also presents opportunities. Specifically, this kind of interdependency also implies that extra interaction between users and a data collector may be helpful depending on the prior responses. In this work, we investigate this issue and provide algorithms for accurately calculating subgraph counts under LDP.

Our contributions. In this paper, we provide algorithms and corresponding performance guarantees for counting triangles and k-stars in graphs under edge Local Differential Privacy. Specifically, our contributions are as follows:

- For triangles, we present two algorithms. The first is based on Warner's RR (Randomized Response) [60] and empirical estimation [32,40,57]. We then present a more sophisticated algorithm that uses an additional round of interaction between users and data collector. We provide upper-bounds on the estimation error for each algorithm, and show that the latter can significantly reduce the estimation error.

- For k-stars, we present a simple algorithm using the Laplacian mechanism. We analyze the upper-bound on the estimation error for this algorithm, and show that it is order optimal in terms of the number of users among all LDP mechanisms that do not use additional interaction.

- We provide lower-bounds on the estimation error for general graph functions including triangle counts and k-star counts in the local model. These are stronger than known upper bounds in the centralized model, and illustrate the limitations of the local model over the central.

- Finally, we evaluate our algorithms on two real datasets, and show that it is indeed possible to accurately estimate subgraph counts in the local model. In particular, we show that the interactive algorithm for triangle counts and the Laplacian algorithm for the k-stars provide small estimation errors when the number of users is large.

We implemented our algorithms with C/C++, and published them as open-source software [1].

2 Related Work

Graph DP. DP on graphs has been widely studied, with most prior work being in the centralized model [13,15,16,27,34,36,42,48,49,52,58,59]. In this model, a number of algorithms have been proposed for releasing subgraph counts [34,36,52], degree distributions [16,27,48], eigenvalues and eigenvectors [59], and synthetic graphs [15,58].

There has also been a handful of work on graph algorithms in the local DP model [46,53,63–65]. For example, Qin *et al.* [46] propose an algorithm for generating synthetic graphs. Zhang *et al.* [65] propose an algorithm for software usage analysis under LDP, where a node represents a software component (e.g., function in a code) and an edge represents a control-flow between components. Neither of these works focus on subgraph counts.

Sun *et al.* [53] propose an algorithm for counting subgraphs in the local model under the assumption that each user allows her friends to see all her connections. However, this assumption does not hold in many practical scenarios; e.g., a Facebook user can change her setting so that friends cannot see her connections. Therefore, we assume that each user knows only her friends rather than all of her friends' friends. The algorithms in [53] cannot be applied to this setting.

Ye *et al.* [63] propose a one-round algorithm for estimating graph metrics including the clustering coefficient. Here they apply Warner's RR (Randomized Response) to an adjacency matrix. However, it introduces a very large bias for triangle counts. In [64], they propose a method for reducing the bias in the estimate of triangle counts. However, the method in [64] introduces some approximation, and it is unclear whether their estimate is unbiased. In this paper, we propose a one-round algorithm for triangles that uses empirical estimation as a post-processing step, and prove that our estimate is unbiased. We also show in Appendix A that our one-round algorithm significantly outperforms the one-round algorithm in [63]. Moreover, we show in Section 5 that our two-rounds algorithm significantly outperforms our one-round algorithm.

Our work also differs from [53, 63, 64] in that we provide lower-bounds on the estimation error.

LDP. Apart from graphs, a number of works have looked at analyzing statistics (e.g., discrete distribution estimation [8, 24, 32, 33, 40, 57, 62], heavy hitters [10, 11, 45]) under LDP.

However, they use LDP in the context of tabular data, and do not consider the kind of complex interdependency in graph data (as described in Section 1). For example, the RR with empirical estimation is optimal in the low privacy regimes for estimating a distribution for tabular data [32, 33]. We apply the RR and empirical estimation to counting triangles, and show that it is suboptimal and significantly outperformed by a more sophisticated two-rounds algorithm.

Upper/lower-bounds. Finally, we note that existing work on upper-bounds and lower-bounds cannot be directly applied to our setting. For example, there are upper-bounds [8, 29, 30, 32, 33, 62] and lower-bounds [7, 18, 19, 21, 29–31] on the estimation error (or sample complexity) in distribution estimation of tabular data. However, they assume that each original data value is independently sampled from an underlying distribution. They cannot be directly applied to our graph setting, because each triangle and each k-star involve multiple edges and are not independent (as described in Section 1). Rashtchian *et al.* [47] provide lower-bounds on communication complexity (i.e., number of queries) of vector-matrix-vector queries for estimating subgraph counts. However, their lower-bounds are not on the estimation error, and cannot be applied to our problem.

3 Preliminaries

3.1 Graphs and Differential Privacy

Graphs. Let $\mathbb{N}$, $\mathbb{Z}_{\geq 0}$, $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$ be the sets of natural numbers, non-negative integers, real numbers, and non-negative real numbers, respectively. For $a \in \mathbb{N}$, let $[a] = \{1, 2, \dots, a\}$.

We consider an undirected graph $G = (V, E)$, where V is a set of nodes (i.e., users) and E is a set of edges. Let $n \in \mathbb{N}$ be the number of users in V, and let $v_i \in V$ the i-th user; i.e., $V = \{v_1, \dots, v_n\}$. An edge $(v_i, v_j) \in E$ represents a relationship between users $v_i \in V$ and $v_j \in V$. The number of edges connected to a single node is called the *degree* of the node. Let $d_{max} \in \mathbb{N}$ be the *maximum degree* (i.e., maximum number of edges connected to a node) in graph G. Let $\mathcal{G}$ be the set of possible graphs G on n users. A graph $G \in \mathcal{G}$ can be represented as a symmetric adjacency matrix $\mathbf{A} = (a_{i,j}) \in \{0,1\}^{n \times n}$, where $a_{i,j} = 1$ if $(v_i, v_j) \in E$ and $a_{i,j} = 0$ otherwise.

Types of DP. DP (Differential Privacy) [22, 23] is known as a gold standard for data privacy. According to the underlying architecture, DP can be divided into two types: *centralized DP* and *LDP (Local DP)*. Centralized DP assumes the centralized model, where a "trusted" data collector collects the original personal data from all users and obfuscates a query (e.g., counting query, histogram query) on the set of personal data. LDP assumes the local model, where each user does not trust even the data collector. In this model, each user obfuscates a query on her personal data by herself and sends the obfuscated data to the data collector.

If the data are represented as a graph, we can consider two types of DP: *edge DP* and *node DP* [27, 49]. Edge DP considers two neighboring graphs $G, G' \in \mathcal{G}$ that differ in one edge. In contrast, node DP considers two neighboring graphs $G, G' \in \mathcal{G}$ in which G' is obtained from G by adding or removing one node along with its adjacent edges.

Although Zhang *et al.* [65] consider node DP in the local model where each node represents a software component, we consider a totally different problem where each node represents a user. In the latter case, node DP requires us to hide the *existence of each user* along with her all edges. However, many applications in the local model send the identity of each user to a server. For example, we can consider a mobile application that sends to a server how many friends a user met today along with her user ID. In this case, the user may not mind sending her user ID, but may want to hide her edge information (i.e., who she met today). Although we cannot use node DP in such applications, we can use edge DP to deny the presence/absence of each edge (friend). Thus we focus on edge DP in the same way as [46, 53, 63, 64].

Below we explain edge DP in the centralized model.

Centralized DP. We call edge DP in the centralized model *edge centralized DP*. Formally, it is defined as follows:

Definition 1 (ε-edge centralized DP)**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. A randomized algorithm $\mathcal{M}$ with domain $\mathcal{G}$ provides* ε-edge centralized DP *if for any two neighboring graphs $G, G' \in \mathcal{G}$ that differ in one edge and any $S \subseteq \mathrm{Range}(\mathcal{M})$,*

$$\Pr[\mathcal{M}(G) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(G') \in S]. \tag{1}$$

Edge centralized DP guarantees that an adversary who has observed the output of $\mathcal{M}$ cannot determine whether it is come from G or G' with a certain degree of confidence. The parameter ε is called the *privacy budget*. If ε is close to zero, then G and G' are almost equally likely, which means that an edge in G is strongly protected.

We also note that edge DP can be used to protect $k \in \mathbb{N}$ edges by using the notion of group privacy [23]. Specifically, if $\mathcal{M}$ provides ε-edge centralized DP, then for any two graphs $G, G' \in \mathcal{G}$ that differ in k edges and any $S \subseteq \mathrm{Range}(\mathcal{M})$, we obtain: $\Pr[\mathcal{M}(G) \in S] \leq e^{k\varepsilon} \Pr[\mathcal{M}(G') \in S]$; i.e., k edges are protected with privacy budget $k\varepsilon$.

3.2 Local Differential Privacy

LDP (Local Differential Privacy) [20, 35] is a privacy metric to protect personal data of each user in the local model. LDP has been originally introduced to protect each user's data record that is independent from the other records. However, in a graph, each edge is connected to two users. Thus, when we define edge DP in the local model, we should consider what we want to protect. In this paper, we consider two definitions of edge DP in the local model: *edge LDP* in [46] and *relationship DP* introduced in this paper. Below, we will explain these two definitions in detail.

Edge LDP. Qin *et al.* [46] defined edge LDP based on a user's *neighbor list*. Specifically, let $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,n}) \in \{0,1\}^n$ be a neighbor list of user v_i. Note that $\mathbf{a}_i$ is the i-th row of the adjacency matrix $\mathbf{A}$ of graph G. In other words, graph G can be represented as neighbor lists $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

Then edge LDP is defined as follows:

Definition 2 (ε-edge LDP [46])**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. For any $i \in [n]$, let $\mathcal{R}_i$ with domain $\{0,1\}^n$ be a randomized algorithm of user v_i. $\mathcal{R}_i$ provides* ε-edge LDP *if for any two neighbor lists $\mathbf{a}_i, \mathbf{a}'_i \in \{0,1\}^n$ that differ in one bit and any $S \subseteq \mathrm{Range}(\mathcal{R}_i)$,*

$$\Pr[\mathcal{R}_i(\mathbf{a}_i) \in S] \leq e^{\varepsilon} \Pr[\mathcal{R}_i(\mathbf{a}'_i) \in S]. \tag{2}$$

Edge LDP in Definition 2 protects a single bit in a neighbor list with privacy budget ε. As with edge centralized DP, edge LDP can also be used to protect $k \in \mathbb{N}$ bits in a neighbor list by using group privacy; i.e., k bits in a neighbor list are protected with privacy budget $k\varepsilon$.

RR (Randomized Response). As a simple example of a randomized algorithm $\mathcal{R}_i$ providing ε-edge LDP, we explain Warner's RR (Randomized Response) [60] applied to a neighbor list, which is called the randomized neighbor list in [46].

Given a neighbor list $\mathbf{a}_i \in \{0,1\}^n$, this algorithm outputs a noisy neighbor lists $\mathbf{b} = (b_1, \ldots, b_n) \in \{0,1\}^n$ by flipping each bit in $\mathbf{a}_i$ with probability $p = \frac{1}{e^{\varepsilon}+1}$; i.e., for each $j \in [n]$, $b_j \neq a_{i,j}$ with probability p and $b_j = a_{i,j}$ with probability $1-p$. Since $\Pr[\mathcal{R}_i(\mathbf{a}_i) \in S]$ and $\Pr[\mathcal{R}_i(\mathbf{a}'_i) \in S]$ in (2) differ by e^{ε} for $\mathbf{a}_i$ and $\mathbf{a}'_i$ that differ in one bit, this algorithm provides ε-edge LDP.

Relationship DP. In graphs such as social networks, it is usually the case that two users share knowledge of the presence of an edge between them. To hide their mutual edge, we must consider that both user's outputs can leak information. We introduce a DP definition called relationship DP that hides *one entire edge in graph G during the whole process:*

Definition 3 (ε-relationship DP)**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. A tuple of randomized algorithms $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$, each of which is with domain $\{0,1\}^n$, provides* ε-relationship DP *if for any two neighboring graphs $G, G' \in \mathcal{G}$ that differ in one edge and any $S \subseteq \mathrm{Range}(\mathcal{R}_1) \times \ldots \times \mathrm{Range}(\mathcal{R}_n)$,*

$$\begin{aligned} &\Pr[(\mathcal{R}_1(\mathbf{a}_1), \ldots, \mathcal{R}_n(\mathbf{a}_n)) \in S] \\ &\leq e^{\varepsilon} \Pr[(\mathcal{R}_1(\mathbf{a}'_1), \ldots, \mathcal{R}_n(\mathbf{a}'_n)) \in S], \end{aligned} \tag{3}$$

where $\mathbf{a}_i$ (resp. $\mathbf{a}'_i$) $\in \{0,1\}^n$ is the i-th row of the adjacency matrix of graph G (resp. G').

Relationship DP is the same as *decentralized DP* in [53] except that the former (resp. latter) assumes that each user knows only her friends (resp. all of her friends' friends).

Edge LDP assumes that user v_i's edge connected to user v_j and user v_j's edge connected to user v_i are different secrets, with user v_i knowing the former and user v_j knowing the latter. Relationship DP assumes that the two secrets are the same.

