Title
Private Graph Statistics and Algorithms in Modern Applications

Author
Imola, Jacob

Publication Date
2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Private Graph Statistics and Algorithms in Modern Applications

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Jacob John Imola

Committee in charge:

Professor Kamalika Chaudhuri, Chair
Professor Sanjoy Dasgupta
Professor Russell Impagliazzo
Professor Molly Roberts

2023

The Dissertation of Jacob John Imola is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

DEDICATION

This work is dedicated to my family. First, to my parents Martha and Mario, who inspired me to strive to be my best from a young age. Second, to my sister Lauren and brother David, for helping and supporting me along the way. Much love to you all!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I extend my sincerest gratitude to my advisor, Professor Kamalika Chaudhuri, for making this entire journey possible. I didn't have much direction or confidence when I arrived at UCSD, but Kamalika saw potential in me and guided me into research. When something finally stuck, Kamalika let me fly more freely and allowed me to develop my own style. She has also provided invaluable connections and collaborations throughout the years, without which many of my research projects would not have been realized.

Second, I greatly appreciate my office mates in lab 4142 with whom I have laughed, learned, opened up, and shared so many meaningful experiences. My PhD certainly wouldn't have been possible without this group. In particular, I would like to extend thanks to Mary Anne Smart, who has helped me with many problems related to work and to life, and who I can always count on to be a voice of reason. Next, thanks to Casey Meehan for his light-heartedness and generosity, especially during the more challenging times. Next, thanks to Robi Bhattacharjee from whose uncanny intelligence and varied interests I have learned lots. Finally, thanks to Amrita Roy Chowdhury, whose extreme aptitude in research and other intellectual pursuits I find quite inspiring and have taught me lots.

Third, I am very grateful to my collaborators over the years. My collaboration with Takao Murakami has been my most successful one, in no small part due to his amazing work ethic and keen eye for interesting research projects. It has been an absolute pleasure to work with him. Thanks to Abhinav Aggarwal, my collaborator and manager during my time at Amazon, for setting up an interesting, enriching research project during my time there. Thanks to Alessandro Epasto, my collaborator/manager at Google, who also set up an amazing research project during my time at Google. Finally, I have benefitted greatly from working with Amrita Roy Chowdhury, who has an incredible amount of knowledge and research expertise.

Lastly, I am so grateful to my friends and family. To my friends from both UCSD and from Carnegie Mellon: we have kept each other in our lives for a reason, and I hope to share many more fun times with you all in the future. To my family: thanks for loving and supporting

me throughout my life.

Chapter 1, in full, is a reprint of the material as it appears in Jacob Imola, Takao Murakami, and Kamalika Chaudhuri, USENIX Security Symposium, 2021. “Locally Differentially Private Analysis of Graph Statistics.” The dissertation author was a joint first author of this paper.

Chapter 2, in full, is a reprint of the material as it appears in Jacob Imola, Takao Murakami, and Kamalika Chaudhuri, 31st USENIX Security Symposium, 2022. “Communication-Efficient Triangle Counting under Local Differential Privacy.” The dissertation author was a joint first author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in Jacob Imola, Takao Murakami, and Kamalika Chaudhuri, Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022. “Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model.” The dissertation author was a joint first author of this paper.

Chapter 4, in full, is currently being prepared for submission for publication of the material. Jacob Imola, Amrita Roy Chowdhury, and Kamalika Chaudhuri. “Robustness of Locally Differentially Private Graph Analysis Under Data Poisoning Attacks.” The dissertation author was the first author of this paper.

Chapter 5, in full, is currently being prepared for submission for publication of the material. Jacob Imola, Alessandro Epasto, Mohammad Mahdian, Vincent Cohen-Addad, and Vahab Mirrokni. “Differentially Private Hierarchical Clustering with Provable Approximation Guarantees.” The dissertation author was the first author of this paper.

VITA

2018	Bachelor of Arts, Carnegie Mellon University
2018-2023	Graduate Research Assistant, University of California San Diego
2021	Research Intern, Amazon, Arlington, VA.
2022	Research Intern, Google N.Y.
2023	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

- Jacob Imola, Amrita Roy Chowdhury, and Kamalika Chaudhuri. *Robustness of Locally Differentially Private Graph Analysis Against Poisoning*. Preprint, 2023.
- Jacob Imola, Alessandro Epasto, Mohammad Mahdian, Vincent Cohen-Addad, and Vahab Mirrokni. *Differentially-Private Hierarchical Clustering with Provable Approximation Guarantees*. In ICML 2023.
- Robi Bhattacharjee, Jacob Imola, Michal Moshkovitz, and Sanjoy Dasgupta. *Online k-means Clustering on Arbitrary Data Streams*. In ALT 2023.
- Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. *Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model*. In CCS 2022.
- Jacob Imola, Shiva Kasiviswanathan, Stephen White, Abhinav Aggarwal, Nathanael Teissier. *Balancing Utility and Scalability in Metric Differential Privacy*. In UAI 2022.
- Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. *Communication-Efficient Triangle Counting under Local Differential Privacy*. In USENIX Security 2022.
- Jacob Imola and Kamalika Chaudhuri. *Privacy Amplification Via Bernoulli Sampling*. In TPDP Workshop at ICML 2021.
- Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. *Locally Differentially Private Analysis of Graph Statistics*. In USENIX Security 2021.
- Kamalika Chaudhuri, Jacob Imola, and Ashwin Machanavajjhala. *Capacity Bounded Differential Privacy*. In NeurIPS 2019.

ABSTRACT OF THE DISSERTATION

Private Graph Statistics and Algorithms in Modern Applications

by

Jacob John Imola

Doctor of Philosophy in Computer Science

University of California San Diego, 2023

Professor Kamalika Chaudhuri, Chair

In many modern machine learning applications, users hold data on a local device and communicate with an untrusted central analyst. Local differential privacy (local DP) is the state-of-the art way to ensure that all data sent by a user will be protected regardless of who eventually sees the data. In this dissertation, I will focus on collecting certain graph statistics, such as the degree vector and number of triangles in the graph, under local DP. I will obtain formal performance guarantees in the presence of different real-world constraints.

First, I will detail a one-round triangle counting algorithm which is simple, but not guaranteed to work well on sparse graphs. To improve these matters, I will propose a two-round algorithm with improved performance for sparse graphs. Unfortunately, it may have prohibitively

large per-user download cost. Second, I will show how one can use edge sampling to reduce download cost, and exhibit another two-round algorithm which trades off between the download cost and error.

Third, given that it requires significant synchronization overhead to implement multiple rounds, I will focus on improved one-round algorithms for triangle counting. I propose such an algorithm which satisfies a slightly weaker notion of DP, the shuffled model. This algorithm uses a novel wedge-shuffling approach to count wedges, and gives rise to an accurate algorithm for counting 4-cycles as well. Fourth, I will shift gears to computing the degree vector in the presence of malicious users who do not follow the local DP protocol. I will detail robust algorithms, leveraging the fact that every edge in a graph is shared between two users, which use edge consistency checks to catch malicious users.

Fifth, I will highlight my work on private hierarchical clustering, a common clustering algorithm used for graphs and other data. I propose two algorithms and a lower bound in the general setting of the problem, and a near-optimal algorithm in the stochastic block model of graphs. The algorithm design and lower bound are of interest to designing future local DP protocols on graphs.

Introduction

In our data-driven world, an increasing amount of data is collected by personal devices surrounding us, such as personal phones and home assistants. This data contains valuable information, but it is usually sensitive, opening the door for privacy attacks. With increased pressure from users and policy-makers to provide better privacy [198], it is becoming standard to not let the sensitive data leave the devices, and to protect users with privacy-preserving schemes.

The state-of-the-art way of protecting user privacy is through local differential privacy (DP), which stipulates that each part of a user's data will not affect the outcome of the server's analysis, regardless of what the analysis is and of any side-information about the user. A large body of work beginning with [121] exists on local differential privacy for tabular data, in which users hold i.i.d. samples from some distribution. We will explore local DP in the much less well-studied area of graph data, which is a common data type used in various types of networks.

For example, we might consider a dataset which consists of recent calls made between users, where phone numbers are public and the calls are private. Local DP is capable of collecting call information present on the phones while ensuring a notion of privacy to the calls themselves. Similarly, the recent rise of decentralized social media networks such as Mastodon [140] and Diaspora [60] is a good candidate for local DP, since there is no trusted central authority to conduct analysis.

In this dissertation, I advance the study of data analysis of graphs under local DP, considering additional constraints imposed in modern applications such as download cost, number of rounds of interaction, and data poisoning.

In Chapters 1, to 3, I will focus on algorithms for subgraph counting. In Chapter 1, I will detail a one-round triangle counting algorithm which is simple, but not guaranteed to work well on sparse graphs. To improve these matters, I then propose a two-round algorithm with better performance for sparse graphs which uses a second round counting query to reduce the noise introduced by privacy. Then, I empirically validate the performance of both algorithms. I conclude this chapter by proving a lower bound for any one-round algorithm for triangle counting. This lower bound was subsequently improved by [77] to show that the one-round algorithm is tight for some graphs.

In Chapter 2, I will show how to reduce the download cost of the two-round algorithm, which can often be prohibitively high, using edge sampling techniques. Using the sampling factor, one can trade off between the error of the algorithm and the download cost. The end result is an algorithm for counting triangles accurately in which users download just a tiny portion of the graph—in our experiments on graphs with millions of nodes, users merely needed to download data on the order of 100MB.

In Chapter 3, given that it requires significant synchronization overhead to implement multiple rounds which is not always possible in practice, I revisit one-round triangle counting algorithms. I propose an improved one-round algorithm which satisfies a slightly weaker notion of DP, the shuffle model. This algorithm uses a novel wedge-shuffling approach to count wedges, and gives rise to an accurate algorithm for counting 4-cycles as well. In experiments, these algorithms outperform the existing one-round, local DP algorithms.

In Chapter 4, I will shift gears to data poisoning in which malicious users who do not follow the local DP protocol send poisoned responses. I will demonstrate that some ubiquitous local DP mechanisms are susceptible to such users. Then, I will detail robust algorithms, leveraging the fact that every edge in a graph is shared between two users, which use edge consistency checks to catch malicious users. Experimentally, I validate the improved robustness of these algorithms.

In Chapter 5, I will describe my work on private hierarchical clustering, a common clustering algorithm used for graphs and other data. I propose two algorithms and a lower bound in the general setting of the problem, and a further algorithm in the stochastic block model of graphs. Though the results are in the central, not local, model of DP, the algorithm design and lower bounds are of interest to designing future local DP protocols on graphs.

Chapter 1

Locally Differentially Private Analysis of Graph Statistics

1.1 Introduction

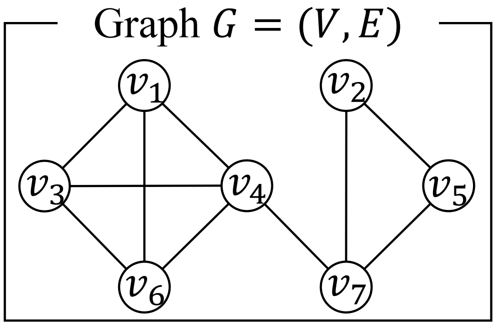
Analysis of network statistics is a useful tool for finding meaningful patterns in graph data, such as social, e-mail, citation and epidemiological networks. For example, the average *degree* (i.e., number of edges connected to a node) in a social graph can reveal the average connectivity. *Subgraph 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 [158]. 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 [168, 73, 70]. 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* [31, 43, 58, 98, 120, 122, 160, 167, 168, 184, 205, 203], 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 [120, 122, 184], degree distribution [58, 98, 167], spectra [205]) and synthetic graphs [43, 203].

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 [153, 178]. Additionally, *decentralized social networks* [161, 176] (e.g., Diaspora [60]) 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) [67, 121]. 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 personal data, it does not suffer from data leakage issues. Therefore, LDP has recently attracted attention from both academia [3, 19, 18, 86, 114, 117, 155, 164, 201, 217] as well as industry [196, 62, 189]. 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 1.2 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 1.1 shows an example of



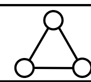
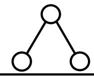
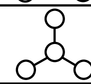
Shape	Name	Count
	Triangle	5
	2-star	20
	3-star	8

Figure 1.1. Example of subgraph counts.

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 1.1, $\frac{3\times 5}{20} = 0.75$).

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 1.1 knows all edges connected to v_2 and all edges between $v_3, \ldots, v_7$ 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 1.1.

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) [208] and empirical estimation [114, 155, 201]. 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 [190].

1.2 Related Work

Graph DP. DP on graphs has been widely studied, with most prior work being in the centralized model [31, 43, 58, 98, 120, 122, 160, 167, 168, 184, 205, 203]. In this model, a number of algorithms have been proposed for releasing subgraph counts [120, 122, 184], degree distributions [58, 98, 167], eigenvalues and eigenvectors [205], and synthetic graphs [43, 203].

There has also been a handful of work on graph algorithms in the local DP model [165, 186, 218, 219, 221]. For example, Qin *et al.* [165] propose an algorithm for generating synthetic graphs. Zhang *et al.* [221] 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.* [186] 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 [186] cannot be applied to this setting.

Ye *et al.* [218] 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 [219], they propose a method for reducing the bias in the estimate of triangle counts. However, the method in [219] 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.1 that our one-round algorithm significantly outperforms the one-round algorithm in [218]. Moreover, we show in Section 1.5 that our two-rounds algorithm significantly outperforms our one-round algorithm.

Our work also differs from [186, 218, 219] 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[3, 86, 114, 117, 155, 201, 217], heavy hitters [19, 18, 164]) 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 1.1). For example, the RR with empirical estimation is optimal in the low privacy regimes for estimating a distribution for tabular data [114, 117]. 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 [3, 114, 117, 217, 111, 110] and lower-bounds [2, 68, 66, 111, 65, 110, 112] 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 1.1). Rashtchian *et al.* [166] 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.

1.3 Preliminaries

1.3.1 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 $\mathscr{G}$ be the set of possible graphs G on n users. A graph $G \in \mathscr{G}$ 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) [73, 70] 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* [98, 168]. Edge DP considers two neighboring graphs $G, G' \in \mathscr{G}$ that differ in one edge. In contrast, node DP considers two neighboring graphs $G, G' \in \mathscr{G}$ in which G' is obtained from G by adding or removing one node along with its adjacent edges.