Note that the threat model of relationship DP is different from that of LDP – some amount of trust must be given to the other users in relationship DP. Specifically, user v_i must trust user v_j to not leak information about their shared edge. If $k \in \mathbb{N}$ users decide not to follow their protocols, then up to k edges incident to user v_i may be compromised. This trust model is stronger than LDP, which assumes nothing about what other users do, but is much weaker than centralized DP in which all edges are in the hands of the central party.

Other than the differing threat models, relationship DP and edge LDP are quite closely related:

Proposition 1. *If randomized algorithms $\mathcal{R}_1, \ldots, \mathcal{R}_n$ provide ε-edge LDP, then $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ provides 2ε-relationship DP.*

Proof. The existence of edge $(v_i, v_j) \in E$ affects two elements $a_{i,j}, a_{j,i} \in \{0,1\}$ in the adjacency matrix $\mathbf{A}$. Then by group privacy [23], Proposition 1 holds. □

Proposition 1 states that when we want to protect one edge as a whole, the privacy budget is at most doubled. Note, however, that some randomized algorithms do not have this doubling issue. For example, we can apply the RR to the i-th neighbor list $\mathbf{a}_i$ so that $\mathcal{R}_i$ outputs noisy bits $(b_1, \ldots, b_{i-1}) \in \{0,1\}^{i-1}$ for only users $v_1, \ldots, v_{i-1}$ with smaller user IDs; i.e., for each $j \in \{1, \ldots, i-1\}$, $b_j \neq a_{i,j}$ with probability $p = \frac{1}{e^\varepsilon+1}$ and $b_j = a_{i,j}$ with probability $1-p$. In other words, we can extend the RR for a neighbor list so that $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ outputs only the lower triangular part of the noisy adjacency matrix. Then all of $\mathcal{R}_1, \ldots, \mathcal{R}_n$ provide ε-edge LDP. In addition, the existence of edge $(v_i, v_j) \in E$ $(i > j)$ affects only one element $a_{i,j}$ in the lower triangular part of $\mathbf{A}$. Thus, $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ provides ε-relationship DP (not 2ε).

Our proposed algorithm in Section 4.3 also has this property; i.e., it provides both ε-edge LDP and ε-relationship DP.

3.3 Global Sensitivity

In this paper, we use the notion of global sensitivity [23] to provide edge centralized DP or edge LDP.

Let $\mathcal{D}$ be the set of possible input data of a randomized algorithm. In edge centralized DP, $\mathcal{D} = \mathcal{G}$. In edge LDP, $\mathcal{D} = \{0,1\}^n$. Let $f : \mathcal{D} \to \mathbb{R}$ be a function that takes data $D \in \mathcal{D}$ as input and outputs some statistics $f(D) \in \mathbb{R}$ about the data. The most basic method for providing DP is to add the Laplacian noise proportional to the global sensitivity [23].

Definition 4 (Global sensitivity)**.** *The global sensitivity of a function $f : \mathcal{D} \to \mathbb{R}$ is given by:*

$$GS_f = \max_{D, D' \in \mathcal{D} : D \sim D'} |f(D) - f(D')|,$$

where $D \sim D'$ represents that D and D' are neighbors; i.e., they differ in one edge in edge centralized DP, and differ in one bit in edge LDP.

In graphs, the global sensitivity GS_f can be very large. For example, adding one edge may result in the increase of triangle (resp. k-star) counts by $n-2$ (resp. $\binom{n}{k-1}$).

One way to significantly reduce the global sensitivity is to use *graph projection* [16, 36, 48], which removes some neighbors from a neighbor list so that the maximum degree d_{max} is upper-bounded by a predetermined value $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$. By using the graph projection with $\tilde{d}_{max} \ll n$, we can enforce small global sensitivity; e.g., the global sensitivity of triangle (resp. k-star) counts is at most $\tilde{d}_{max}$ (resp. $\binom{\tilde{d}_{max}}{k-1}$) after the projection.

Ideally, we would like to set $\tilde{d}_{max} = d_{max}$ to avoid removing neighbors from a neighbor list (i.e., to avoid the loss of utility). However, the maximum degree d_{max} can leak some information about the original graph G. In this paper, we address this issue by privately estimating d_{max} with edge LDP and then using the private estimate of d_{max} as $\tilde{d}_{max}$. This technique is also known as *adaptive clipping* in differentially private stochastic gradient descent (SGD) [44, 54].

Table 1: Basic notations in this paper.

Symbol	Description
n	Number of users.
$G = (V, E)$	Graph with n nodes (users) V and edges E.
v_i	i-th user in V.
d_{max}	Maximum degree of G.
$\tilde{d}_{max}$	Upper-bound on d_{max} (used for projection).
$\mathcal{G}$	Set of possible graphs on n users.
$\mathbf{A} = (a_{i,j})$	Adjacency matrix.
$\mathbf{a}_i$	i-th row of $\mathbf{A}$ (i.e., neighbor list of v_i).
$\mathcal{R}_i$	Randomized algorithm on $\mathbf{a}_i$.
$f_\triangle(G)$	Number of triangles in G.
$f_{k\star}(G)$	Number of k-stars in G.

3.4 Graph Statistics and Utility Metrics

Graph statistics. We consider a graph function that takes a graph $G \in \mathcal{G}$ as input and outputs some graph statistics. Specifically, let $f_\triangle : \mathcal{G} \to \mathbb{Z}_{\geq 0}$ be a graph function that outputs the number of triangles in G. For $k \in \mathbb{N}$, let $f_{k\star} : \mathcal{G} \to \mathbb{Z}_{\geq 0}$ be a graph function that outputs the number of k-stars in G. For example, if a graph G is as shown in Figure 1, then $f_\triangle(G) = 5$, $f_{2\star}(G) = 20$, and $f_{3\star}(G) = 8$. The clustering coefficient can also be calculated from $f_\triangle(G)$ and $f_{2\star}(G)$ as: $\frac{3f_\triangle(G)}{f_{2\star}(G)} = 0.75$.

Table 1 shows the basic notations used in this paper.

Utility metrics. We use the l_2 loss (i.e., squared error) [32, 40, 57] and the relative error [12, 14, 61] as utility metrics.

Specifically, let $\hat{f}(G) \in \mathbb{R}$ be an estimate of graph statistics $f(G) \in \mathbb{R}$. Here f can be instantiated by $f_\triangle$ or $f_{k\star}$; i.e., $\hat{f}_\triangle(G)$ and $\hat{f}_{k\star}(G)$ are the estimates of $f_\triangle(G)$ and $f_{k\star}(G)$, respectively. Let l_2^2 be the l_2 loss function, which maps the estimate $\hat{f}(G)$ and the true value $f(G)$ to the l_2 loss; i.e., $l_2^2(\hat{f}(G), f(G)) = (\hat{f}(G) - f(G))^2$. Note that when we use a randomized algorithm providing edge LDP (or edge centralized DP), $\hat{f}(G)$ depends on the randomness in the algorithm. In our theoretical analysis, we analyze the expectation of the l_2 loss over the randomness, as with [32, 40, 57].

When $f(G)$ is large, the l_2 loss can also be large. Thus in our experiments, we also evaluate the relative error, along with the l_2 loss. The relative error is defined as: $\frac{|\hat{f}(G) - f(G)|}{\max\{f(G), \eta\}}$, where $\eta \in \mathbb{R}_{\geq 0}$ is a very small positive value. Following the convention [12, 14, 61], we set $\eta = 0.001n$ for $f_\triangle$ and $f_{k\star}$.

4 Algorithms

In the local model, there are several ways to model how the data collector interacts with the users [20, 31, 46]. The simplest model would be to assume that the data collector sends a query $\mathcal{R}_i$ to each user v_i once, and then each user v_i independently sends an answer $\mathcal{R}_i(\mathbf{a}_i)$ to the data collector. In this model, there is one-round interaction between each user and the data collector. We call this the *one-round LDP model*. For example, the RR for a neighbor list in Section 3.2 assumes this model.

However, in certain cases it may be possible for the data collector to send a query to each user multiple times. This could allow for more powerful queries that result in more accurate subgraph counts [53] or more accurate synthetic graphs [46]. We call this the *multiple-rounds LDP model*.

In Sections 4.1 and 4.2, we consider the problems of computing $f_{k\star}(G)$ and $f_{\triangle}(G)$ for a graph $G \in \mathcal{G}$ in the one-round LDP model. Our algorithms and bounds highlight limitations of the one-round LDP model. Compared to the centralized graph DP model, the one-round LDP model cannot compute $f_{k\star}(G)$ as accurately. Furthermore, the algorithm for $f_{\triangle}(G)$ does not perform well. In Section 4.3, we propose a more sophisticated algorithm for computing $f_{\triangle}(G)$ in the two-rounds LDP model, and show that it provides much smaller expected l_2 loss than the algorithm in the one-round LDP model. In Section 4.4, we show a general result about lower bounds on the expected l_2 loss of graph statistics in LDP. The proofs of all statements in Section 4 are given in the full version [28].

4.1 One-Round Algorithms for k-Stars

Algorithm. We begin with the problem of computing $f_{k\star}(G)$ in the one-round LDP model. For this model, we introduce a simple algorithm using the Laplacian mechanism, and prove that this algorithm can achieve order optimal expected l_2 loss among all one-round LDP algorithms.

```
Data: Graph $G$ represented as neighbor lists $\mathbf{a}_1, \ldots, \mathbf{a}_n$
      $\in \{0,1\}^n$, privacy budget $\varepsilon \in \mathbb{R}_{\geq 0}$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$.
Result: Private estimate of $f_{k\star}(G)$.
1 $\Delta \leftarrow \binom{\tilde{d}_{max}}{k-1}$;
2 for $i = 1$ to $n$ do
3     $\mathbf{a}_i \leftarrow$ GraphProjection($\mathbf{a}_i, \tilde{d}_{max}$);
      /* $d_i$ is a degree of user $v_i$.            */
4     $d_i \leftarrow \sum_{j=1}^{n} a_{i,j}$;
5     $r_i \leftarrow \binom{d_i}{k}$;
6     $\hat{r}_i \leftarrow r_i + \mathrm{Lap}\left(\frac{\Delta}{\varepsilon}\right)$;
7     *release*($\hat{r}_i$);
8 end
9 return $\sum_{i=1}^{n} \hat{r}_i$
```

Algorithm 1: LocalLap$_{k\star}$

Algorithm 1 shows the one-round algorithm for k-stars. It takes as input a graph G (represented as neighbor lists $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \{0,1\}^n$), the privacy budget ε, and a non-negative integer $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$.

The parameter $\tilde{d}_{max}$ plays a role as an upper-bound on the maximum degree d_{max} of G. Specifically, let $d_i \in \mathbb{Z}_{\geq 0}$ be the degree of user v_i; i.e., the number of "1"s in her neighbor list $\mathbf{a}_i$. In line 3, user v_i uses a function (denoted by `GraphProjection`) that performs graph projection [16, 36, 48] for $\mathbf{a}_i$ as follows. If d_i exceeds $\tilde{d}_{max}$, it randomly selects $\tilde{d}_{max}$ neighbors out of d_i neighbors; otherwise, it uses $\mathbf{a}_i$ as it is. This guarantees that each user's degree never exceeds $\tilde{d}_{max}$; i.e., $d_i \leq \tilde{d}_{max}$ after line 4.

After the graph projection, user v_i counts the number of k-stars $r_i \in \mathbb{Z}_{\geq 0}$ of which she is a center (line 5), and adds the Laplacian noise to r_i (line 6). Here, since adding one edge results in the increase of at most $\binom{\tilde{d}_{max}}{k-1}$ k-stars, the sensitivity of k-star counts for user v_i is at most $\binom{\tilde{d}_{max}}{k-1}$ (after graph projection). Therefore, we add $\mathrm{Lap}(\frac{\Delta}{\varepsilon})$ to r_i, where $\Delta = \binom{\tilde{d}_{max}}{k-1}$ and for $b \in \mathbb{R}_{\geq 0}$ $\mathrm{Lap}(b)$ is a random variable that represents the Laplacian noise with mean 0 and scale b. The final answer of Algorithm 1 is simply the sum of all the noisy k-star counts. We denote this algorithm by LocalLap$_{k\star}$.

The value of $\tilde{d}_{max}$ greatly affects the utility. If $\tilde{d}_{max}$ is too large, a large amount of the Laplacian noise would be added. If $\tilde{d}_{max}$ is too small, a great number of neighbors would be reduced by graph projection. When we have some prior knowledge about the maximum degree d_{max}, we can set $\tilde{d}_{max}$ to an appropriate value. For example, the maximum number of connections allowed on Facebook is 5000 [3]. In this case, we can set $\tilde{d}_{max} = 5000$, and then graph projection does nothing. Given that the number of Facebook monthly active users is over 2.7 billion [6], $\tilde{d}_{max} = 5000$ is much smaller than n. For another example, if we know that the degree is smaller than 1000 for most users, then we can set $\tilde{d}_{max} = 1000$ and perform graph projection for users whose degrees exceed $\tilde{d}_{max}$.

In some applications, the data collector may not have such prior knowledge about $\tilde{d}_{max}$. In this case, we can privately estimate d_{max} by allowing an additional round between each user and the data collector, and use the private estimate of d_{max} as $\tilde{d}_{max}$. We describe how to privately estimate d_{max} with edge LDP at the end of Section 4.1.

Theoretical properties. LocalLap$_{k\star}$ has the following guarantees:

Theorem 1. *LocalLap$_{k\star}$ provides ε-edge LDP.*

Theorem 2. *Let $\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max})$ be the output of LocalLap$_{k\star}$. Then, for all $k \in \mathbb{N}, \varepsilon \in \mathbb{R}_{\geq 0}, \tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, and $G \in \mathcal{G}$ such that the maximum degree d_{max} of G is at most $\tilde{d}_{max}$, $\mathbb{E}[l_2^2(\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max}), f_{k\star}(G))] = O\left(\frac{n\tilde{d}_{max}^{2k-2}}{\varepsilon^2}\right)$.*

The factor of n in the expected l_2 loss of LocalLap$_{k\star}$ comes from the fact that we are adding the Laplacian noise n times. In the centralized model, this factor of n is not there, because the central data collector sees all k-stars; i.e., the data collector knows $f_{k\star}(G)$. The sensitivity of $f_{k\star}$ is at most $2\binom{\tilde{d}_{max}}{k-1}$ (after graph projection) under edge centralized DP. Therefore, we can consider an algorithm that simply adds the Laplacian noise $\mathrm{Lap}(2\binom{\tilde{d}_{max}}{k-1}/\varepsilon)$ to $f_{k\star}(G)$, and outputs $f_{k\star}(G) + \mathrm{Lap}(2\binom{\tilde{d}_{max}}{k-1}/\varepsilon)$. We denote this algorithm by CentralLap$_{k\star}$. Since the bias of the Laplacian noise is

0, $\mathsf{CentralLap}_{k\star}$ attains the expected l_2 loss (= variance) of $O\left(\frac{\tilde{d}_{max}^{2k-2}}{\varepsilon^2}\right)$.

It seems impossible to avoid this factor of n in the one-round LDP model, as the data collector will be dealing with n independent answers to queries. Indeed, this is the case—we prove that the expected l_2 error of $\mathsf{LocalLap}_{k\star}$ is order optimal among all one-round LDP algorithms, and the one-round LDP model cannot improve the factor of n.

Corollary 1. *Let $\hat{f}_{k\star}(G,\tilde{d}_{max},\varepsilon)$ be any one-round LDP algorithm that computes $f_{k\star}(G)$ satisfying ε-edge LDP. Then, for all $k,n,\tilde{d}_{max} \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}_{\geq 0}$ such that n is even, there exists a set of graphs $\mathcal{A} \subseteq \mathcal{G}$ on n nodes such that the maximum degree of each $G \in \mathcal{A}$ is at most $\tilde{d}_{max}$, and $\frac{1}{|\mathcal{A}|}\sum_{G\in\mathcal{A}} \mathbb{E}[l_2^2(\hat{f}_{k\star}(G,\tilde{d}_{max},\varepsilon), f_{k\star}(G))] \geq \Omega\left(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2}\tilde{d}_{max}^{2k-2}n\right)$.*

This is a corollary of a more general result of Section 4.4. Thus, any algorithm computing k-stars cannot avoid the factor of n in its l_2^2 loss. k-stars is an example where the non-interactive graph LDP model is strictly weaker than the centralized DP model.

Nevertheless, we note that $\mathsf{LocalLap}_{k\star}$ can accurately calculate $f_{k\star}(G)$ for a large number of users n. Specifically, the relative error decreases with increase in n because $\mathsf{LocalLap}_{k\star}$ has a factor of n (not n^2) in the expected l_2 error, i.e., $\mathbb{E}[(\hat{f}_{k\star}(G,\varepsilon,\tilde{d}_{max}) - f_{k\star}(G))^2] = O(n)$ and $f_{k\star}(G)^2 \geq \Omega(n^2)$ (when we ignore $\tilde{d}_{max}$ and ε). In our experiments, we show that the relative error of $\mathsf{LocalLap}_{k\star}$ is small when n is large.

Private calculation of d_{max}. By allowing an additional round between each user and the data collector, we can privately estimate d_{max} and use the private estimate of d_{max} as $\tilde{d}_{max}$. Specifically, we divide the privacy budget ε into $\varepsilon_0 \in \mathbb{R}_{\geq 0}$ and $\varepsilon_1 \in \mathbb{R}_{\geq 0}$; i.e., $\varepsilon = \varepsilon_0 + \varepsilon_1$. We first estimate d_{max} with ε_0-edge LDP and then run $\mathsf{LocalLap}_{k\star}$ with the remaining privacy budget ε_1. Note that $\mathsf{LocalLap}_{k\star}$ with the private calculation of d_{max} results in a two-rounds LDP algorithm.

We consider the following simple algorithm. At the first round, each user v_i adds the Laplacian noise $\mathrm{Lap}(\frac{1}{\varepsilon_0})$ to her degree d_i. Let $\hat{d}_i \in \mathbb{R}$ be the noisy degree of v_i; i.e., $\hat{d}_i = d_i + \mathrm{Lap}(\frac{1}{\varepsilon_0})$. Then user v_i sends $\hat{d}_i$ to the data collector. Let $\hat{d}_{max} \in \mathbb{R}$ be the maximum value of the noisy degree; i.e., $\hat{d}_{max} = \max\{\hat{d}_1,\ldots,\hat{d}_n\}$. We call $\hat{d}_{max}$ the *noisy max degree*. The data collector calculates the noisy max degree $\hat{d}_{max}$ as an estimate of d_{max}, and sends $\hat{d}_{max}$ back to all users. At the second round, we run $\mathsf{LocalLap}_{k\star}$ with input G, ε, and $\lfloor \hat{d}_{max} \rfloor$.

At the first round, the calculation of $\hat{d}_{max}$ provides ε_0-edge LDP because each user's degree has the sensitivity 1 under edge LDP. At the second round, Theorem 1 guarantees that $\mathsf{LocalLap}_{k\star}$ provides ε_1-edge LDP. Then by the composition theorem [23], this two-rounds algorithm provides ε-edge LDP in total ($\varepsilon = \varepsilon_0 + \varepsilon_1$).

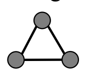
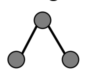
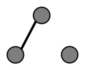
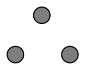
Shape	triangle 	2-edges 	1-edge 	no-edges 
Count in noisy graph G'	m_3	m_2	m_1	m_0

Figure 2: Four types of subgraphs with three nodes.

In our experiments, we show that this algorithm provides the utility close to $\mathsf{LocalLap}_{k\star}$ with the true max degree d_{max}.

4.2 One-Round Algorithms for Triangles.

Algorithm. Now, we focus our attention on the more challenging $f_\triangle$ query. This query is more challenging in the graph LDP model because no user is aware of any triangle; i.e., user v_i is not aware of any triangle formed by (v_i,v_j,v_k), because v_i cannot see any edge $(v_j,v_k) \in E$ in graph G.

One way to count $f_\triangle(G)$ with edge LDP is to apply the RR (Randomized Response) to a neighbor list. For example, user v_i applies the RR to $a_{i,1},\ldots,a_{i,i-1}$ (which corresponds to users $v_1,\ldots,v_{i-1}$ with smaller user IDs) in her neighbor list $\mathbf{a}_i$; i.e., we apply the RR to the lower triangular part of adjacency matrix $\mathbf{A}$, as described in Section 3.2. Then the data collector constructs a noisy graph $G' = (V,E') \in \mathcal{G}$ from the lower triangular part of the noisy adjacency matrix, and estimates the number of triangles from G'. However, simply counting the triangles in G' can introduce a significant bias because G' is denser than G especially when ε is small.

Through clever post-processing known as empirical estimation [32, 40, 57], we are able to obtain an unbiased estimate of $f_\triangle(G)$ from G'. Specifically, a subgraph with three nodes can be divided into four types depending on the number of edges. Three nodes with three edges form a triangle. We refer to three nodes with two edges, one edge, and no edges as *2-edges*, *1-edge*, and *no-edges*, respectively. Figure 2 shows their shapes. $f_\triangle(G)$ can be expressed using m_3, m_2, m_1, and m_0 as follows:

Proposition 2. *Let $G' = (V,E')$ be a noisy graph generated by applying the RR to the lower triangular part of $\mathbf{A}$. Let $m_3,m_2,m_1,m_0 \in \mathbb{Z}_{\geq 0}$ be respectively the number of triangles, 2-edges, 1-edge, and no-edges in G'. Then*

$$\mathbb{E}\left[\frac{e^{3\varepsilon}}{(e^{\varepsilon}-1)^3}m_3 - \frac{e^{2\varepsilon}}{(e^{\varepsilon}-1)^3}m_2 + \frac{e^{\varepsilon}}{(e^{\varepsilon}-1)^3}m_1 - \frac{1}{(e^{\varepsilon}-1)^3}m_0\right] = f_\triangle(G). \tag{4}$$

Therefore, the data collector can count m_3, m_2, m_1, and m_0 from G', and calculate an unbiased estimate of $f_\triangle(G)$ by (4). In Appendix A, we show that the l_2 loss is significantly reduced by this empirical estimation.

Algorithm 2 shows this algorithm. In line 2, user v_i applies the RR with privacy budget ε (denoted by RR_ε) to $a_{i,1},\ldots,a_{i,i-1}$ in her neighbor list $\mathbf{a}_i$, and outputs $R_i =$

```
Data: Graph $G$ represented as neighbor lists
      $\mathbf{a}_1, \dots, \mathbf{a}_n \in \{0,1\}^n$, privacy budget $\varepsilon \in \mathbb{R}_{\geq 0}$.
Result: Private estimate of $f_\triangle(G)$.
1 for $i = 1$ to $n$ do
2     $R_i \leftarrow (RR_\varepsilon(a_{i,1}), \dots, RR_\varepsilon(a_{i,i-1}))$;
3     $release(R_i)$;
4 end
5 $G' = (V, E') \leftarrow$ UndirectedGraph$(R_1, \dots, R_n)$;
  /* Counts $m_3, m_2, m_1, m_0$ in $G'$.                    */
6 $(m_3, m_2, m_1, m_0) \leftarrow$ Count$(G')$;
7 return $\frac{1}{(e^\varepsilon-1)^3}(e^{3\varepsilon}m_3 - e^{2\varepsilon}m_2 + e^\varepsilon m_1 - m_0)$
```

Algorithm 2: LocalRR$_\triangle$

$(RR_\varepsilon(a_{i,1}), \dots, RR_\varepsilon(a_{i,i-1}))$. In other words, we apply the RR to the lower triangular part of $\mathbf{A}$ and there is no overlap between edges sent by users. In line 5, the data collector uses a function (denoted by `UndirectedGraph`) that converts the bits of $(R_1, \dots, R_n)$ into an undirected graph $G' = (V, E')$ by adding edge (v_i, v_j) with $i > j$ to E' if and only if the j-th bit of R_i is 1. Note that G' is biased, as explained above. In line 6, the data collector uses a function (denoted by `Count`) that calculates m_3, m_2, m_1, and m_0 from G'. Finally, the data collector outputs the expression inside the expectation on the left-hand side of (4), which is an unbiased estimator for $f_\triangle(G)$ by Proposition 2. We denote this algorithm by LocalRR$_\triangle$.

Theoretical properties. LocalRR$_\triangle$ provides the following guarantee.

Theorem 3. *LocalRR$_\triangle$ provides ε-edge LDP and ε-relationship DP.*

LocalRR$_\triangle$ does not have the doubling issue (i.e., it provides not 2ε but ε-relationship DP), because we apply the RR to the lower triangular part of $\mathbf{A}$, as explained in Section 3.2.

Unlike the RR and empirical estimation for tabular data [32], the expected l_2 loss of LocalRR$_\triangle$ is complicated. To simplify the utility analysis, we assume that G is generated from the Erdös-Rényi graph distribution $\mathbf{G}(n,\alpha)$ with edge existence probability α; i.e., each edge in G with n nodes is independently generated with probability $\alpha \in [0,1]$.

Theorem 4. *Let $\mathbf{G}(n,\alpha)$ be the Erdös-Rényi graph distribution with edge existence probability $\alpha \in [0,1]$. Let $p = \frac{1}{e^\varepsilon+1}$ and $\beta = \alpha(1-p) + (1-\alpha)p$. Let $\hat{f}_\triangle(G,\varepsilon)$ be the output of LocalRR$_\triangle$. If $G \sim \mathbf{G}(n,\alpha)$, then for all $\varepsilon \in \mathbb{R}_{\geq 0}$, $\mathbb{E}[l_2^2(\hat{f}_\triangle(G,\varepsilon), f_\triangle(G))] = O\left(\frac{e^{6\varepsilon}}{(e^\varepsilon-1)^6}\beta n^4\right)$.*

Note that we assume the Erdös-Rényi model only for the utility analysis of LocalRR$_\triangle$, and do not assume this model for the other algorithms. The upper-bound of LocalRR$_\triangle$ in Theorem 4 is less ideal than the upper-bounds of the other algorithms in that it does not consider all possible graphs $G \in \mathcal{G}$. Nevertheless, we also show that the l_2 loss of LocalRR$_\triangle$ is roughly consistent with Theorem 4 in our experiments using two real datasets (Section 5) and the Barabási-Albert graphs [9], which have power-law degree distribution (Appendix B).