Although Zhang *et al.* [221] 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 [165, 186, 218, 219].

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 $\mathscr{M}$ with domain $\mathscr{G}$ provides* ε-edge centralized DP *if for any two neighboring graphs $G, G' \in \mathscr{G}$ that differ in one edge and any $S \subseteq \mathrm{Range}(\mathscr{M})$,*

$$\Pr[\mathscr{M}(G) \in S] \leq e^{\varepsilon} \Pr[\mathscr{M}(G') \in S]. \tag{1.1}$$

Edge centralized DP guarantees that an adversary who has observed the output of $\mathscr{M}$ 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 [73]. Specifically, if $\mathscr{M}$ provides ε-edge centralized DP, then for any two graphs $G, G' \in \mathscr{G}$ that differ in k edges and any $S \subseteq \mathrm{Range}(\mathscr{M})$, we obtain: $\Pr[\mathscr{M}(G) \in S] \leq e^{k\varepsilon} \Pr[\mathscr{M}(G') \in S]$; i.e., k edges are protected with privacy budget $k\varepsilon$.

1.3.2 Local Differential Privacy

LDP (Local Differential Privacy) [121, 67] 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 [165] and *relationship DP* introduced in this paper. Below, we will explain these two definitions in detail.

Edge LDP. Qin *et al.* [165] 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 [165])**.** *Let* $\varepsilon \in \mathbb{R}_{\geq 0}$. *For any* $i \in [n]$, *let* $\mathscr{R}_i$ *with domain* $\{0,1\}^n$ *be a randomized algorithm of user* v_i. $\mathscr{R}_i$ *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* $S \subseteq \mathrm{Range}(\mathscr{R}_i)$,

$$\Pr[\mathscr{R}_i(\mathbf{a}_i) \in S] \leq e^{\varepsilon} \Pr[\mathscr{R}_i(\mathbf{a}'_i) \in S]. \tag{1.2}$$

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 $\mathscr{R}_i$ providing ε-edge LDP, we explain Warner's RR (Randomized Response) [208] applied to a neighbor list, which is called the randomized neighbor list in [165].

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 $\Pr[\mathscr{R}(\mathbf{a}_i) \in S]$ and $\Pr[\mathscr{R}(\mathbf{a}'_i) \in S]$ in (1.2) 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 $(\mathscr{R}_1, \dots, \mathscr{R}_n)$, each of which is with domain $\{0,1\}^n$, provides* ε-relationship DP *if for any two neighboring graphs $G, G' \in \mathscr{G}$ that differ in one edge and any $S \subseteq \mathrm{Range}(\mathscr{R}_1) \times \ldots \times \mathrm{Range}(\mathscr{R}_n)$,*

$$\begin{aligned} &\Pr[(\mathscr{R}_1(\mathbf{a}_1), \dots, \mathscr{R}_n(\mathbf{a}_n)) \in S] \\ &\leq e^{\varepsilon} \Pr[(\mathscr{R}_1(\mathbf{a}'_1), \dots, \mathscr{R}_n(\mathbf{a}'_n)) \in S], \end{aligned} \tag{1.3}$$

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 [186] 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 $\mathscr{R}_1, \ldots, \mathscr{R}_n$ provide ε-edge LDP, then $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ 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 [73], Proposition 1 holds. $\square$

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 $\mathscr{R}_i$ 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 $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ outputs only the lower triangular part of the noisy adjacency matrix. Then all of $\mathscr{R}_1, \ldots, \mathscr{R}_n$ 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, $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ provides ε-relationship DP (not 2ε).

Our proposed algorithm in Section 1.4.3 also has this property; i.e., it provides both ε-edge LDP and ε-relationship DP.

1.3.3 Global Sensitivity

In this paper, we use the notion of global sensitivity [73] to provide edge centralized DP or edge LDP.

Let $\mathscr{D}$ be the set of possible input data of a randomized algorithm. In edge centralized DP, $\mathscr{D} = \mathscr{G}$. In edge LDP, $\mathscr{D} = \{0,1\}^n$. Let $f : \mathscr{D} \to \mathbb{R}$ be a function that takes data $D \in \mathscr{D}$ 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 [73].

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

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

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* [58, 122, 167], 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) [162, 188].

1.3.4 Graph Statistics and Utility Metrics

Graph statistics. We consider a graph function that takes a graph $G \in \mathscr{G}$ as input and outputs some graph statistics. Specifically, let $f_\triangle : \mathscr{G} \to \mathbb{Z}_{\geq 0}$ be a graph function that outputs the number of triangles in G. For $k \in \mathbb{N}$, let $f_{k\star} : \mathscr{G} \to \mathbb{Z}_{\geq 0}$ be a graph function that outputs the number of k-stars in G. For example, if a graph G is as shown in Figure 1.1, 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 1.1 shows the basic notations used in this paper.

Utility metrics. We use the l_2 loss (i.e., squared error) [114, 201, 155] and the relative error [29, 41, 215] 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 $f_{k\star}(G)$,

Table 1.1. 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).
$\mathscr{G}$	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).
$\mathscr{R}_i$	Randomized algorithm on $\mathbf{a}_i$.
$f_\triangle(G)$	Number of triangles in G.
$f_{k\star}(G)$	Number of k-stars in G.

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 [114, 201, 155].

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 [29, 41, 215], we set $\eta = 0.001n$ for $f_\triangle$ and $f_{k\star}$.

1.4 Algorithms

In the local model, there are several ways to model how the data collector interacts with the users [67, 112, 165]. The simplest model would be to assume that the data collector sends a query $\mathscr{R}_i$ to each user v_i once, and then each user v_i independently sends an answer $\mathscr{R}_i(\mathbf{a}_i)$ 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 1.3.2 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 [186] or more accurate synthetic graphs [165]. We call this the *multiple-rounds LDP model*.

In Sections 1.4.1 and 1.4.2, we consider the problems of computing $f_{k\star}(G)$ and $f_{\triangle}(G)$ for a graph $G \in \mathscr{G}$ 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 1.4.3, 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 1.4.4, 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 1.4 are given in Appendix A.4.

1.4.1 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.

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 [58, 122, 167] 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.

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 $\quad \mathbf{a}_i \leftarrow \texttt{GraphProjection}(\mathbf{a}_i, \tilde{d}_{max})$;
$\quad$ /* d_i is a degree of user v_i. */
4 $\quad d_i \leftarrow \sum_{j=1}^{n} a_{i,j}$;
5 $\quad r_i \leftarrow \binom{d_i}{k}$;
6 $\quad \hat{r}_i \leftarrow r_i + \mathrm{Lap}\left(\frac{\Delta}{\varepsilon}\right)$;
7 $\quad release(\hat{r}_i)$;
8 **end**
9 **return** $\sum_{i=1}^{n} \hat{r}_i$

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

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 $\mathsf{LocalLap}_{k\star}$.

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 [210]. 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 [84], $\tilde{d}_{max} = 5000$ 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 1.4.1.

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

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

Theorem 2. *Let $\hat{f}_{k\star}(G,\varepsilon,\tilde{d}_{max})$ be the output of $\mathsf{LocalLap}_{k\star}$. Then, for all $k \in \mathbb{N}, \varepsilon \in \mathbb{R}_{\geq 0}, \tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, and $G \in \mathscr{G}$ such that the maximum degree d_{max} of G is at most $\tilde{d}_{max}$, we have*

$$\mathbb{E}[l_2^2(\hat{f}_{k\star}(G,\varepsilon,\tilde{d}_{max}), f_{k\star}(G))] = O\left(\frac{n\tilde{d}_{max}^{2k-2}}{\varepsilon^2}\right).$$

The factor of n in the expected l_2 loss of $\mathsf{LocalLap}_{k\star}$ 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 $\mathsf{CentralLap}_{k\star}$. 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 $\mathscr{A} \subseteq \mathscr{G}$ on n nodes such that the maximum degree of each $G \in \mathscr{A}$ is at most $\tilde{d}_{max}$, and* $\frac{1}{|\mathscr{A}|}\sum_{G\in\mathscr{A}} \mathbb{E}[l_2^2(\hat{f}_{k\star}(G,\tilde{d}_{max},\varepsilon), f_{k\star}(G))] \geq \Omega\left(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2}\tilde{d}_{max}^{2k-2}n\right)$.

This is a corollary of a more general result of Section 1.4.4. 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., $\hat{d}_{max} = \max\{\hat{d}_1, \dots, \hat{d}_n\}$. 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 [73], this two-rounds algorithm provides ε-edge LDP in total ($\varepsilon = \varepsilon_0 + \varepsilon_1$).

In our experiments, we show that this algorithm provides the utility close to $\mathsf{LocalLap}_{k\star}$

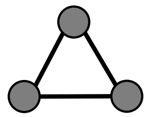
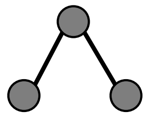
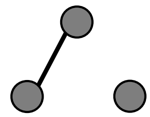
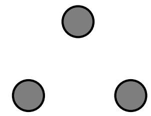
Shape	triangle 	2-edges 	1-edge 	no-edges 
Count in noisy graph G'	m_3	m_2	m_1	m_0

Figure 1.2. Four types of subgraphs with three nodes.

with the true max degree d_{max}.

1.4.2 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 1.3.2. Then the data collector constructs a noisy graph $G' = (V, E') \in \mathscr{G}$ 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 [114, 155, 201], 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 1.2 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*

$$\mathbb{E}\left[\frac{e^{3\varepsilon}}{(e^{\varepsilon}-1)^3}m_3 - \frac{e^{2\varepsilon}}{(e^{\varepsilon}-1)^3}m_2 + \frac{e^{\varepsilon}}{(e^{\varepsilon}-1)^3}m_1 - \frac{1}{(e^{\varepsilon}-1)^3}m_0\right] = f_{\triangle}(G). \tag{1.4}$$

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 (1.4). In Appendix A.1, we show that the l_2 loss is significantly reduced by this empirical estimation.

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

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

$$R_i = (RR_{\varepsilon}(a_{i,1}), \ldots, RR_{\varepsilon}(a_{i,i-1})).$$

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 $(R_1, \ldots, R_n)$ 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 (1.4), which is an unbiased estimator for $f_\triangle(G)$ by Proposition 2. We denote this algorithm by $\mathsf{LocalRR}_\triangle$.

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

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

$\mathsf{LocalRR}_\triangle$ 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 1.3.2.

Unlike the RR and empirical estimation for tabular data [114], the expected l_2 loss of $\mathsf{LocalRR}_\triangle$ 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 $\mathsf{LocalRR}_\triangle$. 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 $\mathsf{LocalRR}_\triangle$, and do not assume this model for the other algorithms. The upper-bound of $\mathsf{LocalRR}_\triangle$ in Theorem 4 is less ideal than the upper-bounds of the other algorithms in that it does not consider all possible graphs $G \in \mathscr{G}$. Nevertheless, we also show that the l_2 loss of $\mathsf{LocalRR}_\triangle$ is roughly consistent with Theorem 4 in our experiments using two real datasets (Section 1.5) and the Barabási-Albert graphs [16], which have power-law degree distribution (Appendix A.2).

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 $\mathsf{LocalRR}_\triangle$ ($= O(n^4)$) is much larger than the l_2 loss of $\mathsf{LocalRR}_{k\star}$ ($= O(n)$). 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 $\mathsf{CentralLap}_\triangle$. $\mathsf{CentralLap}_\triangle$ attains the expected l_2 loss ($=$ variance) of $O\left(\frac{\tilde{d}_{max}^2}{\varepsilon^2}\right)$.

The large l_2 loss of $\mathsf{LocalRR}_\triangle$ 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.

1.4.3 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 1.4.2, 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 $\mathsf{LocalRR}_\triangle$ 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 $\mathsf{LocalLap}_{k\star}$. 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 $\mathsf{LocalRR}_\triangle$). This is an intuition behind our proposed two-rounds algorithm.

As with the RR in Section 1.4.2, 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*

$$\mathbb{E}\left[\frac{1}{1-2p_1}\sum_{i=1}^n w_i\right] = f_\triangle(G). \tag{1.5}$$

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 (1.5), and sends it to the data collector. Then the data collector calculates an unbiased estimate of $f_\triangle(G)$ by (1.5).

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 $\mathsf{Local2Rounds}_\triangle$.

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

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

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 $f_\triangle(G)$.

$p_1 \leftarrow \frac{1}{e^{\varepsilon_1}+1}$;

`/* First round. */`

for $i = 1$ **to** n **do**

  $R_i \leftarrow (RR_{\varepsilon_1}(a_{i,1}), \ldots, RR_{\varepsilon_1}(a_{i,i-1}))$;

  $release(R_i)$;

end

$G' = (V, E') \leftarrow \texttt{UndirectedGraph}(R_1, \ldots, R_{i-1})$;

`/* Second round. */`

for $i = 1$ **to** n **do**

  $\mathbf{a}_i \leftarrow \texttt{GraphProjection}(\mathbf{a}_i, \tilde{d}_{max})$;

  $t_i \leftarrow |\{(v_i, v_j, v_k) : j < k < i, a_{i,j} = a_{i,k} = 1, (v_j, v_k) \in E'\}|$;

  $s_i \leftarrow |\{(v_i, v_j, v_k) : j < k < i, a_{i,j} = a_{i,k} = 1\}|$;

  $w_i \leftarrow t_i - p_1 s_i$;

  $\hat{w}_i \leftarrow w_i + \mathrm{Lap}(\frac{\tilde{d}_{max}}{\varepsilon_2})$;

  $release(\hat{w}_i)$;

end

return $\frac{1}{1-2p_1} \sum_{i=1}^{n} \hat{w}_i$

Algorithm 3: $\mathsf{Local2Rounds}_\triangle$

As with $\textsf{LocalRR}_\triangle$, $\textsf{Local2Rounds}_\triangle$ 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 $\textsf{Local2Rounds}_\triangle$. Then, for all $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, and $G \in \mathscr{G}$ 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 1.4.1. Furthermore, the private calculation of d_{max} does not increase the number of rounds; i.e., we can run $\textsf{Local2Rounds}_\triangle$ 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 $\textsf{Local2Rounds}_\triangle$ 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, $\textsf{LocalLap}_{k\star}$ provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP and $(\varepsilon_1 + \varepsilon_2)$-relationship DP (Theorem 5). Then by the composition theorem [73], 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 $\textsf{Local2Rounds}_\triangle$

with the true max degree d_{max}.