The parameters α and β are edge existence probabilities in the original graph G and noisy graph G', respectively. Although α is very small in a sparse graph, β can be large for small ε. For example, if $\alpha \approx 0$ and $\varepsilon = 1$, then $\beta \approx \frac{1}{e+1} = 0.27$.

Theorem 4 states that for large n, the l_2 loss of LocalRR$_\triangle$ ($= O(n^4)$) is much larger than the l_2 loss of LocalRR$_{k\star}$ ($= O(n)$). This follows from the fact that user v_i is not aware of any triangle formed by (v_i, v_j, v_k), as explained above.

In contrast, counting $f_\triangle(G)$ in the centralized model is much easier because the data collector sees all triangles in G; i.e., the data collector knows $f_\triangle(G)$. The sensitivity of $f_\triangle$ is at most $\tilde{d}_{max}$ (after graph projection). Thus, we can consider a simple algorithm that outputs $f_\triangle(G) + \mathrm{Lap}(\tilde{d}_{max}/\varepsilon)$. We denote this algorithm by CentralLap$_\triangle$. CentralLap$_\triangle$ attains the expected l_2 loss ($=$ variance) of $O\left(\frac{\tilde{d}_{max}^2}{\varepsilon^2}\right)$.

The large l_2 loss of LocalRR$_\triangle$ is caused by the fact that each edge is released independently with some probability of being flipped. In other words, there are three independent random variables that influence any triangle in G'. The next algorithm, using interaction, reduces this influencing number from three to one by using the fact that a user knows the existence of two edges for any triangle that involves the user.

4.3 Two-Rounds Algorithms for Triangles

Algorithm. Allowing for two-rounds interaction, we are able to compute $f_\triangle$ with a significantly improved l_2 loss, albeit with a higher per-user communication overhead. As described in Section 4.2, it is impossible for user v_i to see edge $(v_j, v_k) \in E$ in graph $G = (V, E)$ at the first round. However, if the data collector publishes a noisy graph $G' = (V, E')$ calculated by LocalRR$_\triangle$ at the first round, then user v_i can see a noisy edge $(v_j, v_k) \in E'$ in the noisy graph G' at the second round. Then user v_i can count the number of *noisy triangles* formed by (v_i, v_j, v_k) such that $(v_i, v_j) \in E$, $(v_i, v_k) \in E$, and $(v_j, v_k) \in E'$, and send the noisy triangle counts with the Laplacian noise to the data collector in an analogous way to LocalLap$_{k\star}$. Since user v_i always knows that two edges (v_i, v_j) and (v_i, v_k) exist in G, there is only one noisy edge in any noisy triangle (whereas all edges are noisy in LocalRR$_\triangle$). This is an intuition behind our proposed two-rounds algorithm.

As with the RR in Section 4.2, simply counting the noisy triangles can introduce a bias. Therefore, we calculate an empirical estimate of $f_\triangle(G)$ from the noisy triangle counts. Specifically, the following is the empirical estimate of $f_\triangle(G)$:

Proposition 3. *Let $G' = (V, E')$ be a noisy graph generated by applying the RR with privacy budget $\varepsilon_1 \in \mathbb{R}_{\geq 0}$ to the lower triangular part of $\mathbf{A}$. Let $p_1 = \frac{1}{e^{\varepsilon_1}+1}$. Let $t_i \in \mathbb{Z}_{\geq 0}$ be the number of triplets (v_i, v_j, v_k) such that $j < k < i$, $(v_i, v_j) \in E$, $(v_i, v_k) \in E$, and $(v_j, v_k) \in E'$. Let $s_i \in \mathbb{Z}_{\geq 0}$ be the number* *of triplets (v_i, v_j, v_k) such that $j < k < i$, $(v_i, v_j) \in E$, and $(v_i, v_k) \in E$. Let $w_i = t_i - p_1 s_i$. Then*

$$\mathbb{E}\left[\frac{1}{1-2p_1}\sum_{i=1}^n w_i\right] = f_\triangle(G). \quad (5)$$

Note that in Proposition 3, we count only triplets (v_i, v_j, v_k) with $j < k < i$ to use only the lower triangular part of $\mathbf{A}$. t_i is the number of noisy triangles user v_i can see at the second round. s_i is the number of 2-stars of which user v_i is a center. Since t_i and s_i can reveal information about an edge in G, user v_i adds the Laplacian noise to w_i $(= t_i - p_1 s_i)$ in (5), and sends it to the data collector. Then the data collector calculates an unbiased estimate of $f_\triangle(G)$ by (5).

```
Data: Graph $G$ represented as neighbor lists
      $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \{0,1\}^n$, two privacy budgets
      $\varepsilon_1, \varepsilon_2 > 0$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$.
Result: Private estimate of $f_\triangle(G)$.
1  $p_1 \leftarrow \frac{1}{e^{\varepsilon_1}+1}$;
   /* First round.                                   */
2  for $i = 1$ to $n$ do
3      $R_i \leftarrow (RR_{\varepsilon_1}(a_{i,1}), \ldots, RR_{\varepsilon_1}(a_{i,i-1}))$;
4      $release(R_i)$;
5  end
6  $G' = (V, E') \leftarrow$ UndirectedGraph$(R_1, \ldots, R_{i-1})$;
   /* Second round.                                  */
7  for $i = 1$ to $n$ do
8      $\mathbf{a}_i \leftarrow$ GraphProjection$(\mathbf{a}_i, \tilde{d}_{max})$;
9      $t_i \leftarrow |\{(v_i, v_j, v_k) : j < k < i, a_{i,j} = a_{i,k} =$
           $1, (v_j, v_k) \in E'\}|$;
10     $s_i \leftarrow |\{(v_i, v_j, v_k) : j < k < i, a_{i,j} = a_{i,k} = 1\}|$;
11     $w_i \leftarrow t_i - p_1 s_i$;
12     $\hat{w}_i \leftarrow w_i + \mathrm{Lap}(\frac{\tilde{d}_{max}}{\varepsilon_2})$;
13     $release(\hat{w}_i)$;
14 end
15 return $\frac{1}{1-2p_1}\sum_{i=1}^n \hat{w}_i$
```
Algorithm 3: Local2Rounds$_\triangle$

Algorithm 3 contains the formal description of this process. It takes as input a graph G, the privacy budgets $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$ at the first and second rounds, respectively, and a non-negative integer $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$. At the first round, we apply the RR to the lower triangular part of $\mathbf{A}$ (i.e., there is no overlap between edges sent by users) and use the `UndirectedGraph` function to obtain a noisy graph $G' = (V, E')$ by the RR in the same way as Algorithm 2. Note that G' is biased. We calculate an unbiased estimate of $f_\triangle(G)$ from G' at the second round.

At the second round, each user v_i calculates $\hat{w}_i = w_i + \mathrm{Lap}(\frac{\tilde{d}_{max}}{\varepsilon_2})$ by adding the Laplacian noise to w_i in Proposition 3 whose sensitivity is at most $\tilde{d}_{max}$ (as we will prove in Theorem 5). Finally, we output $\frac{1}{1-2p_1}\sum_{i=1}^n \hat{w}_i$, which is an unbiased estimate of $f_\triangle(G)$ by Proposition 3. We call this algorithm Local2Rounds$_\triangle$.

Theoretical properties. Local2Rounds$_\triangle$ has the following guarantee.

Theorem 5. *Local2Rounds$_\triangle$ provides $(\varepsilon_1+\varepsilon_2)$-edge LDP and $(\varepsilon_1+\varepsilon_2)$-relationship DP.*

As with LocalRR$_\triangle$, Local2Rounds$_\triangle$ does not have the doubling issue; i.e., it provides ε-relationship DP (not 2ε). This follows from the fact that we use only the lower triangular part of $\mathbf{A}$; i.e., we assume $j < k < i$ in counting t_i and s_i.

Theorem 6. *Let $\hat{f}_\triangle(G, \varepsilon_1, \varepsilon_2, \tilde{d}_{max})$ be the output of Local2Rounds$_\triangle$. Then, for all $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, and $G \in \mathcal{G}$ such that the maximum degree d_{max} of G is at most $\tilde{d}_{max}$, $\mathbb{E}[l_2^2(\hat{f}_\triangle(G, \varepsilon_1, \varepsilon_2, \tilde{d}_{max}), f_\triangle(G))] \leq O\left(\frac{e^{\varepsilon_1}}{(1-e^{\varepsilon_1})^2}\left(\tilde{d}_{max}^3 n + \frac{e^{\varepsilon_1}}{\varepsilon_2^2}\tilde{d}_{max}^2 n\right)\right)$.*

Theorem 6 means that for triangles, the l_2 loss is reduced from $O(n^4)$ to $O(\tilde{d}_{max}^3 n)$ by introducing an additional round.

Private calculation of d_{max}. As with k-stars, we can privately calculate d_{max} by using the method described in Section 4.1. Furthermore, the private calculation of d_{max} does not increase the number of rounds; i.e., we can run Local2Rounds$_\triangle$ with the private calculation of d_{max} in two rounds.

Specifically, let $\varepsilon_0 \in \mathbb{R}_{\geq 0}$ be the privacy budget for the private calculation of d_{max}. At the first round, each user v_i adds $\mathrm{Lap}(\frac{1}{\varepsilon_0})$ to her degree d_i, and sends the noisy degree $\hat{d}_i$ $(= d_i + \mathrm{Lap}(\frac{1}{\varepsilon_0}))$ to the data collector, along with the outputs $R_i = (RR_\varepsilon(a_{i,1}), \ldots, RR_\varepsilon(a_{i,i-1}))$ of the RR. The data collector calculates the noisy max degree $\hat{d}_{max}$ $(= \max\{\hat{d}_1, \ldots, \hat{d}_n\})$ as an estimate of d_{max}, and sends it back to all users. At the second round, we run Local2Rounds$_\triangle$ with input G (represented as $\mathbf{a}_1, \ldots, \mathbf{a}_n$), ε_1, ε_2, and $\lfloor \hat{d}_{max} \rfloor$.

At the first round, the calculation of $\hat{d}_{max}$ provides ε_0-edge LDP. Note that it provides $2\varepsilon_0$-relationship DP (i.e., it has the doubling issue) because one edge $(v_i, v_j) \in E$ affects both of the degrees d_i and d_j by 1. At the second round, LocalLap$_{k\star}$ provides $(\varepsilon_1+\varepsilon_2)$-edge LDP and $(\varepsilon_1+\varepsilon_2)$-relationship DP (Theorem 5). Then by the composition theorem [23], this two-rounds algorithm provides $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP and $(2\varepsilon_0+\varepsilon_1+\varepsilon_2)$-relationship DP. Although the total privacy budget is larger for relationship DP, the difference $(= \varepsilon_0)$ can be very small. In fact, we set $(\varepsilon_0, \varepsilon_1, \varepsilon_2) = (0.1, 0.45, 0.45)$ or $(0.2, 0.9, 0.9)$ in our experiments (i.e., the difference is 0.1 or 0.2), and show that this algorithm provides almost the same utility as Local2Rounds$_\triangle$ with the true max degree d_{max}.

Time complexity. We also note that Local2Rounds$_\triangle$ has an advantage over LocalRR$_\triangle$ in terms of the time complexity.

Specifically, LocalRR$_\triangle$ is inefficient because the data collector has to count the number of triangles m_3 in the noisy graph G'. Since the noisy graph G' is dense (especially when ε is small) and there are $\binom{n}{3}$ subgraphs with three nodes in

G', the number of triangles is $m_3 = O(n^3)$. Then, the time complexity of $\mathsf{LocalRR}_\triangle$ is also $O(n^3)$, which is not practical for a graph with a large number of users n. In fact, we implemented $\mathsf{LocalRR}_\triangle$ ($\varepsilon = 1$) with C/C++ and measured its running time using one node of a supercomputer (ABCI: AI Bridging Cloud Infrastructure [4]). When $n = 5000$, 10000, 20000, and 40000, the running time was 138, 1107, 9345, and 99561 seconds, respectively; i.e., the running time was almost cubic in n. We can also estimate the running time for larger n. For example, when $n = 1000000$, $\mathsf{LocalRR}_\triangle$ ($\varepsilon = 1$) would require about 35 years ($= 1107 \times 100^3/(3600 \times 24 \times 365)$).

In contrast, the time complexity of $\mathsf{Local2Rounds}_\triangle$ is $O(n^2 + nd_{max}^2)$[1]. The factor of n^2 comes from the fact that the size of the noisy graph G' is $O(n^2)$. This also causes a large communication overhead, as explained below.