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

Specifically, $\mathsf{LocalRR}_\triangle$ 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 [5]). 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.

[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.

	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 $G \sim \mathbf{G}(n,\alpha)$)	$O\left(\frac{e^{\varepsilon}}{(e^{\varepsilon}-1)^2}\left(d_{max}^3 n + \frac{e^{\varepsilon}}{\varepsilon^2} d_{max}^2 n\right)\right)$

Table 1.2. 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 $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$.

1.4.4 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

$$\hat{f}(G) = \tilde{f}(\mathscr{R}_1(\mathbf{a}_1), \ldots, \mathscr{R}_n(\mathbf{a}_n)), \tag{1.6}$$

where $\mathscr{R}_1, \ldots, \mathscr{R}_n$ satisfy ε-edge LDP or ε-relationship DP and $\tilde{f}$ is an aggregate function that takes $\mathscr{R}_1(\mathbf{a}_1), \ldots, \mathscr{R}_n(\mathbf{a}_n)$ as input and outputs $\hat{f}(G)$. Here we assume that $\mathscr{R}_1, \ldots, \mathscr{R}_n$ 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:

Definition 5. *[(n,D)-independent cube for f] Let $D \in \mathbb{R}_{\geq 0}$. For $\kappa \in \mathbb{N}$, let $G = (V,E) \in \mathscr{G}$ be a graph on $n = 2\kappa$ nodes, and let $M = \{(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}), \ldots, (v_{i_{2k-1}}, v_{i_{2\kappa}})\}$ 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 $\mathscr{A} = \{(V, E \cup N) : N \subseteq M\}$. Note that $\mathscr{A}$ is a set of 2^κ graphs. We say $\mathscr{A}$ is an (n,D)-*independent cube for f *if for all $G' = (V,E') \in \mathscr{A}$, 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 $e \in M$.

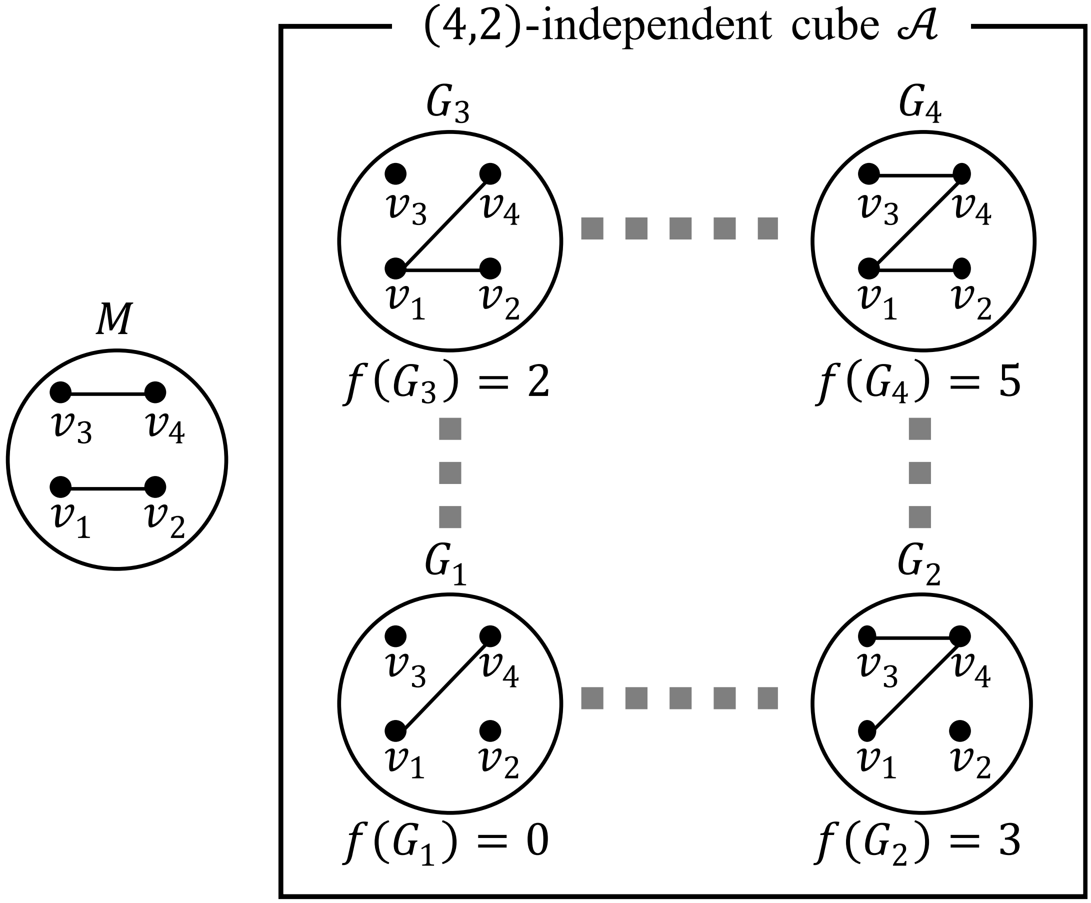


Figure 1.3. $(4,2)$-independent cube $\mathscr{A}$ for given function f. In this example, $M = \{(v_1,v_2),(v_3,v_4)\}$, $G_1 = (V,E)$, $\mathscr{A} = \{(V, E \cup N) : N \subseteq M\}$, $C_{(v_1,v_2)} = 2$, and $C_{(v_3,v_4)} = 3$. Adding (v_1,v_2) and (v_3,v_4) increase f by 2 and 3, respectively.

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 1.3 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 [89]. 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,

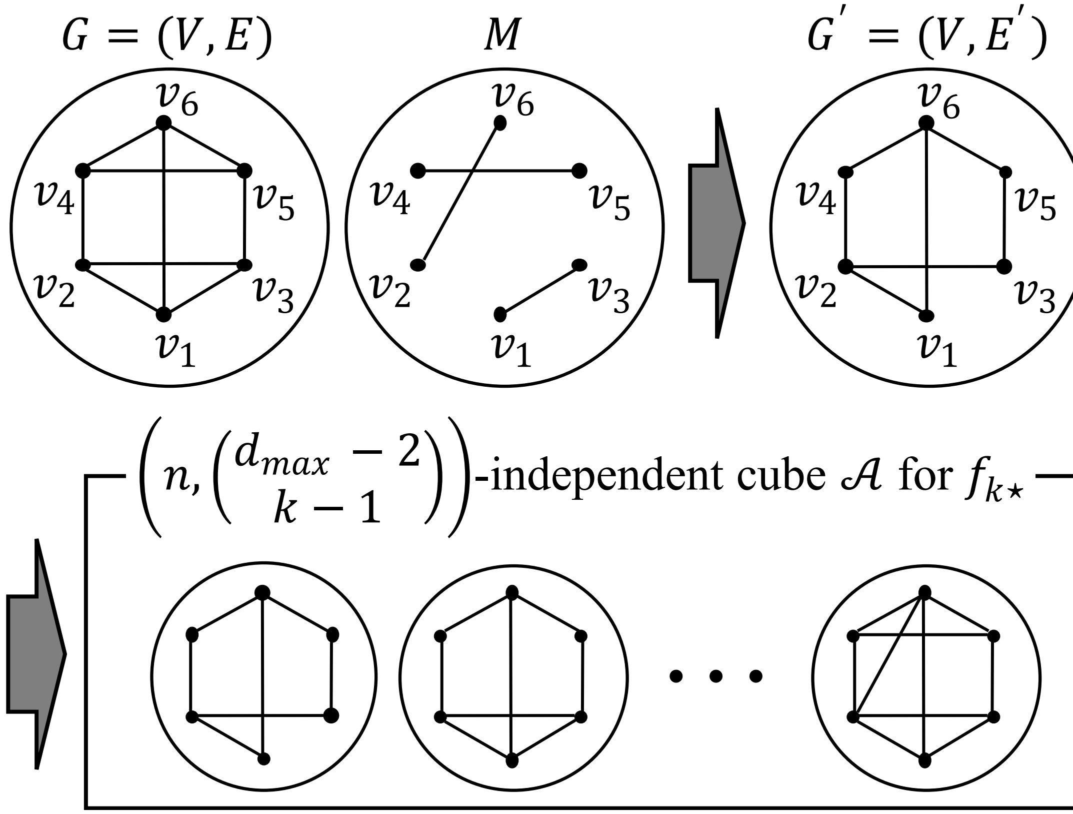


Figure 1.4. Construction of an independent cube for a k-star function ($n = 6$, $d_{max} = 4$). From a 3-regular graph $G = (V,E)$ and $M = \{(v_1,v_3),(v_2,v_6),(v_4,v_5)\}$, we make a graph $G' = (V,E')$ such that $E' = E \setminus M$. Then $\mathscr{A} = \{(V,E' \cup N) : N \subseteq M\}$ forms an $(n, 2\binom{d_{max}-2}{k-1})$-independent cube for $f_{k\star}$.

$\mathscr{A} = \{(V,E' \cup N) : N \subseteq M\}$ 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 $\mathscr{A}$ is at most d_{max}. Figure 1.4 shows how to construct an independent cube for a k-star function when $n = 6$ 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* (1.6)*, where $\mathscr{R}_1, \dots, \mathscr{R}_n$ are independently run. Let $\mathscr{A}$ be an (n,D)-independent cube for f. If $(\mathscr{R}_1, \dots, \mathscr{R}_n)$ provides ε-relationship DP, then we have*

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

A corollary of Theorem 7 is that if $\mathscr{R}_1, \dots, \mathscr{R}_n$ 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 A.3, we also construct an $(n, \frac{d_{max}}{2} - 2)$ independent cube for $f_\triangle$ and establish a lower bound of $\Omega(\frac{e^{2\varepsilon}}{(e^{2\varepsilon}+1)^2} nd_{max}^2)$ for $f_\triangle$.

The upper and lower bounds on the l_2 losses shown in this section appear in Table 1.2.

1.5 Experiments

Based on our theoretical results in Section 1.4, 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.

1.5.1 Experimental Set-up

We used the following two large-scale datasets:

IMDB. The Internet Movie Database (denoted by IMDB) [1] 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) [129] 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 1.4. 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., $\mathsf{LocalRR}_\triangle$ in Section 1.4.2), (2) the two-rounds algorithm in the local model (i.e., $\mathsf{Local2Rounds}_\triangle$ in Section 1.4.3), and (3) the Laplacian mechanism in the centralized model (i.e., $\mathsf{CentralLap}\triangle$ in Section 1.4.2).

Algorithms for k-stars. For algorithms for estimating $f_{k\star}(G)$, we used the following two algorithms: (1) the Laplacian mechanism in the local model (i.e., $\mathsf{LocalLap}_k\star$ in Section 1.4.1) and (2) the Laplacian mechanism in the centralized model (i.e., $\mathsf{CentralLap}_k\star$ in Section 1.4.1).

For each algorithm, we evaluated the l_2 loss and the relative error (as described in Section 1.3.4), 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 A.2, we also report experimental results using artificial graphs based on the Barabási-Albert model [16].

1.5.2 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 1.5 and 1.6 shows the results ($\varepsilon = 1$). Here we did not evaluate $\mathsf{LocalRR}_\triangle$ when n was larger than 10000, because $\mathsf{LocalRR}_\triangle$ was inefficient (as described in Section 1.4.3 "Time complexity"). In $\mathsf{Local2Rounds}_\triangle$, 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 1.5.2.

Figure 1.5 shows that $\mathsf{Local2Rounds}_\triangle$ significantly outperforms $\mathsf{LocalRR}_\triangle$. Specifically, the l_2 loss of $\mathsf{Local2Rounds}_\triangle$ is smaller than that of $\mathsf{LocalRR}_\triangle$ by a factor of about 10^2. The difference between $\mathsf{Local2Rounds}_\triangle$ and $\mathsf{LocalRR}_\triangle$ is larger in Orkut. This is because Orkut is more sparse, as described in Section 1.5.1. 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 $\mathsf{Local2Rounds}_\triangle$ and $\mathsf{LocalRR}_\triangle$ can be expressed as $O(nd_{max}^3)$ and $O(n^4)$, respectively. Thus $\mathsf{Local2Rounds}_\triangle$ significantly outperforms $\mathsf{LocalRR}_\triangle$, especially in sparse graphs.

Figures 1.5 and 1.6 show that the l_2 loss is roughly consistent with our upper-bounds in terms of n. Specifically, $\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, $\mathsf{CentralLap}_\triangle$, $\mathsf{LocalLap}_{k\star}$, and $\mathsf{CentralLap}_{k\star}$ 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 1.5 and 1.6 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 1.7 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 $\mathsf{Local2Rounds}_\triangle$, we set $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$.

Figure 1.7 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 about 5000, 200, and 16 for both the datasets in $\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, and $\mathsf{CentralLap}_\triangle$, respectively. They are roughly consistent with our theoretical results that for small ε, the expected l_2 loss of

$\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, and $\mathsf{CentralLap}_\triangle$ is $O(\varepsilon^{-6})$[2], $O(\varepsilon^{-4})$, and $O(\varepsilon^{-2})$, respectively.

Figure 1.7 also shows that $\mathsf{Local2Rounds}_\triangle$ significantly outperforms $\mathsf{LocalRR}_\triangle$ especially when ε is small, which is also consistent with our theoretical results. Conversely, the difference between $\mathsf{LocalRR}_\triangle$ and $\mathsf{Local2Rounds}_\triangle$ 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, $\mathsf{LocalRR}_\triangle$ with such a large value of ε does not guarantee strong privacy, because it outputs the true value in most cases. $\mathsf{Local2Rounds}_\triangle$ significantly outperforms $\mathsf{LocalRR}_\triangle$ when we want to estimate $f_\triangle(G)$ or $f_{k\star}(G)$ with a strong privacy guarantee; e.g., $\varepsilon \leq 1$ [131].

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 1.3.4.