Communication overhead. In $\mathsf{Local2Rounds}_\triangle$, each user need to see the noisy graph G' of size $O(n^2)$ to count t_i and s_i. This results in a per-user communication overhead of $O(n^2)$. Although we do not simulate the communication overhead in our experiments that use $\mathsf{Local2Rounds}_\triangle$, the $O(n^2)$ overhead might limit its application in very large graphs. An interesting avenue of future work is how to compress the graph size (e.g., via graph projection or random projection) to reduce both the time complexity and the communication overhead.

4.4 Lower Bounds

We show a general lower bound on the l_2 loss of private estimators $\hat{f}$ of real-valued functions f in the one-round LDP model. Treating ε as a constant, we have shown that when $\tilde{d}_{max} = d_{max}$, the expected l_2 loss of $\mathsf{LocalLaplace}_{k\star}$ is $O(nd_{max}^{2k-2})$ (Theorem 2). However, in the centralized model, we can use the Laplace mechanism with sensitivity $2\binom{d_{max}}{k-1}$ to obtain l_2^2 errors of $O(d_{max}^{2k-2})$ for $f_{k\star}$. Thus, we ask if the factor of n is necessary in the one-round LDP model.

We answer this question affirmatively. We show for many types of queries f, there is a lower bound on $l_2^2(f(G), \hat{f}(G))$ for any private estimator $\hat{f}$ of the form

$$\hat{f}(G) = \tilde{f}(\mathcal{R}_1(\mathbf{a}_1), \ldots, \mathcal{R}_n(\mathbf{a}_n)), \tag{6}$$

where $\mathcal{R}_1, \ldots, \mathcal{R}_n$ satisfy ε-edge LDP or ε-relationship DP and $\tilde{f}$ is an aggregate function that takes $\mathcal{R}_1(\mathbf{a}_1), \ldots, \mathcal{R}_n(\mathbf{a}_n)$ as input and outputs $\hat{f}(G)$. Here we assume that $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are independently run, meaning that they are in the one-round setting. For our lower bound, we require that input edges to f be "independent" in the sense that adding an edge to an input graph G independently change f by at least $D \in \mathbb{R}$. The specific structure of input graphs we require is as follows:

[1]When we evaluate $\mathsf{Local2Rounds}_\triangle$ in our experiments, we can apply the RR to only edges that are required at the second round; i.e., $(v_j, v_k) \in G'$ in line 8 of Algorithm 3. Then the time complexity of $\mathsf{Local2Rounds}_\triangle$ can be reduced to $O(nd_{max}^2)$ in total. We also confirmed that when $n = 1000000$, the running time of $\mathsf{Local2Rounds}_\triangle$ was 311 seconds on one node of the ABCI. Note, however, that this does *not* protect individual privacy, because it reveals the fact that users v_j and v_k are friends with u_i to the data collector.

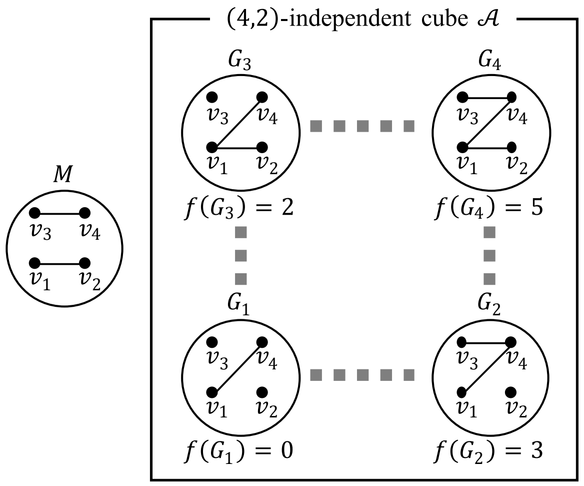


Figure 3: $(4,2)$-independent cube $\mathcal{A}$ for f. In this example, $M = \{(v_1,v_2),(v_3,v_4)\}$, $G_1 = (V,E)$, $\mathcal{A} = \{(V, E \cup N) : N \subseteq M\}$, $C_{(v_1,v_2)} = 2$, and $C_{(v_3,v_4)} = 3$. Adding (v_1,v_2) and (v_3,v_4) increase f by 2 and 3, respectively.

Definition 5. *[(n,D)-independent cube for f] Let $D \in \mathbb{R}_{\geq 0}$. For $\kappa \in \mathbb{N}$, let $G = (V,E) \in \mathcal{G}$ be a graph on $n = 2\kappa$ nodes, and let $M = \{(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}), \ldots, (v_{i_{2\kappa-1}}, v_{i_{2\kappa}})\}$ for integers $i_j \in [n]$ be a set of edges such that each of $i_1, \ldots, i_{2\kappa}$ is distinct (i.e., perfect matching on the nodes). Suppose that M is disjoint from E; i.e., $(v_{i_{2j-1}}, v_{i_{2j}}) \notin E$ for any $j \in [\kappa]$. Let $\mathcal{A} = \{(V, E \cup N) : N \subseteq M\}$. Note that $\mathcal{A}$ is a set of 2^κ graphs. We say $\mathcal{A}$ is an (n,D)-*independent cube for f *if for all $G' = (V,E') \in \mathcal{A}$, we have*

$$f(G') = f(G) + \sum_{e \in E' \cap M} C_e,$$

where $C_e \in \mathbb{R}$ satisfies $|C_e| \geq D$ for any $e \in M$.

Such a set of inputs has an "independence" property because, regardless of which edges from M has been added before, adding edge $e \in M$ always changes f by C_e. Figure 3 shows an example of a $(4,2)$-independent cube for f.

We can also construct a independent cube for a k-star function as follows. Assume that n is even. It is well known in graph theory that if n is even, then for any $d \in [n-1]$, there exists a d-regular graph where every node has degree d [25]. Therefore, there exists a $(d_{max}-1)$-regular graph $G = (V,E)$ of size n. Pick an arbitrary perfect matching M on the nodes. Now, let $G' = (V,E')$ such that $E' = E \setminus M$. Every node in G' has degree between $d_{max}-2$ and $d_{max}-1$. Adding an edge in M to G' will produce at least $2\binom{d_{max}-2}{k-1}$ new k-stars. Thus, $\mathcal{A} = \{(V, E' \cup N) : N \subseteq M\}$ forms an $(n, 2\binom{d_{max}-2}{k-1})$-independent cube for $f_{k\star}$. Note that the maximum degree of each graph in $\mathcal{A}$ is at most d_{max}. Figure 4 shows how to construct an independent cube for a k-star function when $n = 6$

	Centralized	One-round local		Two-rounds local
	Upper Bound	Lower Bound	Upper Bound	Upper Bound
$f_{k\star}$	$O\left(\frac{d_{max}^{2k-2}}{\varepsilon^2}\right)$	$\Omega\left(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2} d_{max}^{2k-2} n\right)$	$O\left(\frac{d_{max}^{2k-2}}{\varepsilon^2} n\right)$	$O\left(\frac{d_{max}^{2k-2}}{\varepsilon^2} n\right)$
$f_\triangle$	$O\left(\frac{d_{max}^{2}}{\varepsilon^2}\right)$	$\Omega\left(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2} d_{max}^{2} n\right)$	$O\left(\frac{e^{6\varepsilon}}{(e^{\varepsilon}-1)^6} n^4\right)$ (when $G \sim \mathbf{G}(n,\alpha)$)	$O\left(\frac{e^{\varepsilon}}{(e^{\varepsilon}-1)^2} (d_{max}^3 n + \frac{e^{\varepsilon}}{\varepsilon^2} d_{max}^2 n)\right)$

Table 2: Bounds on l_2 losses for privately estimating $f_{k\star}$ and $f_\triangle$ with ε-edge LDP. For upper-bounds, we assume that $\tilde{d}_{max} = d_{max}$. For the centralized model, we use the Laplace mechanism. For the one-round $f_\triangle$ algorithm, we apply Theorem 4 with constant α. For the two-round protocol $f_\triangle$ algorithm, we apply Theorem 6 with $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$.

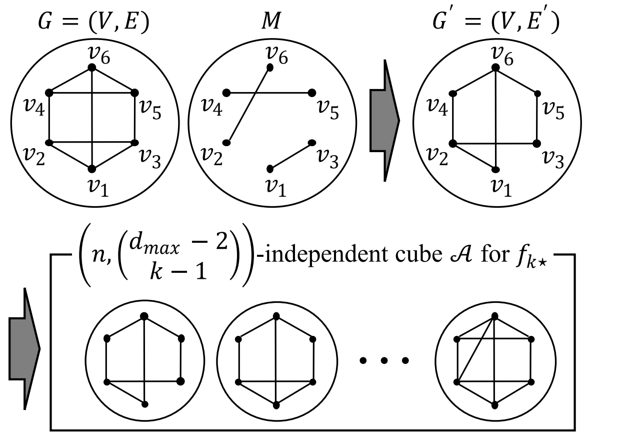

$G = (V, E)$
M
$G' = (V, E')$
v_1
v_2
v_3
v_4
v_5
v_6
$\left(n, \binom{d_{max}-2}{k-1}\right)$-independent cube $\mathcal{A}$ for $f_{k\star}$

Figure 4: Construction of an independent cube for a k-star function ($n = 6$, $d_{max} = 4$). From a 3-regular graph $G = (V, E)$ and $M = \{(v_1, v_3), (v_2, v_6), (v_4, v_5)\}$, we make a graph $G' = (V, E')$ such that $E' = E \setminus M$. Then $\mathcal{A} = \{(V, E' \cup N) : N \subseteq M\}$ forms an $(n, 2\binom{d_{max}-2}{k-1})$-independent cube for $f_{k\star}$.

and $d_{max} = 4$.

Using the structure that the (n, D)-independent cube imposes on f, we can prove a lower bound:

Theorem 7. *Let $\hat{f}(G)$ have the form of* (6)*, where $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are independently run. Let $\mathcal{A}$ be an (n, D)-independent cube for f. If $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ provides ε-relationship DP, then we have*

$$\frac{1}{\mathcal{A}} \sum_{G \in \mathcal{A}} \mathbb{E}[l_2^2(f(G), \hat{f}(G))] = \Omega\left(\frac{e^{\varepsilon}}{(e^{\varepsilon}+1)^2} nD^2\right).$$

A corollary of Theorem 7 is that if $\mathcal{R}_1, \ldots, \mathcal{R}_n$ satisfy ε-edge LDP, then they satisfy 2ε -relationship DP and thus for edge LDP we have a lower bound of $\Omega\left(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2} nD^2\right)$.

Theorem 7, combined with the fact that there exists an $(n, 2\binom{d_{max}-2}{k-1})$-independent cube for a k-star function implies Corollary 1. In Appendix C, we also construct an $(n, \frac{d_{max}}{2} - 2)$ independent cube for $f_\triangle$ and establish a lower bound of $\Omega(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2} n d_{max}^2)$ for $f_\triangle$.

The upper and lower bounds on the l_2 losses shown in this section appear in Table 2.

5 Experiments

Based on our theoretical results in Section 4, we would like to pose the following questions:

- For triangle counts, how much does the two-rounds interaction help over a single round in practice?
- What is the privacy-utility trade-off of our LDP algorithms (i.e., how beneficial are our LDP algorithms)?

We conducted experiments to answer to these questions.

5.1 Experimental Set-up

We used the following two large-scale datasets:

IMDB. The Internet Movie Database (denoted by IMDB) [2] includes a bipartite graph between 896308 actors and 428440 movies. We assumed actors as users. From the bipartite graph, we extracted a graph G^* with 896308 nodes (actors), where an edge between two actors represents that they have played in the same movie. There are 57064358 edges in G^*, and the average degree in G^* is 63.7 ($= \frac{57064358}{896308}$).

Orkut. The Orkut online social network dataset (denoted by Orkut) [37] includes a graph G^* with 3072441 users and 117185083 edges. The average degree in G^* is 38.1 ($= \frac{117185083}{3072441}$). Therefore, Orkut is more sparse than IMDB (whose average degree in G^* is 63.7).

For each dataset, we randomly selected n users from the whole graph G^*, and extracted a graph $G = (V, E)$ with n users. Then we estimated the number of triangles $f_\triangle(G)$, the number of k-stars $f_{k\star}(G)$, and the clustering coefficient ($= \frac{3f_\triangle(G)}{f_{2\star}(G)}$) using ε-edge LDP (or ε-edge centralized DP) algorithms in Section 4. Specifically, we used the following algorithms:

Algorithms for triangles. For algorithms for estimating $f_\triangle(G)$, we used the following three algorithms: (1) the RR (Randomized Response) with the empirical estimation method in the local model (i.e., LocalRR$_\triangle$ in Section 4.2), (2) the two-rounds algorithm in the local model (i.e., Local2Rounds$_\triangle$ in Section 4.3), and (3) the Laplacian mechanism in the centralized model (i.e., CentralLap$\triangle$ in Section 4.2).

Algorithms for k-stars. For algorithms for estimating $f_{k\star}(G)$, we used the following two algorithms: (1) the Lapla-

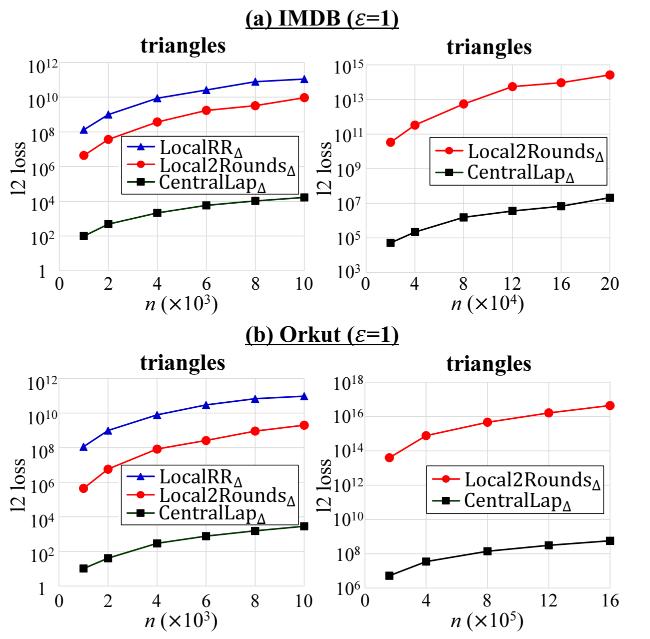


Figure 5: Relation between the number of users n and the l_2 loss in triangle counts when $\varepsilon = 1$ ($\varepsilon_1 = \varepsilon_2 = \frac{1}{2}$, $\tilde{d}_{max} = d_{max}$). Here we do not evaluate LocalRR$_\triangle$ when $n > 10000$, because it is inefficient (see Section 4.3 "Time complexity").

cian mechanism in the local model (i.e., LocalLap$_{k\star}$ in Section 4.1) and (2) the Laplacian mechanism in the centralized model (i.e., CentralLap$_{k\star}$ in Section 4.1).

For each algorithm, we evaluated the l_2 loss and the relative error (as described in Section 3.4), while changing the values of n and ε. To stabilize the performance, we attempted $\gamma \in \mathbb{N}$ ways to randomly select n users from G^*, and averaged the utility value over all the γ ways to randomly select n users. When we changed n from 1000 to 10000, we set $\gamma = 100$ because the variance was large. For other cases, we set $\gamma = 10$.

In Appendix B, we also report experimental results using artificial graphs based on the Barabási-Albert model [9].

5.2 Experimental Results

Relation between n and the l_2 loss. We first evaluated the l_2 loss of the estimates of $f_\triangle(G)$, $f_{2\star}(G)$, and $f_{3\star}(G)$ while changing the number of users n. Figures 5 and 6 shows the results ($\varepsilon = 1$). Here we did not evaluate LocalRR$_\triangle$ when n was larger than 10000, because LocalRR$_\triangle$ was inefficient (as described in Section 4.3 "Time complexity"). In Local2Rounds$_\triangle$, we set $\varepsilon_1 = \varepsilon_2 = \frac{1}{2}$. As for $\tilde{d}_{max}$, we set $\tilde{d}_{max} = d_{max}$ (i.e., we assumed that d_{max} is publicly available and did not perform graph projection) because we want to examine how well our theoretical results hold in our experiments. We also evaluate the effectiveness of the private calculation of d_{max} at the end of Section 5.2.

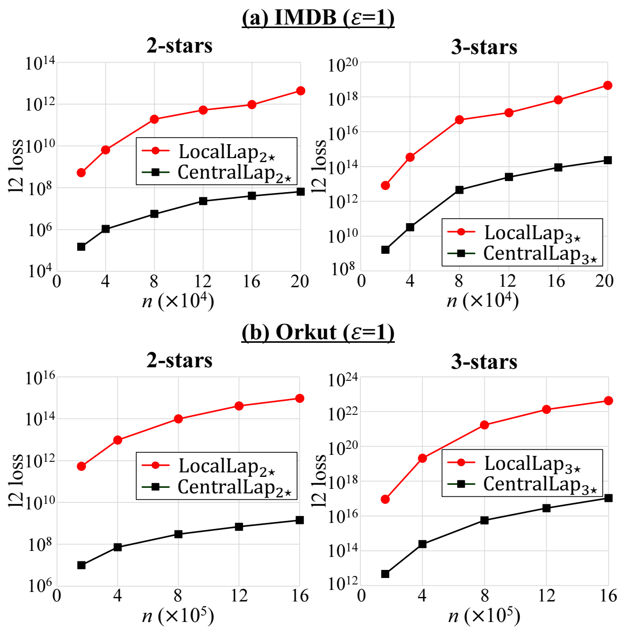


Figure 6: Relation between the number of users n and the l_2 loss in k-star counts when $\varepsilon = 1$ ($\varepsilon_1 = \varepsilon_2 = \frac{1}{2}$, $\tilde{d}_{max} = d_{max}$).

Figure 5 shows that Local2Rounds$_\triangle$ significantly outperforms LocalRR$_\triangle$. Specifically, the l_2 loss of Local2Rounds$_\triangle$ is smaller than that of LocalRR$_\triangle$ by a factor of about 10^2. The difference between Local2Rounds$_\triangle$ and LocalRR$_\triangle$ is larger in Orkut. This is because Orkut is more sparse, as described in Section 5.1. For example, when $n = 10000$, the maximum degree d_{max} in G was 73.5 and 27.8 on average in IMDB and Orkut, respectively. Recall that for a fixed ε, the expected l_2 loss of Local2Rounds$_\triangle$ and LocalRR$_\triangle$ can be expressed as $O(nd_{max}^3)$ and $O(n^4)$, respectively. Thus Local2Rounds$_\triangle$ significantly outperforms LocalRR$_\triangle$, especially in sparse graphs.

Figures 5 and 6 show that the l_2 loss is roughly consistent with our upper-bounds in terms of n. Specifically, LocalRR$_\triangle$, Local2Rounds$_\triangle$, CentralLap$_\triangle$, LocalLap$_{k\star}$, and CentralLap$_{k\star}$ achieve the expected l_2 loss of $O(n^4)$, $O(nd_{max}^3)$, $O(d_{max}^2)$, $O(nd_{max}^{2k-2})$, and $O(d_{max}^{2k-2})$, respectively. Here note that each user's degree increases roughly in proportion to n (though the degree is much smaller than n), as we randomly select n users from the whole graph G^*. Assuming that $d_{max} = O(n)$, Figures 5 and 6 are roughly consistent with the upper-bounds. The figures also show the limitations of the local model in terms of the utility when compared to the centralized model.

Relation between ε and the l_2 loss. Next we evaluated the l_2 loss when we changed the privacy budget ε in edge LDP. Figure 7 shows the results for triangles and 2-stars ($n = 10000$). Here we omit the result of 3-stars because it is similar to that of 2-stars. In Local2Rounds$_\triangle$, we set $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$.

Figure 7 shows that the l_2 loss is roughly consistent with our upper-bounds in terms of ε. For example, when we decrease ε from 0.4 to 0.1, the l_2 loss increases by a factor of

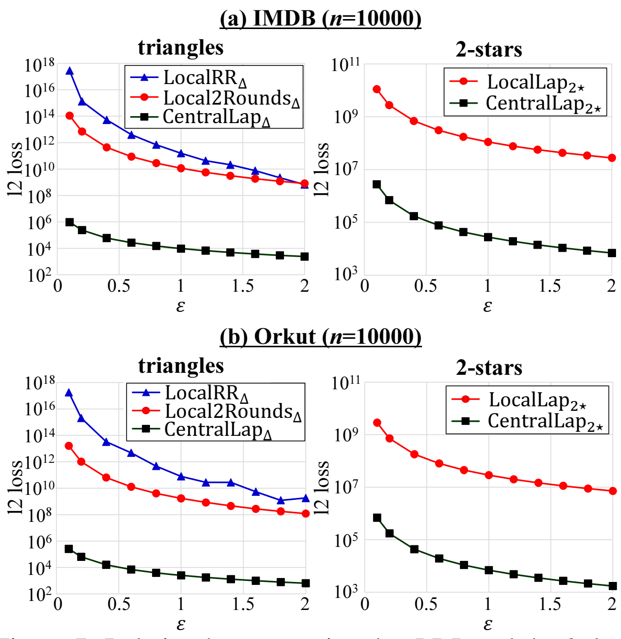


Figure 7: Relation between ε in edge LDP and the l_2 loss when $n = 10000$ ($\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$, $\tilde{d}_{max} = d_{max}$).

about 5000, 200, and 16 for both the datasets in LocalRR$_\triangle$, Local2Rounds$_\triangle$, and CentralLap$_\triangle$, respectively. They are roughly consistent with our theoretical results that for small ε, the expected l_2 loss of LocalRR$_\triangle$, Local2Rounds$_\triangle$, and CentralLap$_\triangle$ is $O(\varepsilon^{-6})$[2], $O(\varepsilon^{-4})$, and $O(\varepsilon^{-2})$, respectively.

Figure 7 also shows that Local2Rounds$_\triangle$ significantly outperforms LocalRR$_\triangle$ especially when ε is small, which is also consistent with our theoretical results. Conversely, the difference between LocalRR$_\triangle$ and Local2Rounds$_\triangle$ is small when ε is large. This is because when ε is large, the RR outputs the true value with high probability. For example, when $\varepsilon \geq 5$, the RR outputs the true value with $\frac{e^\varepsilon}{e^\varepsilon+1} > 0.993$. However, LocalRR$_\triangle$ with such a large value of ε does not guarantee strong privacy, because it outputs the true value in most cases. Local2Rounds$_\triangle$ significantly outperforms LocalRR$_\triangle$ when we want to estimate $f_\triangle(G)$ or $f_{k\star}(G)$ with a strong privacy guarantee; e.g., $\varepsilon \leq 1$ [38].

Relative error. As the number of users n increases, the numbers of triangles $f_\triangle(G)$ and k-stars $f_{k\star}(G)$ increase. This causes the increase of the l_2 loss. Therefore, we also evaluated the relative error, as described in Section 3.4.

Figure 8 shows the relation between n and the relative error (we omit the result of 3-stars because it is similar to that of 2-stars). In the local model, we used Local2Rounds$_\triangle$ and LocalLap$_{k\star}$ for estimating $f_\triangle(G)$ and $f_{k\star}(G)$, respectively (we did not use Local2RR$_\triangle$, because it is both inaccurate and inefficient). For both algorithms, we set $\varepsilon = 1$ or 2 ($\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$ in Local2Rounds$_\triangle$) and $\tilde{d}_{max} = d_{max}$. Then

^{2}We used $e^\varepsilon \approx \varepsilon + 1$ to derive the upper-bound of LocalRR$_\triangle$ for small ε.

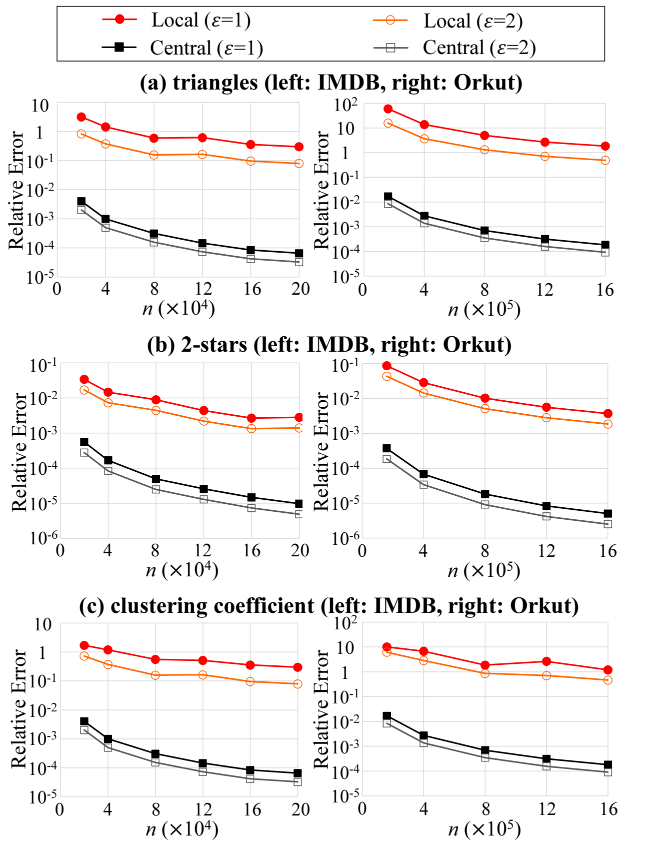


Figure 8: Relation between n and the relative error. In the local model, we used Local2Rounds$_\triangle$ ($\varepsilon = 1$ or 2) and LocalLap$_{k\star}$ ($\varepsilon = 1$ or 2) for estimating triangle counts $f_\triangle(G)$ and k-star counts $f_{k\star}(G)$, respectively ($\tilde{d}_{max} = d_{max}$).

we estimated the clustering coefficient as: $\frac{3\hat{f}_\triangle(G,\varepsilon_1,\varepsilon_2,d_{max})}{\hat{f}_{k\star}(G,\varepsilon,d_{max})}$, where $\hat{f}_\triangle(G,\varepsilon_1,\varepsilon_2,d_{max})$ and $\hat{f}_{k\star}(G,\varepsilon,d_{max})$ are the estimates of $f_\triangle(G)$ and $f_{k\star}(G)$, respectively. If the estimate of the clustering coefficient is smaller than 0 (resp. larger than 1), we set the estimate to 0 (resp. 1) because the clustering coefficient is always between 0 and 1. In the centralized model, we used CentralLap$_\triangle$ and CentralLap$_{k\star}$ ($\varepsilon = 1$ or 2, $\tilde{d}_{max} = d_{max}$) and calculated the clustering coefficient in the same way.