Figure 1.8 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 $\mathsf{Local2Rounds}_\triangle$ and $\mathsf{LocalLap}_{k\star}$ for estimating $f_\triangle(G)$ and $f_{k\star}(G)$, respectively (we did not use $\mathsf{Local2RR}_\triangle$, because it is both inaccurate and inefficient). For both algorithms, we set $\varepsilon = 1$ or 2 ($\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$ in $\mathsf{Local2Rounds}_\triangle$) and $\tilde{d}_{max} = d_{max}$. Then 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 $\mathsf{CentralLap}_\triangle$ and $\mathsf{CentralLap}_{k\star}$ ($\varepsilon = 1$ or 2, $\tilde{d}_{max} = d_{max}$) and calculated the clustering coefficient in the same way.

Figure 1.8 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 $f_{k\star,v_i}(G) =$

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

$\sum_{i=1}^{n} f_{k\star,v_i}(G)$. 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, $\mathsf{Local2Rounds}_{\triangle}$, $\mathsf{LocalLap}_{k\star}$, $\mathsf{CentralLap}_{\triangle}$, and $\mathsf{CentralLap}_{k\star}$ 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 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 1.4.1 and 1.4.3).

Specifically, we used $\mathsf{Local2Rounds}_{\triangle}$ and $\mathsf{LocalLap}_{k\star}$ 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 1.4.1 and 1.4.3.

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 $\mathsf{Local2Rounds}_{\triangle}$ or $\mathsf{LocalLap}_{k\star}$. In $\mathsf{Local2Rounds}_{\triangle}$, 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 1.8.

Figure 1.9 shows the results. Figure 1.9 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 1.1, 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.

1.6 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 [120].

Chapter 1, in full, is a reprint of the material as it appears in Jacob Imola, Takao Murakami, and Kamalika Chaudhuri, USENIX Security Symposium, 2021. "Locally Differentially Private Analysis of Graph Statistics." The dissertation author was a joint first author of this paper.

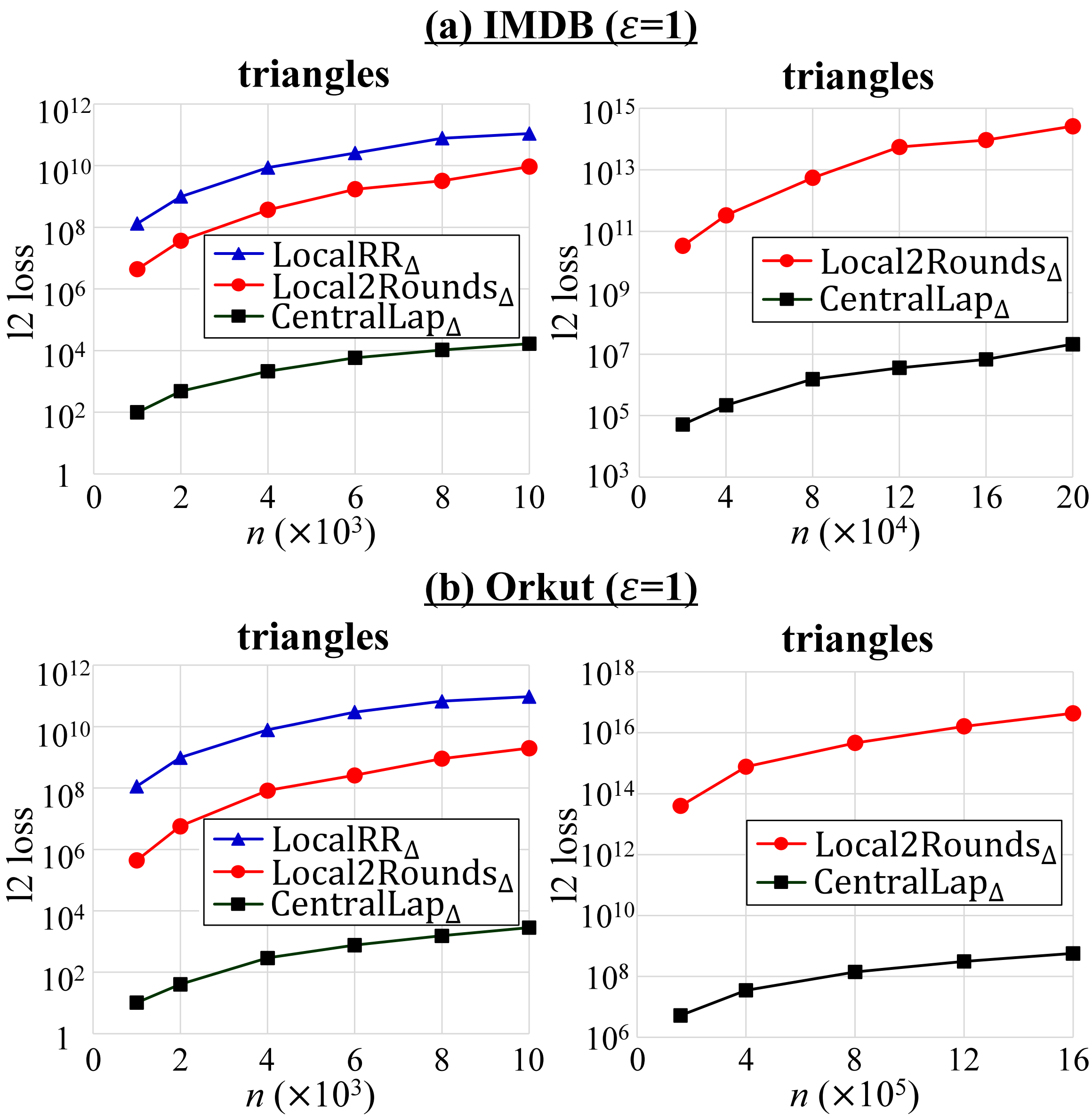


Figure 1.5. Relation between the number of users n and the l_2 loss in triangle counts when $\varepsilon = 1$ ($\varepsilon_1 = \varepsilon_2 = \frac{1}{2}$, $\tilde{d}_{max} = d_{max}$). Here we do not evaluate LocalRR$_\triangle$ when $n > 10000$, because it is inefficient (see Section 1.4.3 “Time complexity”).

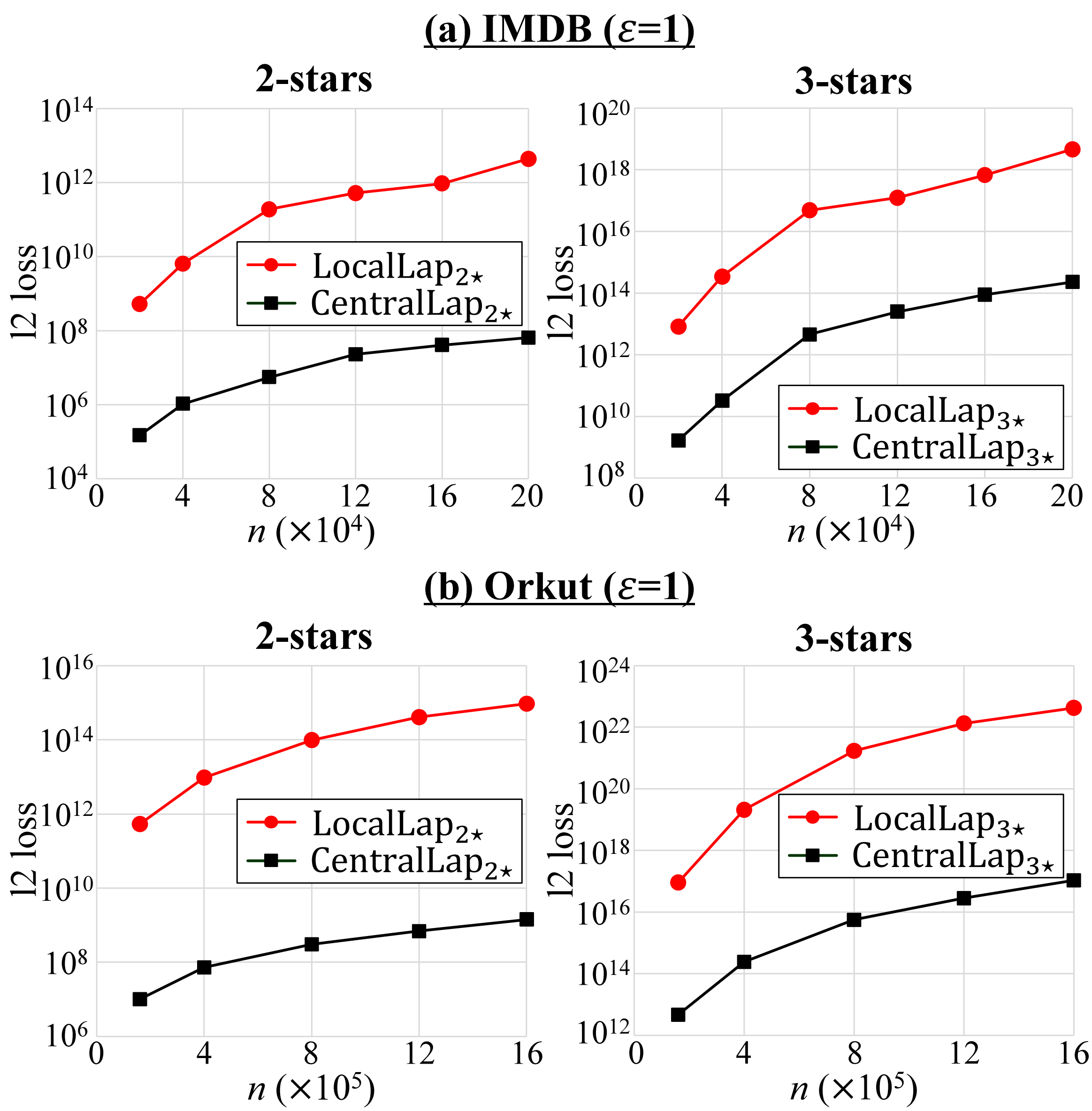


Figure 1.6. Relation between the number of users n and the l_2 loss in k-star counts when $\varepsilon = 1$ ($\varepsilon_1 = \varepsilon_2 = \frac{1}{2}$, $\tilde{d}_{max} = d_{max}$).

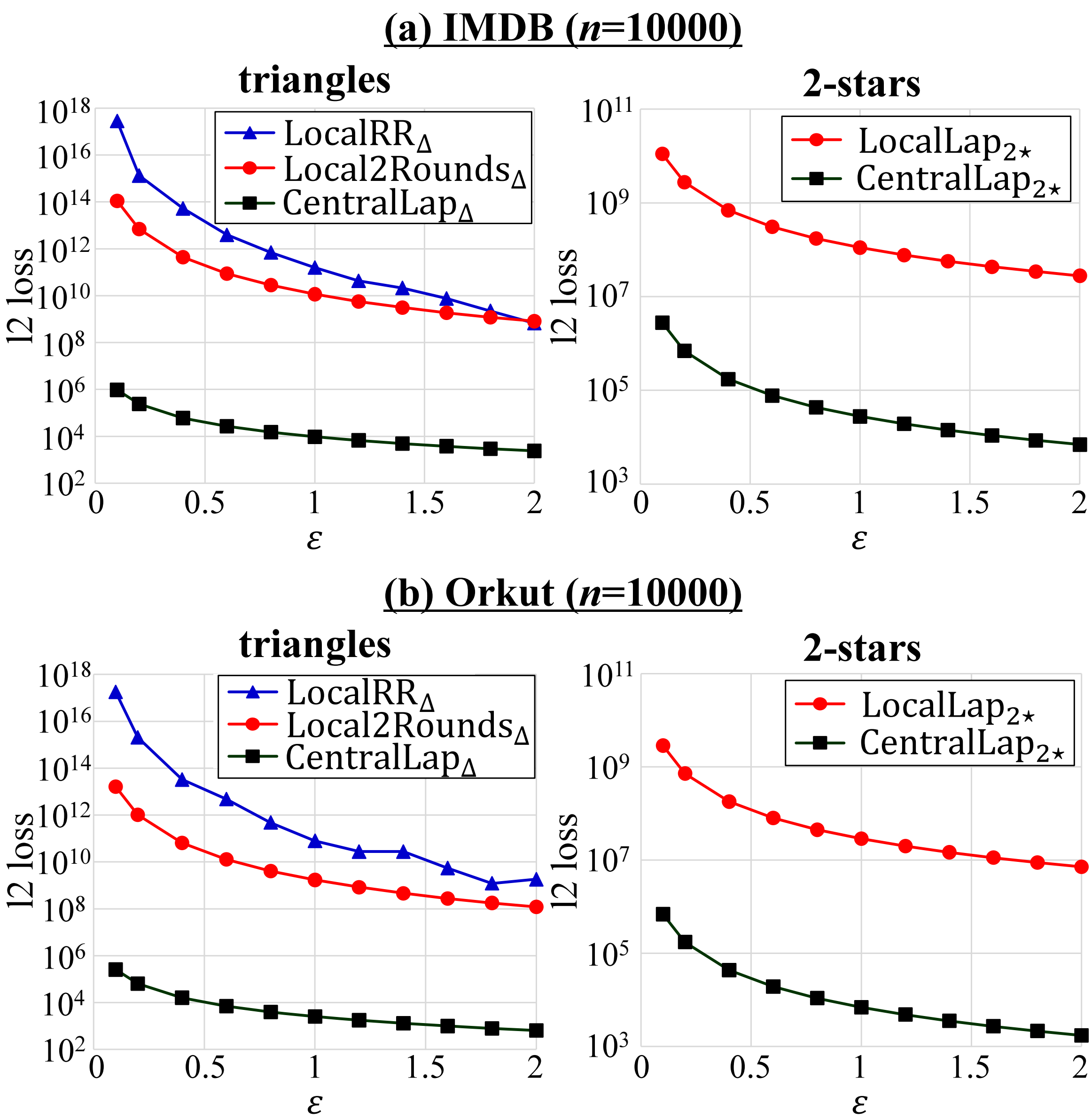


Figure 1.7. Relation between ε in edge LDP and the l_2 loss when $n = 10000$ ($\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$, $\tilde{d}_{max} = d_{max}$).