Figure 8 shows that for all cases, the relative error decreases with increase in n. This is because both $f_\triangle(G)$ and $f_{k\star}(G)$ significantly increase with increase in n. Specifically, let $f_{\triangle,v_i}(G) \in \mathbb{Z}_{\geq 0}$ the number of triangles that involve user v_i, and $f_{k\star,v_i}(G) \in \mathbb{Z}_{\geq 0}$ be the number of k-stars of which user v_i is a center. Then $f_\triangle(G) = \frac{1}{3}\sum_{i=1}^n f_{\triangle,v_i}(G)$ and $f_{k\star,v_i}(G) = \sum_{i=1}^n f_{k\star,v_i}(G)$. Since both $f_{\triangle,v_i}(G)$ and $f_{k\star,v_i}(G)$ increase with increase in n, both $f_\triangle(G)$ and $f_{k\star}(G)$ increase *at least* in proportion to n. Thus $f_\triangle(G)^2 \geq \Omega(n^2)$ and $f_{k\star}(G)^2 \geq \Omega(n^2)$. In contrast, Local2Rounds$_\triangle$, LocalLap$_{k\star}$, CentralLap$_\triangle$, and CentralLap$_{k\star}$ achieve the expected l_2 loss of $O(n)$, $O(n)$, $O(1)$, and $O(1)$, respectively (when we ignore d_{max} and ε), all of which are smaller than $O(n^2)$. Therefore, the relative error

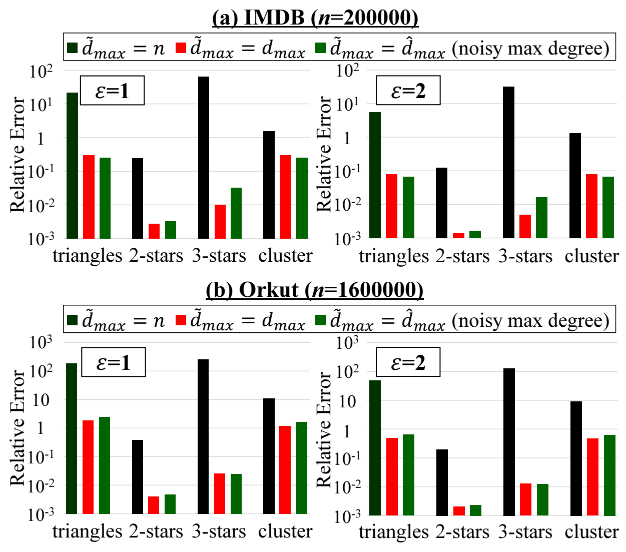


Figure 9: Relative error when $\tilde{d}_{max} = n$ (#users), d_{max} (max degree), or $\hat{d}_{max}$ (noisy max degree). We used Local2Rounds$_\triangle$ ($\varepsilon = 1$ or 2) and LocalLap$_{k\star}$ ($\varepsilon = 1$ or 2) for estimating triangle counts $f_\triangle(G)$ and k-star counts $f_{k\star}(G)$, respectively.

decreases with increase in n.

This result demonstrates that we can accurately estimate graph statistics for large n in the local model. In particular, the relative error is smaller in IMDB because IMDB is denser and includes a larger number of triangles and k-stars; i.e., the denominator of the relative error is large. For example, when $n = 200000$ and $\varepsilon = 1$, the relative error is 0.30 and 0.0028 for triangles and 2-stars, respectively. Note that the clustering coefficient requires 2ε because we need to estimate both $f_\triangle(G)$ and $f_{k\star}(G)$. Yet, we can still accurately calculate the clustering coefficient with a moderate privacy budget; e.g., the relative error of the clustering coefficient is 0.30 when the privacy budget is 2 (i.e., $\varepsilon = 1$). If n is larger, then ε would be smaller at the same value of the relative error.

Private calculation of d_{max}. We have so far assumed that $\tilde{d}_{max} = d_{max}$ (i.e., d_{max} is publicly available) in our experiments. We finally evaluate the methods to privately calculate d_{max} with ε_0-edge LDP (described in Sections 4.1 and 4.3).

Specifically, we used Local2Rounds$_\triangle$ and LocalLap$_{k\star}$ for estimating $f_\triangle(G)$ and $f_{k\star}(G)$, respectively, and evaluated the following three methods for setting $\tilde{d}_{max}$: (i) $\tilde{d}_{max} = n$; (ii) $\tilde{d}_{max} = d_{max}$; (iii) $\tilde{d}_{max} = \hat{d}_{max}$, where $\hat{d}_{max}$ is the private estimate of d_{max} (noisy max degree) in Sections 4.1 and 4.3.

We set $n = 200000$ in IMDB and $n = 1600000$ in Orkut. Regarding the total privacy budget ε in edge LDP for estimating $f_\triangle(G)$ or $f_{k\star}(G)$, we set $\varepsilon = 1$ or 2. We used $\frac{\varepsilon}{10}$ for privately calculating d_{max} (i.e., $\varepsilon_0 = \frac{\varepsilon}{10}$), and the remaining privacy budget $\frac{9\varepsilon}{10}$ as input to Local2Rounds$_\triangle$ or LocalLap$_{k\star}$. In Local2Rounds$_\triangle$, we set $\varepsilon_1 = \varepsilon_2$; i.e., we set $(\varepsilon_0, \varepsilon_1, \varepsilon_2) = (0.1, 0.45, 0.45)$ or $(0.2, 0.9, 0.9)$. Then we estimated the clustering coefficient in the same way as Figure 8.

Figure 9 shows the results. Figure 9 shows that the algorithms with $\tilde{d}_{max} = \hat{d}_{max}$ (noisy max degree) achieves the relative error close to (sometimes almost the same as) the algorithms with $\tilde{d}_{max} = d_{max}$ and significantly outperforms the algorithms with $\tilde{d}_{max} = n$. This means that we can privately estimate d_{max} without a significant loss of utility.

Summary of results. In summary, our experimental results showed that the estimation error of triangle counts is significantly reduced by introducing the interaction between users and a data collector. The results also showed that we can achieve small relative errors (much smaller than 1) for subgraph counts with privacy budget $\varepsilon = 1$ or 2 in edge LDP.

As described in Section 1, non-private subgraph counts may reveal some friendship information, and a central server may face data breaches. Our LDP algorithms are highly beneficial because they enable us to analyze the connection patterns in a graph (i.e., subgraph counts) or to understand how likely two friends of an individual will also be a friend (i.e., clustering coefficient) while strongly protecting individual privacy.

6 Conclusions

We presented a series of algorithms for counting triangles and k-stars under LDP. We showed that an additional round can significantly reduce the estimation error in triangles, and the algorithm based on the Laplacian mechanism provides an order optimal error in the non-interactive local model. We also showed lower-bounds for general functions including triangles and k-stars. We conducted experiments using two real datasets, and showed that our algorithms achieve small relative errors, especially when the number of users is large.

As future work, we would like to develop algorithms for other subgraph counts such as cliques and k-triangles [34].

Acknowledgments

Kamalika Chaudhuri and Jacob Imola would like to thank ONR under N00014-20-1-2334 and UC Lab Fees under LFR 18-548554 for research support. Takao Murakami was supported in part by JSPS KAKENHI JP19H04113.

References

[1] Tool: LDP graph statistics. https://github.com/LDPGraphStatistics/LDPGraphStatistics.

[2] 12th Annual Graph Drawing Contest. http://mozart.diei.unipg.it/gdcontest/contest2005/index.html, 2005.

[3] What to Do When Your Facebook Profile is Maxed Out on Friends. https://authoritypublishing.

com/social-media/what-to-do-when-your-facebook-profile-is-maxed-out-on-friends/, 2012.

[4] AI bridging cloud infrastructure (ABCI). https://abci.ai/, 2020.

[5] The diaspora* project. https://diasporafoundation.org/, 2020.

[6] Facebook Reports Third Quarter 2020 Results. https://investor.fb.com/investor-news/press-release-details/2020/Facebook-Reports-Third-Quarter-2020-Results/default.aspx, 2020.

[7] Jayadev Acharya, Clément L. Canonne, Yuhan Liu, Ziteng Sun, and Himanshu Tyagi. Interactive inference under information constraints. *CoRR*, 2007.10976, 2020.

[8] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *Proc. AISTATS'19*, pages 1120–1129, 2019.

[9] Albert-László Barabási. *Network Science*. Cambridge University Press, 2016.

[10] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. In *Proc. NIPS'17*, pages 2285—-2293, 2017.

[11] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proc. STOC'15*, pages 127–135, 2015.

[12] Vincent Bindschaedler and Reza Shokri. Synthesizing plausible privacy-preserving location traces. In *Proc. S&P'16*, pages 546–563, 2016.

[13] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *Proc. FOCS'12*, pages 410–419, 2012.

[14] Rui Chen, Gergely Acs, and Claude Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proc. CCS'12*, pages 638–649, 2012.

[15] Xihui Chen, Sjouke Mauw, and Yunior Ramírez-Cruz. Publishing community-preserving attributed social graphs with a differential privacy guarantee. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, (4):131–152, 2020.

[16] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proc. SIGMOD'16*, pages 123–138, 2016.

[17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Proc. NIPS'17*, pages 3574–3583, 2017.

[18] John Duchi and Ryan Rogers. Lower Bounds for Locally Private Estimation via Communication Complexity. *arXiv:1902.00582 [math, stat]*, May 2019. arXiv: 1902.00582.

[19] John Duchi, Martin Wainwright, and Michael Jordan. Minimax Optimal Procedures for Locally Private Estimation. *arXiv:1604.02390 [cs, math, stat]*, November 2017. arXiv: 1604.02390.

[20] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *Proc. FOCS'13*, pages 429–438, 2013.

[21] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy, data processing inequalities, and minimax rates. *CoRR*, 1302.3203, 2014.

[22] Cynthia Dwork. Differential privacy. In *Proc. ICALP'06*, pages 1–12, 2006.

[23] Cynthia Dwork and Aaron Roth. *The Algorithmic Foundations of Differential Privacy*. Now Publishers, 2014.

[24] Giulia Fanti, Vasyl Pihur, and Ulfar Erlingsson. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2016(3):1–21, 2016.

[25] Ghurumuruhan Ganesan. Existence of connected regular and nearly regular graphs. *CoRR*, 1801.08345, 2018.

[26] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference (SciPy'08)*, pages 11–15, 2008.

[27] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proc. ICDM'09*, pages 169–178, 2009.

[28] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. *CoRR*, 2010.08688, 2021.

[29] Matthew Joseph, Janardhan Kulkarni, Jieming Mao, and Zhiwei Steven Wu. Locally Private Gaussian Estimation. *arXiv:1811.08382 [cs, stat]*, October 2019. arXiv: 1811.08382.

[30] Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The Role of Interactivity in Local Differential Privacy. *arXiv:1904.03564 [cs, stat]*, November 2019. arXiv: 1904.03564.

[31] Matthew Joseph, Jieming Mao, and Aaron Roth. Exponential separations in local differential privacy. In *Proc. SODA'20*, pages 515–527, 2020.

[32] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In *Proc. ICML'16*, pages 2436–2444, 2016.

[33] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *Journal of Machine Learning Research*, 17(1):492–542, 2016.

[34] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[35] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, and Sofya Raskhodnikova. What can we learn privately? In *Proc. FOCS'08*, pages 531–540, 2008.

[36] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Proc. TCC'13*, pages 457–476, 2013.

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

[38] Ninghui Li, Min Lyu, and Dong Su. *Differential Privacy: From Theory to Practice*. Morgan & Claypool Publishers, 2016.

[39] Chris Morris. Hackers had a banner year in 2019. https://fortune.com/2020/01/28/2019-data-breach-increases-hackers/, 2020.

[40] Takao Murakami and Yusuke Kawamoto. Utility-optimized local differential privacy mechanisms for distribution estimation. In *Proc. USENIX Security'19*, pages 1877–1894, 2019.

[41] M. E. J. Newman. Random graphs with clustering. *Physical Review Letters*, 103(5):058701, 2009.

[42] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proc. STOC'07*, pages 75–84, 2007.

[43] Thomas Paul, Antonino Famulari, and Thorsten Strufe. A survey on decentralized online social networks. *Computer Networks*, 75:437–452, 2014.

[44] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. AdaCliP: Adaptive clipping for private SGD. *CoRR*, 1908.07643, 2019.

[45] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proc. CCS'16*, pages 192–203, 2016.