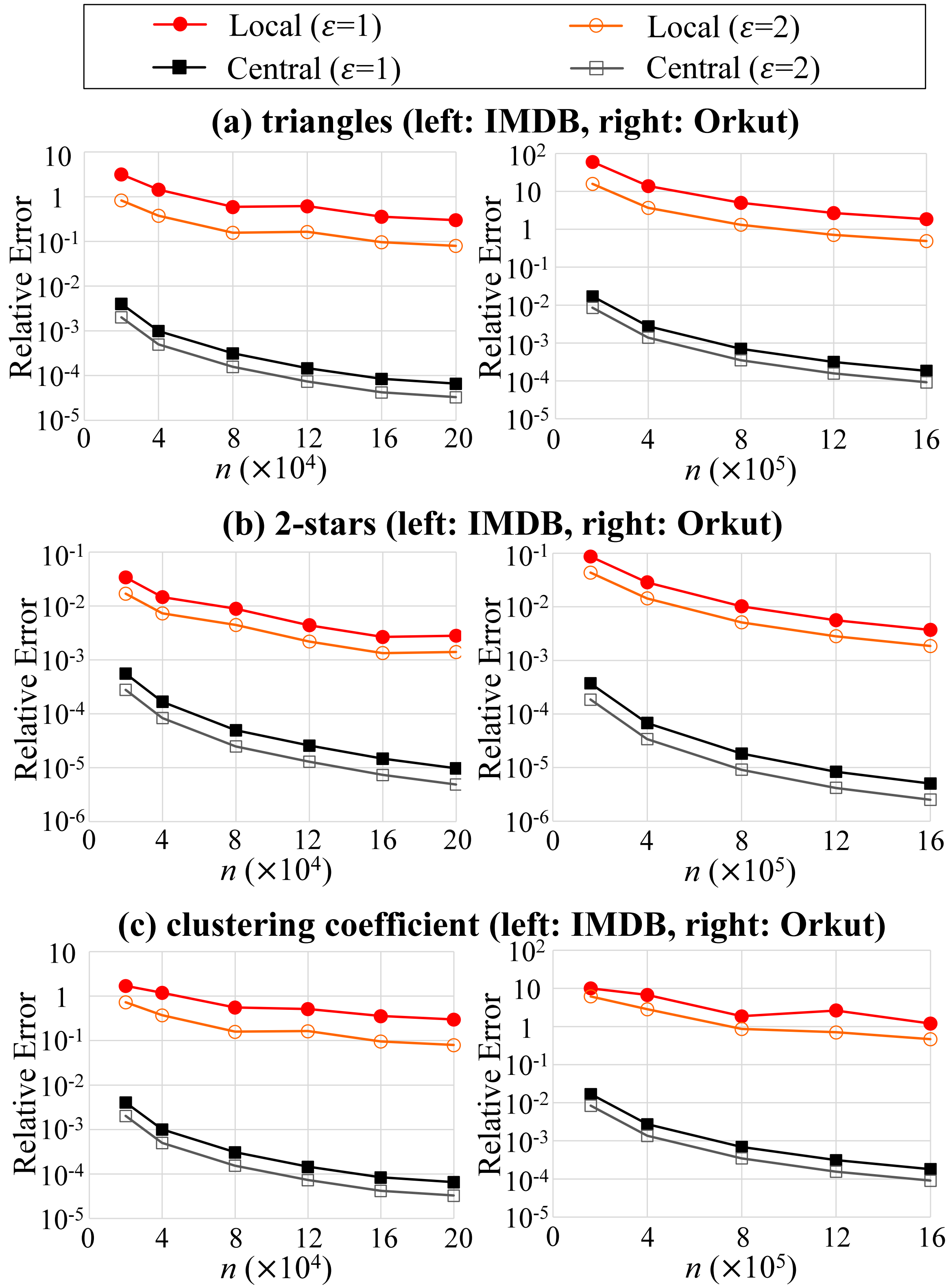


Figure 1.8. Relation between n and the relative error. In the local model, we used $\mathsf{Local2Rounds}_{\triangle}$ ($\varepsilon = 1$ or 2) and $\mathsf{LocalLap}_{k\star}$ ($\varepsilon = 1$ or 2) for estimating triangle counts $f_{\triangle}(G)$ and k-star counts $f_{k\star}(G)$, respectively ($\tilde{d}_{max} = d_{max}$).

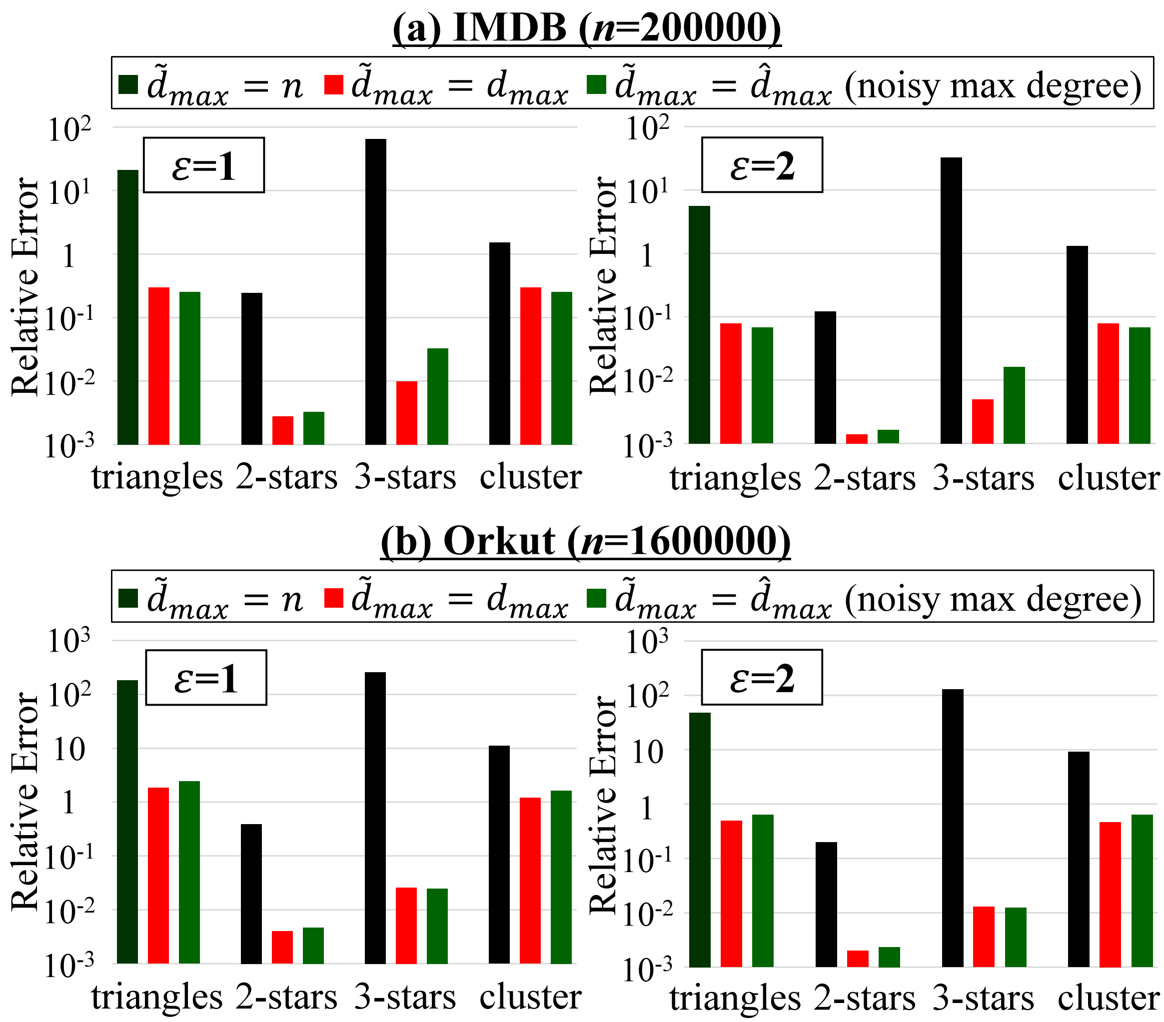


Figure 1.9. Relative error when $\tilde{d}_{max} = n$ (#users), d_{max} (max degree), or $\hat{d}_{max}$ (noisy max degree). We used Local2Rounds$_\triangle$ ($\varepsilon = 1$ or 2) and LocalLap$_{k\star}$ ($\varepsilon = 1$ or 2) for estimating triangle counts $f_\triangle(G)$ and k-star counts $f_{k\star}(G)$, respectively.

Chapter 2

Communication-Efficient Triangle Counting under Local Differential Privacy

2.1 Introduction

Counting subgraphs (e.g., triangles, stars, cycles) is one of the most basic tasks for analyzing connection patterns in various graph data, e.g., social, communication, and collaboration networks. For example, a triangle is given by a set of three nodes with three edges, whereas a k-star is given by a central node connected to k other nodes. These subgraphs play a crucial role in calculating a *clustering coefficient* ($= \frac{3\times\#\text{triangles}}{\#\text{2-stars}}$) (see Figure 2.1). The clustering coefficient measures the average probability that two friends of a user will also be a friend in a social graph [158]. Therefore, it is useful for measuring the effectiveness of friend suggestions. In addition, the clustering coefficient represents the degree to which users tend to cluster together. Thus, if it is large in some services/communities, we can effectively apply social recommendations [125] to the users. Triangles and k-stars are also useful for constructing graph models [170, 109]; see also [194] for other applications of triangle counting. However, graph data often involve sensitive data such as sensitive edges (friendships), and they can be leaked from exact numbers of triangles and k-stars [102].

To analyze subgraphs while protecting user privacy, DP (Differential Privacy) [73] has been widely adopted as a privacy metric [64, 102, 120, 186, 218, 219, 223]. DP protects user privacy against adversaries with arbitrary background knowledge and is known as a gold standard for data privacy. According to the underlying model, DP can be categorized into *central (or global) DP* and *LDP (Local DP)*. Central DP assumes a scenario where a central server has personal data of all users. Although accurate analysis of subgraphs is possible under this model [64, 120, 223], there is a risk that the entire graph is leaked from the server by illegal access or internal fraud [152, 35]. In addition, central DP cannot be applied to *decentralized social networks* [60, 140, 148, 161] where the entire graph is distributed across many servers. We can even consider *fully decentralized applications* where a server does not have any original edge, e.g., a mobile app that sends a noisy degree (noisy number of friends) to the server, which then estimates a degree distribution. Central DP cannot be used in such applications.

In contrast, LDP assumes a scenario where each user obfuscates her personal data (friends list in the case of graphs) by herself and sends the obfuscated data to a possibly malicious server; i.e., it does not assume trusted servers. Thus, it does not suffer from a data breach and can also be applied to the decentralized applications. LDP has been widely studied in tabular data where each row corresponds to a user's personal data (e.g., age, browser setting, location) [3, 18, 196, 114, 155, 201] and also in graph data [102, 165, 218, 219]. For example, k-star counts can be very accurately estimated under LDP because each user can count k-stars of which she is a center and sends a noisy version of her k-star count to the server [102].

However, more complex subgraphs such as triangles are much harder to count under LDP because each user cannot see edges between other users. For example, in Figure 2.1, user v_1 cannot see edges between v_2, v_3, and v_6 and therefore cannot count triangles involving v_1. Thus, existing algorithms [102, 218, 219] obfuscate each user's edges (rather than her triangle count) by RR (Randomized Response) [208] and send noisy edges to a server. Consequently, the server suffers from a prohibitively large estimation error (e.g., relative error $> 10^2$ in large graphs, as shown in Appendix B.2) because all three edges are noisy in any noisy triangle the server sees.

A recent study [102] shows that the estimation error in locally private triangle counting is significantly reduced by introducing an additional round of interaction between users and the server. Specifically, if the server publishes the noisy graph (all noisy edges) sent by users at the

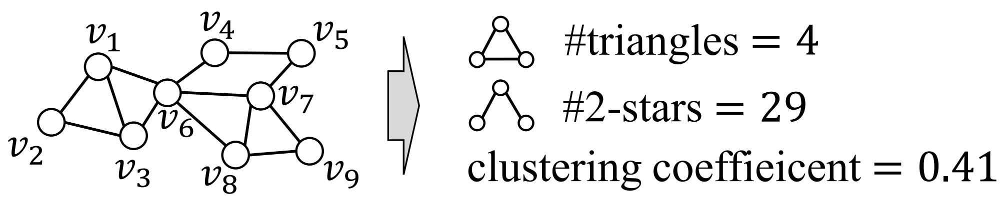


Figure 2.1. Triangles, 2-stars, and clustering coefficient.

first round, then each user can count her noisy triangles such that *only one edge* is noisy (as she knows two edges connected to her). Thus, the algorithm in [102] sends each user's noisy triangle count (with additional noise) to the server at the second round. Then the server can accurately estimate the triangle count. This algorithm also requires a much smaller number of interactions (i.e., only two) than collaborative approaches [113, 182] that generally require many interactions.

Unfortunately, the algorithm in [102] is still impractical for a large-scale graph. Specifically, the noisy graph sent by users is dense, hence extremely large for a large-scale graph, e.g., 500 Gbits for a graph of a million users. The problem is that *every user* needs to download such huge data; e.g., when the download speed is 20 Mbps (which is a recommended speed in YouTube [220]), every user needs about 7 hours to download the noisy graph. Since the communication ability might be limited for some users, the algorithm in [102] cannot be used for applications with large and diverse users.

In summary, existing triangle algorithms under LDP suffer from either a prohibitively large estimation error or a prohibitively high communication cost. They also suffer from the same issues when calculating the clustering coefficient.

Our Contributions. We propose locally private triangle counting algorithms with a small estimation error and small communication cost. Our contributions are as follows:

- We propose two-rounds triangle algorithms consisting of *edge sampling* after RR and *selecting edges each user downloads*. In particular, we show that a simple extension of [102] with edge sampling suffers from a large estimation error for a large or dense graph where the number of 4-cycles (such as v_1-v_2-v_3-v_6-v_1 in Figure 2.1) is large. To address this issue, we propose some strategies for selecting edges to download to reduce the error caused by the 4-cycles, which we call the *4-cycle trick*.

- We show that the algorithms with the 4-cycle trick still suffer from a large estimation error due to large Laplacian noise for each user. To significantly reduce the Laplacian noise, we propose a *double clipping* technique, which clips a degree (the number of edges) of each user with LDP and then clips the number of noisy triangles.

- We evaluate our algorithms using two real datasets. We show that our entire algorithms with the 4-cycle trick and double clipping dramatically reduce the communication cost of [102]. For example, for a graph with about 900000 users, we reduce the download cost from 400 Gbits (6 hours when 20 Mbps) to 160 Mbits (8 seconds) or less while keeping the relative error much smaller than 1.

Thus, locally private triangle counting is now much more practical. In Appendix B.3, we also show that we can estimate the clustering coefficient with a small estimation error and download cost. For example, our algorithms are useful for measuring the effectiveness of friend suggestions or social recommendations in decentralized social networks, e.g., Diaspora [60], Mastodon [140]. Our source code is available at [192].

All the proofs of our privacy and utility analysis are given in Appendices B.6, B.7, and B.8.

Technical Novelty. Below we explain more about the technical novelty of this paper. Although we focus on two-rounds local algorithms in the same way as [102], we introduce several new algorithmic ideas previously unknown in the literature.

First, our 4-cycle trick is totally new. Although some studies focus on 4-cycle counting [23, 118, 138, 143], this work is the first to use 4-cycles to improve communication efficiency. Second, selective download of parts of a centrally computed quantity is also new. This is not limited to graphs – even in machine learning, there are no such strategic download techniques previously, to our knowledge. Third, our utility analysis of our triangle algorithms (Theorem 9) is different from [102] in that ours introduces subgraphs such as 4-cycles and k-stars. This leads us to our 4-cycle trick. Fourth, we propose two triangle algorithms that introduce the 4-cycle trick and show that the more tricky one provides the best performance because of its low sensitivity in DP.

Finally, our double clipping is new. Andrew *et al.* [9] propose an adaptive clipping technique, which applies clipping twice. However, they focus on federated averaging, and their problem setting is different from our graph setting. In particular, they require a private quantile of the norm distribution. In contrast, we need only a much simpler estimate: a private degree. Here, we use the fact that the degree has a small sensitivity (sensitivity $= 1$) in DP for edges. We also provide a new, reasonably tight bound on the probability that the noisy triangle count exceeds a clipping threshold (Theorem 11). Thanks to the two differences, we obtain a significant communication improvement: two or three orders of magnitude.

2.2 Related Work

Triangle Counting. Triangle counting has been extensively studied in a non-private setting [25, 24, 49, 76, 180, 187, 193, 211] (it is almost a sub-field in itself) because it requires high time complexity for large graphs.

Edge sampling [24, 76, 193, 211] is one of the most basic techniques to improve scalability. Although edge sampling is simple, it is quite effective – it is reported in [211] that edge sampling outperforms other sampling techniques such as node sampling and triangle sampling. Based on this, we adopt edge sampling after RR[1] with new techniques such as the 4-cycle trick and double clipping. Our entire algorithms significantly improve the communication cost, as well as the space and time complexity, under LDP (see Sections 2.5.3 and 2.6).

[1]We also note that a study in [159] proposes a graph publishing algorithm in the central model that independently changes 1-cells (edges) to 0-cells (no edges) with some probability and then changes a fixed number of 0-cells to 1-cells *without replacement*. However, each 0-cell is *not* independently sampled in this case, and consequently, their proof that relies on the independence of the noise to each 0-cell is incorrect. In contrast, our algorithms provide DP because we apply sampling after RR, i.e., post-processing.

DP on Graphs. For private graph analysis, DP has been widely adopted as a privacy metric. Most of them adopt central (or global) DP [58, 64, 98, 120, 122, 167, 223], which suffers from the data breach issue.

LDP on graphs has recently studied in some studies, e.g., synthetic data generation [165], subgraph counting [102, 186, 218, 219]. A study in [186] proposes subgraph counting algorithms in a setting where each user allows her friends to see all her connections. However, this setting is unsuitable for many applications; e.g., in Facebook, a user can easily change her setting so that her friends cannot see her connections.

Thus, we consider a model where each user can see only her friends. In this model, some one-round algorithms [218, 219] and two-rounds algorithms[102] have been proposed. However, they suffer from a prohibitively large estimation error or high communication cost, as explained in Section 2.1.

Recently proposed network LDP protocols [56] consider, instead of a central server, collecting private data with user-to-user communication protocols along a graph. They focus on sums, histograms, and SGD (Stochastic Gradient Descent) and do not provide subgraph counting algorithms. Moreover, they focus on hiding each user's private dataset rather than hiding an edge in a graph. Thus, their approach cannot be applied to our task of subgraph counting under LDP for edges. The same applies to another work [174] that improves the utility of an averaging query by correlating the noise of users according to a graph.

LDP. RR [114, 208] and RAPPOR [196] have been widely used for tabular data in LDP. Our work uses RR in part of our algorithm but builds off of it significantly. One noteworthy result in this area is HR (Hadamard Response) [3], which is state-of-the-art for tabular data and requires low communication. However, this result is not applied to graph data and does not address the communication issues considered in this paper. Specifically, applying HR to each bit in a neighbor list will result in $O(n^2)$ (n: #users) download cost in the same way as the previous work [102] that uses RR. Applying HR to an entire neighbor list (which has 2^n possible values) will similarly result in $O(n \log 2^n) = O(n^2)$ download cost.

Previous work on distribution estimation [114, 155, 201] or heavy hitters [18] addresses a different problem than ours, as they assume that every user has i.i.d. (independent and identically distributed) samples. In our setting, a user's neighbor list is non-i.i.d. (as one edge is shared by two users), which does not fit into their statistical framework.

2.3 Preliminaries

2.3.1 Notations

We begin with basic notations. Let $\mathbb{N}$, $\mathbb{R}$, $\mathbb{Z}_{\geq 0}$, and $\mathbb{R}_{\geq 0}$ be the sets of natural numbers, real numbers, non-negative integers, and non-negative real numbers, respectively. For $z \in \mathbb{N}$, let $[z]$ a set of natural numbers from 1 to z; i.e., $[z] = \{1, 2, \ldots, z\}$.

Let $G = (V, E)$ be an undirected graph, where V is a set of nodes and $E \subseteq V \times V$ is a set of edges. Let $n \in \mathbb{N}$ be the number of nodes in V. Let $v_i \in V$ be the i-th node; i.e., $V = \{v_1, \ldots, v_n\}$. We consider a social graph where each node in V represents a user and an edge $(v_i, v_j) \in E$ represents that v_i is a friend with v_j. Let $d_{max} \in \mathbb{N}$ be the maximum degree of G. Let $\mathscr{G}$ be a set of graphs with n nodes. Let $f_\triangle : \mathscr{G} \to \mathbb{Z}_{\geq 0}$ be a triangle count query that takes $G \in \mathscr{G}$ as input and outputs a triangle count $f_\triangle(G)$ (i.e., number of triangles) in G.

Let $\mathbf{A} = (a_{i,j}) \in \{0, 1\}^{n \times n}$ be a symmetric adjacency matrix corresponding to G; i.e., $a_{i,j} = 1$ if and only if $(v_i, v_j) \in E$. We consider a local privacy model [165, 102], where each user obfuscates her *neighbor list* $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,n}) \in \{0, 1\}^n$ (i.e., the i-th row of $\mathbf{A}$) using a *local randomizer* $\mathscr{R}_i$ with domain $\{0, 1\}^n$ and sends obfuscated data $\mathscr{R}_i(\mathbf{a}_i)$ to a server. We also assume a two-rounds algorithm in which user v_i downloads a message M_i from the server at the second round.

We also show the basic notations in Table B.1 of Appendix B.1.

2.3.2 Local Differential Privacy on Graphs

LDP on Graphs. When we apply LDP (Local DP) to graphs, we follow the direction of *edge DP* [160, 168] that has been developed for the central DP model. In edge DP, the existence of an edge between any two users is protected; i.e., two computations, one using a graph with the edge and one using the graph without the edge, are indistinguishable. There is also another privacy notion called *node DP* [98, 221], which hides the existence of one user along with all her edges. However, in the local model, many applications send a user ID to a server; e.g., each user sends the number of her friends along with her user ID. For such applications, we cannot use node DP but can use edge DP to hide her edges, i.e., friends. Thus, we focus on edge DP in the local model in the same way as [102, 165, 186, 218, 219].

Specifically, assume that user v_i uses her local randomizer $\mathscr{R}_i$. We assume that the server and other users can be honest-but-curious adversaries and that they can obtain all edges except for user v_i's edges as prior knowledge. Then we use the following definition for $\mathscr{R}_i$:

Definition 6 (ε-edge LDP [165])**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. For $i \in [n]$, let $\mathscr{R}_i$ be a local randomizer of user v_i that takes $\mathbf{a}_i$ as input. We say $\mathscr{R}_i$ 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 $s \in \mathrm{Range}(\mathscr{R}_i)$,*

$$\Pr[\mathscr{R}_i(\mathbf{a}_i) = s] \leq e^{\varepsilon} \Pr[\mathscr{R}_i(\mathbf{a}'_i) = s]. \tag{2.1}$$

For example, a local randomizer $\mathscr{R}_i$ that applies Warner's RR (Randomized Response) [208], which flips 0/1 with probability $\frac{1}{e^{\varepsilon}+1}$, to each bit of $\mathbf{a}_i$ provides ε-edge LDP.

The parameter ε is called the privacy budget. When ε is small (e.g., $\varepsilon \leq 1$ [131]), each bit is strongly protected by edge LDP. Edge LDP can also be used to hide *multiple bits* – by group privacy [73], two neighbor lists $\mathbf{a}_i, \mathbf{a}'_i \in \{0,1\}^n$ that differ in $k \in \mathbb{N}$ bits are indistinguishable up to the factor $k\varepsilon$.

Edge LDP is useful for protecting a neighbor list $\mathbf{a}_i$ of each user v_i. For example, a user in Facebook can change her setting so that anyone (except for the central server) cannot see her friend list $\mathbf{a}_i$. Edge LDP hides $\mathbf{a}_i$ even from the server.

As with regular LDP, the guarantee of edge LDP does not break even if the server or other users act maliciously. However, adding or removing an edge affects the neighbor list of two users. This means that each user needs to trust her friend to not reveal an edge between them. This also applies to Facebook – even if v_i keeps $\mathbf{a}_i$ secret, her edge with v_j can be disclosed if v_j reveals $\mathbf{a}_j$. To protect each edge during the whole process, we use another privacy notion called relationship DP [102]:

Definition 7 (ε-relationship DP [102])**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. For $i \in [n]$, let $\mathscr{R}_i$ be a local randomizer of user v_i that takes $\mathbf{a}_i$ as input. We say $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ provides* ε-relationship DP *if for any two neighboring graphs $G, G' \in \mathscr{G}$ that differ in one edge and any $(s_1, \ldots, s_n) \in \mathrm{Range}(\mathscr{R}_1) \times \ldots \times \mathrm{Range}(\mathscr{R}_n)$,*

$$
\begin{aligned}
&\Pr[(\mathscr{R}_1(\mathbf{a}_1), \ldots, \mathscr{R}_n(\mathbf{a}_n)) = (s_1, \ldots, s_n)] \\
&\leq e^{\varepsilon} \Pr[(\mathscr{R}_1(\mathbf{a}'_1), \ldots, \mathscr{R}_n(\mathbf{a}'_n)) = (s_1, \ldots, s_n)], \qquad (2.2)
\end{aligned}
$$

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').

If users v_i and v_j follow the protocol, (2.2) holds for graphs G, G' that differ in (v_i, v_j). Thus, relationship DP applies to all edges of a user whose neighbors are trustworthy.

While users need to trust other friends to maintain a relationship DP guarantee, only one edge per user is at risk for each malicious friend that does not follow the protocol. This is because only one edge can exist between two users. Thus, although the trust assumption in relationship DP is stronger than that of LDP, it is much weaker than that of central DP in which all edges can be revealed by the server.

It is possible to use a tuple of local randomizers with edge LDP to obtain a relationship DP guarantee:

Proposition 4 (Edge LDP and relationship DP [102])**.** *If each of local randomizers $\mathscr{R}_1, \ldots, \mathscr{R}_n$ provides ε-edge LDP, then $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ provides 2ε-relationship DP. Additionally, if each $\mathscr{R}_i$ uses only bits $a_{i,1}, \ldots, a_{i,i-1}$ for users with smaller IDs (i.e., only the lower triangular part of $\mathbf{A}$), then $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ provides ε-relationship DP.*

The doubling factor in ε comes from the fact that (2.2) applies to an entire edge, whereas (2.1) applies to just one neighbor list, and adding an entire edge may cause changes to two neighbor lists. However, if each $\mathscr{R}_i$ ignores bits $a_{i,i}, \ldots, a_{i,n}$ for users with larger IDs, then this doubling factor can be avoided. Our algorithms also use only the lower triangular part of $\mathbf{A}$ to avoid this doubling issue.

Interaction among Users and Multiple Rounds. While interaction in LDP has been studied before [112], neither of Definitions 6 and 7 allows the interaction among users in a one-round protocol where user v_i sends $\mathscr{R}_i(\mathbf{a}_i)$ to the server.

However, the interaction among users is possible in a multi-round protocol. Specifically, at the first round, user v_i applies a randomizer $\mathscr{R}_i^1$ and sends $\mathscr{R}_i^1(\mathbf{a}_i)$ to the server. At the second round, the server calculates a message M_i for v_i by performing some post-processing on $\mathscr{R}_i^1(\mathbf{a}_i)$, possibly with the private outputs by other users. Let λ_i be the post-processing algorithm on $\mathscr{R}_i^1(\mathbf{a}_i)$; i.e., $M_i = \lambda_i(\mathscr{R}_i^1(\mathbf{a}_i))$. The server sends M_i to v_i. Then, v_i uses a randomizer $\mathscr{R}_i^2(M_i)$ that depends on M_i and sends $\mathscr{R}_i^2(M_i)(\mathbf{a}_i)$ back to the server. This entire computation provides DP by a (general) sequential composition [131]:

Proposition 5 (Sequential composition of edge LDP)**.** *For $i \in [n]$, let $\mathscr{R}_i^1$ be a local randomizer of user v_i that takes $\mathbf{a}_i$ as input. Let λ_i be a post-processing algorithm on $\mathscr{R}_i^1(\mathbf{a}_i)$, and $M_i = \lambda_i(\mathscr{R}_i^1(\mathbf{a}_i))$ be its output. Let $\mathscr{R}_i^2(M_i)$ be a local randomizer of v_i that depends on M_i. If $\mathscr{R}_i^1$ provides ε_1-edge LDP and for any $M_i \in \mathrm{Range}(\lambda_i)$, $\mathscr{R}_i^2(M_i)$ provides ε_2-edge LDP, then the sequential composition $(\mathscr{R}_i^1(\mathbf{a}_i), \mathscr{R}_i^2(M_i)(\mathbf{a}_i))$ provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP.*

We provide a proof of Proposition 5 in Appendix B.6.