[46] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proc. CCS'17*, pages 425–438, 2017.

[47] Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. *CoRR*, 2006.14015, 2020.

[48] Sofya Raskhodnikova and Adam Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *CoRR*, 1504.07912, 2015.

[49] Sofya Raskhodnikova and Adam Smith. *Differentially Private Analysis of Graphs*, pages 543–547. Springer, 2016.

[50] Andrea De Salve, Paolo Mori, and Laura Ricci. A survey on privacy in decentralized online social networks. *Computer Science Review*, 27:154–176, 2018.

[51] Tara Seals. Data breaches increase 40% in 2016. https://www.infosecurity-magazine.com/news/data-breaches-increase-40-in-2016/, 2017.

[52] Shuang Song, Susan Little, Sanjay Mehta, Staal Vinterboy, and Kamalika Chaudhuri. Differentially private continual release of graph statistics. *CoRR*, 1809.02575, 2018.

[53] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qui, Hui (Wendy) Wang, and Ting Yu. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proc. CCS'19*, pages 703–717, 2019.

[54] Om Thakkar, Galen Andrew, and H. Brendan McMahan. Differentially private learning with adaptive clipping. *CoRR*, 1905.03871, 2019.

[55] Abhradeep Guha Thakurta, Andrew H. Vyrros, Umesh S. Vaishampayan, Gaurav Kapoor, Julien Freudiger, Vivek Rangarajan Sridhar, and Doug Davidson. Learning New Words, US Patent 9,594,741, Mar. 14 2017.

[56] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proc. CCS'14*, pages 1054–1067, 2014.

[57] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *Proc. USENIX Security'17*, pages 729–745, 2017.

[58] Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *Transactions on Data Privacy*, 6(2), 2013.

[59] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *Proc. PAKDD'13*, pages 329–340, 2013.

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

[61] Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. ireduct: Differential privacy with reduced relative errors. In *Proc. SIGMOD'11*, pages 229–240, 2011.

[62] Min Ye and Alexander Barga. Optimal schemes for discrete distribution estimation under local differential privacy. In *Proc. ISIT'17*, pages 759—-763, 2017.

[63] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. Towards locally differentially private generic graph metric estimation. In *Proc. ICDE'20*, pages 1922–1925, 2020.

[64] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. LF-GDPR: A framework for estimating graph metrics with local differential privacy. *IEEE Transactions on Knowledge and Data Engineering (Early Access)*, pages 1–16, 2021.

[65] Hailong Zhang, Sufian Latif, Raef Bassily, and Atanas Rountev. Differentially-private control-flow node coverage for software usage analysis. In *Proc. USENIX Security'20*, pages 1021–1038, 2020.

A Effectiveness of empirical estimation in $\mathsf{LocalRR}_\triangle$

In Section 4.2, we presented $\mathsf{LocalRR}_\triangle$, which uses the empirical estimation method after the RR. Here we show the effectiveness of empirical estimation by comparing $\mathsf{LocalRR}_\triangle$ with the RR without empirical estimation [46, 63].

As the RR without empirical estimation, we applied the RR to the lower triangular part of the adjacency matrix $\mathbf{A}$; i.e., we ran lines 1 to 6 in Algorithm 2. Then we output the number of noisy triangles m_3. We denote this algorithm by RR w/o emp.

Figure 10 shows the l_2 loss of $\mathsf{LocalRR}_\triangle$ and RR w/o emp when we changed n from 1000 to 10000 or ε in edge LDP

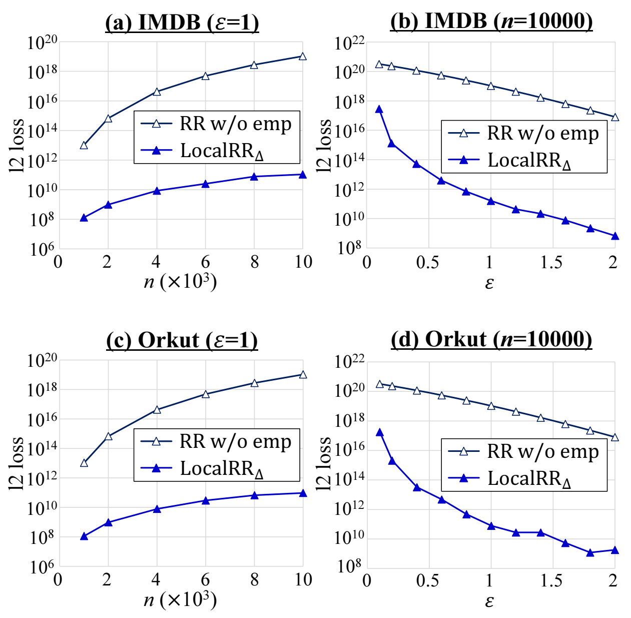


Figure 10: l_2 loss of $\mathsf{LocalRR}_\triangle$ and the RR without empirical estimation (RR w/o emp).

from 0.1 to 2. The experimental set-up is the same as Section 5.1. Figure 10 shows that $\mathsf{LocalRR}_\triangle$ significantly outperforms RR w/o emp, which means that the l_2 loss is significantly reduced by empirical estimation. As shown in Section 5, the l_2 loss of $\mathsf{LocalRR}_\triangle$ is also significantly reduced by an additional round of interaction.

B Experiments on Barabási-Albert Graphs

Experimental set-up. In Section 5, we evaluated our algorithms using two real datasets: IMDB and Orkut. We also evaluated our algorithms using artificial graphs that have power-law degree distributions. We used the BA (Barabási-Albert) model [9] to generate such graphs.

In the BA model, an artificial graph (referred to as a BA graph) is grown by adding new nodes one at a time. Each new node is connected to $\lambda \in \mathbb{N}$ existing nodes with probability proportional to the degree of the existing node. In our experiments, we used NetworkX [26], a Python package for graph analysis, to generate BA graphs.

We generated a BA graph G^* with 1000000 nodes using NetworkX. For the attachment parameter λ, we set $\lambda = 10$ or 50. When $\lambda = 10$ (resp. 50), the average degree of G^* was 10.0 (resp. 50.0). For each case, we randomly generated n users from the whole graph G^*, and extracted a graph $G = (V, E)$ with the n users. Then we estimated the number of triangles $f_\triangle(G)$ and the number of 2-stars $f_{2\star}(G)$. For triangles, we evaluated $\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, and $\mathsf{CentralLap}_\triangle$. For 2-stars, we evaluated $\mathsf{LocalLap}_{2\star}$ and $\mathsf{CentralLap}_{2\star}$. In $\mathsf{Local2Rounds}_\triangle$, we set $\varepsilon_1 = \varepsilon_2$. For $\tilde{d}_{max}$, we set $\tilde{d}_{max} = d_{max}$.

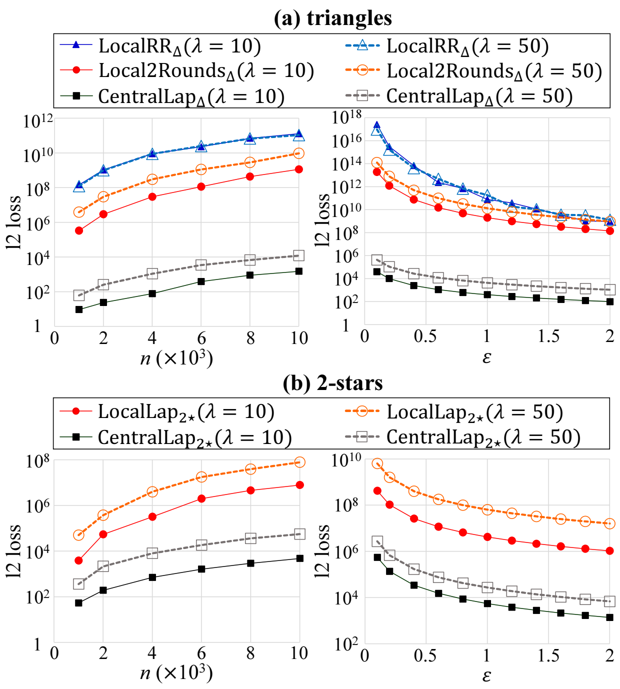


Figure 11: l_2 loss in the Barabási-Albert graph datasets (left: $\varepsilon = 1$, right: $n = 10000$). We set the attachment parameter λ in the BA model to $\lambda = 10$ or 50, and $\tilde{d}_{max}$ to $\tilde{d}_{max} = d_{max}$.

We evaluated the l_2 loss while changing n and ε. We attempted $\gamma \in \mathbb{N}$ ways to randomly select n users from G^*, and averaged the l_2 loss over all the γ ways to randomly select n users. As with Section 5, we set $\gamma = 100$ and changed n from 1000 to 10000 while fixing $\varepsilon = 1$. Then we set $\gamma = 10$ and changed ε from 0.1 to 2 while fixing $n = 10000$.

Experimental results. Figure 11 shows the results. Overall, Figure 11 has a similar tendency to Figures 5, 6, and 7. For example, Local2Rounds$_\triangle$ significantly outperforms LocalRR$_\triangle$, especially when the graph G is sparse; i.e., $\lambda = 10$. In Local2Rounds$_\triangle$, CentralLap$\triangle$, LocalLap$_2\star$, and CentralLap$_2\star$, the l_2 loss increases with increase in λ. This is because the maximum degree d_{max} $(= \tilde{d}_{max})$ increases with increase in λ.

Figure 11 also shows that the l_2 loss is roughly consistent with our upper-bounds in Section 4. For example, recall that LocalRR$_\triangle$, Local2Rounds$_\triangle$, CentralLap$_\triangle$, LocalLap$_{2\star}$, and CentralLap$_{2\star}$ achieve the expected l_2 loss of $O(n^4)$, $O(nd_{max}^3)$, $O(d_{max}^2)$, $O(nd_{max}^2)$, and $O(d_{max}^2)$, respectively. Assuming that $d_{max} = O(n)$, the left panels of Figure 11 are roughly consistent with these upper-bounds. In addition, the right panels of Figure 11 show that when we set $\lambda = 10$ and decrease ε from 0.4 to 0.1, the l_2 loss increases by a factor of about 3800, 250, and 16 in LocalRR$_\triangle$, Local2Rounds$_\triangle$, and CentralLap$_\triangle$, respectively. They are roughly consistent with our upper-bounds – for small ε, the expected l_2 loss of LocalRR$_\triangle$, Local2Rounds$_\triangle$, and CentralLap$_\triangle$ is $O(\varepsilon^{-6})$, $O(\varepsilon^{-4})$, and $O(\varepsilon^{-2})$, respectively.

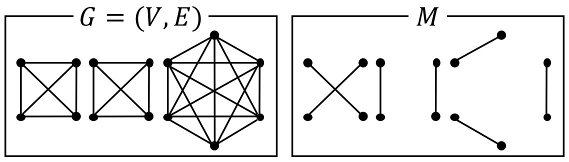


Figure 12: Examples of G and M for constructing an independent cube for $f_\triangle$ ($n = 14$, $d_{max} = 8$, $\eta_1 = 3$, $\eta_2 = 2$).

In summary, for both the two real datasets and the BA graphs, our experimental results showed the following findings: (1) Local2Rounds$_\triangle$ significantly outperforms LocalRR$_\triangle$, especially when the graph G is sparse; (2) our experimental results are roughly consistent with our upper-bounds.

C Construction of an $(n, \frac{d_{max}}{2} - 2)$ independent cube for $f_\triangle$

Suppose that n is even and d_{max} is divisible by 4. Since $d_{max} < n$, it is possible to write $n = \eta_1 \frac{d_{max}}{2} + \eta_2$ for integers η_1, η_2 such that $\eta_1 \geq 1$ and $1 \leq \eta_2 < \frac{d_{max}}{2}$. Because $\eta_1 \frac{d_{max}}{2}$ and n are even, we must have η_2 is even. Now, we can write $n = (\eta_1 - 1)\frac{d_{max}}{2} + (\eta_2 + \frac{d_{max}}{2})$. Thus, we can define a graph $G = (V, E)$ on n nodes consisting of $(\eta_1 - 1)$ cliques of even size $\frac{d_{max}}{2}$ and one final clique of an even size $\eta_2 + \frac{d_{max}}{2} \in (\frac{d_{max}}{2}, d_{max})$ with all cliques disjoint.

Since $G = (V, E)$ consists of even-sized cliques, it contains a perfect matching M. Figure 12 shows examples of G and M, where $n = 14$, $d_{max} = 8$, $\eta_1 = 3$, and $\eta_2 = 2$. Let $G' = (V, E')$ such that $E' = E \setminus M$. Let $\mathcal{A} = \{(V, E' \cup N : N \subseteq M\}$. Each edge in G is part of at least $\frac{d_{max}}{2} - 2$ triangles. For each pair of edges in M, the triangles of G of which they are part are disjoint. Thus, for any edge $e \in M$, removing e from a graph in $\mathcal{A}$ will remove at least $\frac{d_{max}}{2} - 2$ triangles. This implies that $\mathcal{A}$ is an $(n, \frac{d_{max}}{2} - 2)$ independent cube for $f_\triangle$.