Global Sensitivity. We use the notion of global sensitivity [73] to provide edge LDP:

Definition 8. *In edge LDP (Definition 6), the global sensitivity of a function $f : \{0,1\}^n \rightarrow \mathbb{R}$ is given by:*

$$GS_f = \max_{\mathbf{a}_i, \mathbf{a}'_i \in \{0,1\}^n, \mathbf{a}_i \sim \mathbf{a}'_i} |f(\mathbf{a}_i) - f(\mathbf{a}'_i)|,$$

where $\mathbf{a}_i \sim \mathbf{a}'_i$ represents that $\mathbf{a}_i$ and $\mathbf{a}'_i$ differ in one bit.

For example, adding the Laplacian noise with mean 0 scale $\frac{GS_f}{\varepsilon}$ (denoted by $\mathrm{Lap}(\frac{GS_f}{\varepsilon})$) to $f(\mathbf{a}_i)$ provides ε-edge LDP.

2.3.3 Utility and Communication-Efficiency

Utility. We consider a private estimate of $f_\triangle(G)$. Our private estimator $\hat{f}_\triangle : \mathscr{G} \rightarrow \mathbb{R}$ is a post-processing of local randomizers $(\mathscr{R}_1, \dots, \mathscr{R}_n)$ that satisfy ε-edge LDP. Following previous work, we use the l_2 loss (i.e., squared error) [114, 201, 155] and the relative error [29, 41, 215] as utility metrics.

Specifically, let l_2^2 be the expected l_2 loss function on a graph G, which maps the estimate $\hat{f}_\triangle(G)$ and the true value $f_\triangle(G)$ to the expected l_2 loss; i.e., $l_2^2(f_\triangle(G), \hat{f}_\triangle(G)) = \mathbb{E}[(\hat{f}_\triangle(G) - f_\triangle(G))^2]$. The expectation is taken over the randomness in the estimator $\hat{f}$, which is necessarily a randomized algorithm since it satisfies edge LDP. In our theoretical analysis, we analyze the expected l_2 loss, as with [114, 201, 155].

Note that the l_2 loss is large when $f_\triangle(G)$ is large. Therefore, in our experiments, we use the relative error given by $\frac{|\hat{f}_\triangle(G) - f_\triangle(G)|}{\max\{f_\triangle(G), \eta\}}$, where $\eta \in \mathbb{R}_{\geq 0}$ is a small value. Following convention [29, 41, 215], we set η to $0.001n$. The estimate is very accurate when the relative error is much smaller than 1.

Communication-Efficiency. A prominent concern when performing local computations is that the computing power of individual users is often limited. Of particular concern to our private estimators, and a bottleneck of previous work in locally private triangle counting [102], is the communication overhead between users and the server. This communication takes the form of users *downloading* any necessary data required to compute their local randomizers and *uploading* the output of their local randomizers. We distinguish the two quantities because often downloading is cheaper than uploading.

Consider a τ-round protocol, where $\tau \in \mathbb{N}$. At round $j \in [\tau]$, user v_i applies a local randomizer $\mathscr{R}_i^j(M_i^j)$ to her neighbor list $\mathbf{a}_i$, where M_i^j is a message sent from the server to user v_i during round j. We define the *download cost* as the number of bits required to describe M_i^j and the *upload cost* as the number of bits required to describe $\mathscr{R}_i^j(M_i^j)(\mathbf{a}_i)$. Over all rounds and all users, we evaluate the *maximum per-user download/upload cost*, which is given by:

$$\mathrm{Cost}_{DL} = \max_{i=1}^{n} \sum_{j=1}^{\tau} \mathbb{E}[|M_i^j|] \quad \text{(bits)} \tag{2.3}$$

$$\mathrm{Cost}_{UL} = \max_{i=1}^{n} \sum_{j=1}^{\tau} \mathbb{E}[|\mathscr{R}_i^j(M_i^j)(\mathbf{a}_i)|] \quad \text{(bits).} \tag{2.4}$$

The above expectations go over the probability distributions of computing the local randomizers and any post-processing done by the server. We evaluate the maximum of the expected download/upload cost over users.

2.4 Communication-Efficient Triangle Counting Algorithms

The current state-of-the-art triangle counting algorithm [102] under edge LDP suffers from an extremely large per-user download cost; e.g., every user has to download a message of 400 Gbits or more when $n = 900000$. Therefore, it is impractical for a large graph. To address this issue, we propose three communication-efficient triangle algorithms under edge LDP.

We explain the overview and details of our proposed algorithms in Sections 2.4.1 and 2.4.2, respectively. Then we analyze the theoretical properties of our algorithms in Section 2.4.3.

2.4.1 Overview

Motivation. The drawback of the triangle algorithm in [102] is a prohibitively high download cost at the second round. This comes from the fact that in their algorithm, each user v_i applies

Warner's RR (Randomized Response) [208] to bits for smaller user IDs in her neighbor list $\mathbf{a}_i$ (i.e., lower triangular part of $\mathbf{A}$) and then downloads the whole noisy graph. Since Warner's RR outputs 1 (edge) with high probability (e.g., about 0.5 when ε is close to 0), the number of edges in the noisy graph is extremely large—about half of the $\binom{n}{2}$ possible edges will be edges.

In this paper, we address this issue by introducing two strategies: *sampling edges* and *selecting edges each user downloads*. First, each user v_i samples each 1 (edge) after applying Warner's RR. Edge sampling has been widely studied in a non-private triangle counting problem [24, 76, 193, 211]. In particular, Wu *et al.* [211] compare various non-private triangle algorithms (e.g., edge sampling, node sampling, triangle sampling) and show that edge sampling provides almost the lowest estimation error. They also formally prove that edge sampling outperforms node sampling. Thus, sampling edges after Warner's RR is a natural choice for our private setting.

Second, we propose three strategies for selecting edges each user downloads. The first strategy is to simply select all noisy edges; i.e., each user downloads the whole noisy graph in the same way as [102]. The second and third strategies select some edges (rather than all edges) in a more clever manner so that the estimation error is significantly reduced. We provide a more detailed explanation in Section 2.4.2.

Algorithm Overview. Figure 2.2 shows the overview of our proposed algorithms.

At the first round, each user v_i obfuscates bits for smaller user IDs in her neighbor list $\mathbf{a}_i$ by an LDP mechanism which we call the *ARR (Asymmetric Randomized Response)* and sends the obfuscated bits to a server. The ARR is a combination of Warner's RR and edge sampling; i.e., we apply Warner's RR that outputs 1 or 0 as it is with probability p_1 ($= \frac{e^\varepsilon}{e^\varepsilon+1}$) and then sample each 1 with probability $p_2 \in [0,1]$. Unlike Warner's RR, the ARR is asymmetric in that the flip probability in the whole process is different depending on the input value. As with Warner's RR, the ARR provides edge LDP. We can also significantly reduce the number of 1s (hence the communication cost) by setting p_2 small.

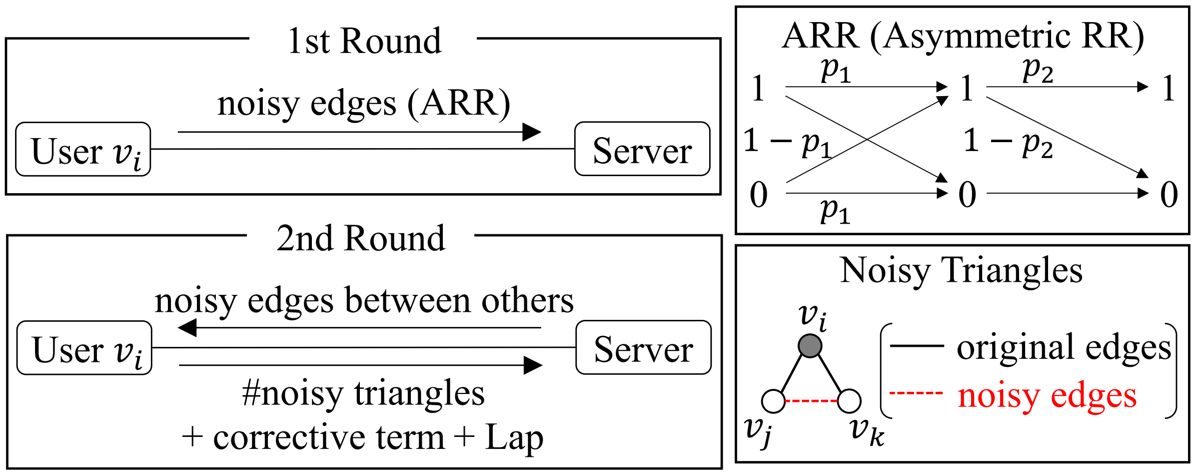


Figure 2.2. Overview of our communication-efficient triangle counting algorithms ($p_1 = \frac{e^\varepsilon}{e^\varepsilon+1}$, $p_2 \in [0,1]$).

At the second round, the server calculates a message M_i for user v_i consisting of some or all noisy edges between others. We propose three strategies for calculating M_i. User v_i downloads M_i from the server. Then, since user v_i knows her edges, v_i can count *noisy triangles* (v_i, v_j, v_k) such that $j < k < i$ and only one edge (v_j, v_k) is noisy, as shown in Figure 2.2. The condition $j < k < i$ is imposed to use only the lower triangular part of $\mathbf{A}$, i.e., to avoid the doubling issue in Section 2.3.2. User v_i adds a corrective term and the Laplacian noise to the noisy triangle count and sends it to a server. The corrective term is added to enable the server to obtain an unbiased estimate of $f_\triangle(G)$. The Laplacian noise provides edge LDP. Finally, the server calculates an unbiased estimate of $f_\triangle(G)$ from the noisy data sent by users. By composition (Proposition 5), our algorithms provide edge LDP in total.

Remark. Note that it is also possible for the server to calculate an unbiased estimate of $f_\triangle(G)$ at the first round. However, this results in a prohibitively large estimation error because all edges sent by users are noisy; i.e., three edges are noisy in any triangle. In contrast, only one edge is noisy in each noisy triangle at the second round because each user v_i knows two original edges connected to v_i. Consequently, we can obtain an unbiased estimate with a much smaller variance. See Appendix B.2 for a detailed comparison.

2.4.2 Algorithms

ARR. First, we formally define the ARR. The ARR has two parameters: $\varepsilon \in \mathbb{R}_{\geq 0}$ and $\mu \in [0, \frac{e^\varepsilon}{e^\varepsilon+1}]$. The parameter ε is the privacy budget, and μ controls the communication cost.

Let $ARR_{\varepsilon,\mu}$ be the ARR with parameters ε and μ. It takes $0/1$ as input and outputs $0/1$ with the following probability:

$$\Pr[ARR_{\varepsilon,\mu}(1) = b] = \begin{cases} \mu & (b = 1) \\ 1 - \mu & (b = 0) \end{cases} \tag{2.5}$$

$$\Pr[ARR_{\varepsilon,\mu}(0) = b] = \begin{cases} \mu\rho & (b = 1) \\ 1 - \mu\rho & (b = 0), \end{cases} \tag{2.6}$$

where $\rho = e^{-\varepsilon}$. By Figure 2.2, we can view this randomizer as a combination of Warner's RR [208] and edge sampling, where $\mu = p_1 p_2$. In fact, the ARR with $\mu = p_1 = \frac{e^\varepsilon}{e^\varepsilon+1}$ (i.e., $p_2 = 1$) is equivalent to Warner's RR.

Each user v_i applies the ARR to bits for smaller user IDs in her neighbor list $\mathbf{a}_i$; i.e., $\mathscr{R}_i(\mathbf{a}_i) = (ARR_{\varepsilon,\mu}(a_{i,1}), \ldots, ARR_{\varepsilon,\mu}(a_{i,i-1}))$. Then v_i sends $\mathscr{R}_i(\mathbf{a}_i)$ to the server. Since applying Warner's RR to $\mathbf{a}_i$ provides ε-edge LDP (as described in Section 2.3.2) and the sampling is a post-processing process, applying the ARR to $\mathbf{a}_i$ also provides ε-edge LDP by the immunity to post-processing [73].

Let $E' \subseteq V \times V$ be a set of noisy edges sent by users.

Which Noisy Edges to Download? Now, the main question tackled in this paper is: *Which noisy edges should each user v_i download at the second round?* Note that user v_i is not allowed to download only a set of noisy edges that form noisy triangles (i.e., $\{(v_j, v_k) \in E' | (v_i, v_j) \in E, (v_i, v_k) \in E\}$), because it tells the server who are friends with v_i. In other words, user v_i cannot leak her original edges to the server when she downloads noisy edges; the server must choose which part of E' to include in the message M_i it sends her.

Thus, a natural solution would be to download *all noisy edges between others* (with smaller user IDs); i.e., $M_i = \{(v_j, v_k) \in E' | j < k < i\}$. We denote our algorithm with this full download strategy by $\mathsf{ARRFull}_\triangle$. The (inefficient) two-rounds algorithm in [102] is a special case of $\mathsf{ARRFull}_\triangle$ without sampling ($\mu = p_1$). In other words, $\mathsf{ARRFull}_\triangle$ is a generalization of the two-rounds algorithm in [102] using the ARR.

In this paper, we show that we can do much better than $\mathsf{ARRFull}_\triangle$. Specifically, we prove in Section 2.4.3 that $\mathsf{ARRFull}_\triangle$ results in a high estimation error when the number of 4-cycles (cycles of length 4) in G is large. Intuitively, this can be explained as follows. Suppose that v_i, v_j, $v_{i'}$, and v_k ($j < k < i$, $j < k < i'$) form a 4-cycle. There is no triangle in this graph. However, if there is a noisy edge between v_j and v_k, then two (incorrect) noisy triangles appear: (v_i, v_j, v_k) counted by v_i and ($v_{i'}$, v_j, v_k) counted by $v_{i'}$. More generally, let E_{ijk} (resp. $E_{i'jk}$) $\in \{0,1\}$ be a random variable that takes 1 if (v_i, v_j, v_k) (resp. ($v_{i'}$, v_j, v_k)) forms a noisy triangle and 0 otherwise. Then, the covariance $\mathrm{Cov}(E_{ijk}, E_{i'jk})$ between E_{ijk} and $E_{i'jk}$ is large because the presence/absence of a single noisy edge (v_j, v_k) affects the two noisy triangles.

To address this issue, we introduce a trick that makes the two noisy triangles *less correlated with each other*. We call this the *4-cycle trick*. Specifically, we propose two algorithms in which the server uses noisy edges connected to v_i when it calculates a message M_i for v_i. In the first algorithm, the server selects noisy edges (v_j, v_k) such that one noisy edge is connected from v_k to v_i; i.e., $M_i = \{(v_j, v_k) \in E' | (v_i, v_k) \in E', j < k < i\}$. We call this algorithm $\mathsf{ARROneNS}_\triangle$, as one noisy edge is connected to v_i. In the second algorithm, the server selects noisy edges (v_j, v_k) such that two noisy edges are connected from these nodes to v_i; i.e., $M_i = \{(v_j, v_k) \in E' | (v_i, v_j) \in E', (v_i, v_k) \in E', j < k < i\}$. We call this algorithm $\mathsf{ARRTwoNS}_\triangle$, as two noisy edges are connected to v_i. Note that user v_i does not leak her original edges to the server at the time of download in these algorithms, because the server uses only noisy edges E' sent by users to calculate M_i.

Figure 2.3 shows our three algorithms. The download cost Cost_{DL} in (2.3) is $O(\mu n^2 \log n)$, $O(\mu^2 n^2 \log n)$, and $O(\mu^3 n^2 \log n)$, respectively, when we regard ε as a constant. In our experi-

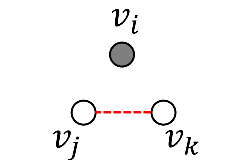
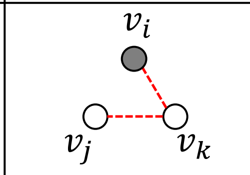
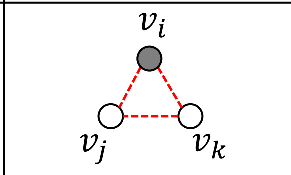
	ARRFull$_\Delta$	ARROneNS$_\Delta$	ARRTwoNS$_\Delta$
Noisy edges between others (v_j and v_k) to DL			
Cost$_{DL}$	$O(\mu n^2 \log n)$	$O(\mu^2 n^2 \log n)$	$O(\mu^3 n^2 \log n)$

Figure 2.3. Noisy edges to download in our three algorithms.

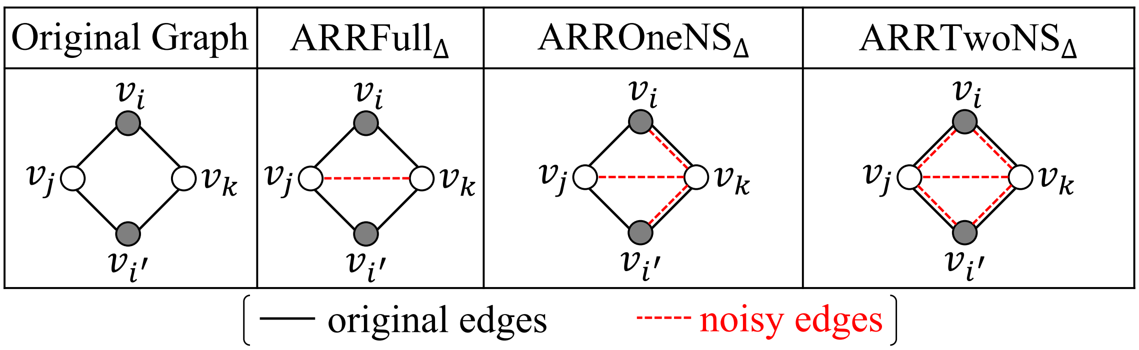


Figure 2.4. 4-cycle trick. ARRFull$_\triangle$ counts two (incorrect) noisy triangles when one noisy edge appears. ARROneNS$_\triangle$ and ARRTwoNS$_\triangle$ avoid this by increasing independent noise.

ments, we set the parameter μ in the ARR so that μ in ARRFull$_\triangle$ is equal to μ^2 in ARROneNS$_\triangle$ and also equal to μ^3 in ARRTwoNS$_\triangle$; e.g., $\mu = 10^{-6}$, 10^{-3}, and 10^{-2} in ARRFull$_\triangle$, ARROneNS$_\triangle$, and ARRTwoNS$_\triangle$, respectively. Then the download cost is the same between the three algorithms.

Figure 2.4 shows our 4-cycle trick. ARRFull$_\triangle$ counts two (incorrect) noisy triangles when a noisy edge (v_j, v_k) appears. In contrast, ARROneNS$_\triangle$ (resp. ARRTwoNS$_\triangle$) counts both the two noisy triangles only when three (resp. five) independent noisy edges appear, as shown in Figure 2.4. Thus, this bad event happens with a much smaller probability. For example, ARRFull$_\triangle$ ($\mu = 10^{-6}$), ARROneNS$_\triangle$ ($\mu = 10^{-3}$), and ARRTwoNS$_\triangle$ ($\mu = 10^{-2}$) count both the two noisy triangles with probability 10^{-6}, 10^{-9}, and 10^{-10}, respectively. The covariance $\mathrm{Cov}(E_{ijk}, E_{i'jk})$ of ARROneNS$_\triangle$ and ARRTwoNS$_\triangle$ is also much smaller than that of ARRFull$_\triangle$.

In our experiments, we show that ARROneNS$_\triangle$ and ARRTwoNS$_\triangle$ significantly outperforms $\mathsf{ARRFull}_\triangle$ for a large-scale graph or dense graph, in both of which the number of 4-cycles in G is large.

$\mathsf{ARROneNS}_\triangle$ vs. $\mathsf{ARRTwoNS}_\triangle$. One might expect that $\mathsf{ARRTwoNS}_\triangle$ outperforms $\mathsf{ARROneNS}_\triangle$ because $\mathsf{ARRTwoNS}_\triangle$ addresses the 4-cycle issue more aggressively; i.e., the number of independent noisy edges in a 4-cycle is larger in $\mathsf{ARRTwoNS}_\triangle$, as shown in Figure 2.4. However, $\mathsf{ARROneNS}_\triangle$ can reduce the global sensitivity of the Laplacian noise at the second round more effectively than $\mathsf{ARRTwoNS}_\triangle$, as explained in Section 2.5. Consequently, $\mathsf{ARROneNS}_\triangle$, which is the most tricky algorithm, achieves the smallest estimation error in our experiments. See Sections 2.5 and 2.6 for details of the global sensitivity and experiments, respectively.

Three Algorithms. Below we explain the details of our three algorithms. For ease of explanation, we assume that the maximum degree d_{max} is public in Section 2.4.2[2]. Note, however, that our double clipping (which is proposed to significantly reduce the global sensitivity) in Section 2.5 does *not* assume that d_{max} is public. Consequently, our entire algorithms do *not* require the assumption that d_{max} is public.

Recall that the server calculates a message M_i for v_i as:

$$M_i = \{(v_j, v_k) \in E' | j < k < i\} \tag{2.7}$$

$$M_i = \{(v_j, v_k) \in E' | (v_i, v_k) \in E', j < k < i\} \tag{2.8}$$

$$M_i = \{(v_j, v_k) \in E' | (v_i, v_j) \in E', (v_i, v_k) \in E', j < k < i\} \tag{2.9}$$

in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, $\mathsf{ARRTwoNS}_\triangle$, respectively.

Algorithm 4 shows our three algorithms. These algorithms are processed differently in lines 2 and 9; "F", "O", "T" are shorthands for $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively. The privacy budgets for the first and second rounds are $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$, respectively.

[2]For example, d_{max} is public in Facebook: $d_{max} = 5000$ [210]. If the server does not have prior knowledge about d_{max}, she can privately estimate d_{max} and use graph projection to guarantee that each user's degree never exceeds the private estimate of d_{max} [102]. In any case, the assumption in Section 2.4.2 does not undermine our algorithms, because our entire algorithms with double clipping in Section 2.5 does *not* assume that d_{max} is public.

Data: Graph $G \in \mathscr{G}$ represented as neighbor lists $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \{0,1\}^n$, privacy budgets $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$, $d_{max} \in \mathbb{Z}_{\geq 0}$, $\mu \in [0, \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1}]$.

Result: Private estimate $\hat{f}_\triangle(G)$ of $f_\triangle(G)$.

1 [s] $\rho \leftarrow e^{-\varepsilon_1}$;

2 [v_i, s] $\mu^* \leftarrow \mu$, μ^2, and μ^3 in F, O, and T, respectively;

`/* First round. */`

3 **for** $i = 1$ **to** n **do**

4   [v_i] $\mathbf{r}_i \leftarrow (ARR_{\varepsilon_1,\mu}(a_{i,1}), \ldots, ARR_{\varepsilon_1,\mu}(a_{i,i-1}))$;

5   [v_i] Upload $\mathbf{r}_i = (r_{i,1}, \ldots, r_{i,i-1})$ to the server;

6 **end**

7 [s] $E' = \{(v_j, v_k) : r_{k,j} = 1, j < k\}$;

`/* Second round. */`

8 **for** $i = 1$ **to** n **do**

9   [s] Compute M_i by (2.7), (2.8), and (2.9) in F, O, and T, respectively;

10   [v_i] Download M_i from the server;

11   [v_i] $t_i \leftarrow |\{(v_i, v_j, v_k) : a_{i,j} = a_{i,k} = 1, (v_j, v_k) \in M_i, j < k < i\}|$;

12   [v_i] $s_i \leftarrow |\{(v_i, v_j, v_k) : a_{i,j} = a_{i,k} = 1, j < k < i\}|$;

13   [v_i] $w_i \leftarrow t_i - \mu^* \rho s_i$;

14   [v_i] $\hat{w}_i \leftarrow w_i + \mathrm{Lap}(\frac{d_{max}}{\varepsilon_2})$;

15   [v_i] Upload $\hat{w}_i$ to the server;

16 **end**

17 [s] $\hat{f}_\triangle(G) \leftarrow \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^n \hat{w}_i$;

18 **return** $\hat{f}_\triangle(G)$

Algorithm 4: Our three algorithms. "F", "O", "T" are shorthands for ARRFull$_\triangle$, ARROneNS$_\triangle$, and ARRTwoNS$_\triangle$, respectively. [v_i] and [s] represent that the process is run by v_i and the server, respectively.

The first round appears in lines 3-7 of Algorithm 4. In this round, each user applies $ARR_{\varepsilon_1,\mu}$ defined by (2.5) and (2.6) to bits $a_{i,1}, \ldots, a_{i,i-1}$ for smaller user IDs in her neighbor list $\mathbf{a}_i$, i.e., lower triangular part of $\mathbf{A}$. Let $\mathbf{r}_i = (r_{i,1}, \ldots, r_{i,i-1}) \in \{0,1\}^{i-1}$ be the obfuscated bits of v_i. User v_i uploads $\mathbf{r}_i$ to the server. Then the server combines the noisy edges together, forming $E' = \{(v_j, v_k) : r_{k,j} = 1, j < k\}$.

The second round appears in lines 8-17 of Algorithm 4. In this round, the server computes a message M_i by (2.7), (2.8), or (2.9), and user v_i downloads it. Then user v_i calculates the number $t_i \in \mathbb{Z}_{\geq 0}$ of noisy triangles (v_i, v_j, v_k) such that only one edge (v_j, v_k) is noisy, as shown in Figure 2.2. User v_i also calculate a corrective term $s_i \in \mathbb{Z}_{\geq 0}$. The corrective term s_i is the number of possible triangles involving v_i and is computed to obtain an unbiased estimate of $f_\triangle(G)$. User v_i calculates $w_i = t_i - \mu^* \rho s_i$, where $\rho = e^{-\varepsilon_1}$ and $\mu^* = \mu$, μ^2, and μ^3 in "F", "O", and "T", respectively. Then v_i adds the Laplacian noise Lap($\frac{d_{max}}{\varepsilon_2}$) to w_i to provide ε_2-edge LDP and sends the noisy value $\hat{w}_i$ ($= w_i + \mathrm{Lap}(\frac{d_{max}}{\varepsilon_2})$) to the server. Note that adding one edge increases both t_i and s_i by at most d_{max}. Thus, the global sensitivity of w_i is at most d_{max}. Finally, the server calculates an estimate of $f_\triangle(G)$ as: $\hat{f}_\triangle(G) = \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^{n} \hat{w}_i$. As we prove later, $\hat{f}_\triangle(G)$ is an unbiased estimate of $f_\triangle(G)$.

2.4.3 Theoretical Analysis

We now introduce the theoretical guarantees on the privacy, communication, and utility of our algorithms.

Privacy. We first show the privacy guarantees:

Theorem 8. *For $i \in [n]$, let $\mathscr{R}_i^1, \mathscr{R}_i^2(M_i)$ be the randomizers used by user v_i in rounds 1 and 2 of Algorithm 4. Let $\mathscr{R}_i(\mathbf{a}_i) = (\mathscr{R}_i^1(\mathbf{a}_i), \mathscr{R}_i^2(M_i)(\mathbf{a}_i))$ be the composition of the two randomizers. Then, $\mathscr{R}_i$ satisfies $(\varepsilon_1 + \varepsilon_2)$-edge LDP and $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ satisfies $(\varepsilon_1 + \varepsilon_2)$-relationship DP.*

Note that the doubling issue in Section 2.3.2 does not occur, because we use only the lower triangular part of $\mathbf{A}$. By the immunity to post-processing, the estimate $\hat{f}_\triangle(G)$ also satisfies $(\varepsilon_1 + \varepsilon_2)$-edge LDP and $(\varepsilon_1 + \varepsilon_2)$-relationship DP.

Communication. Recall that we evaluate the algorithms based on their download cost (2.3) and upload cost (2.4).

Download Cost: The download cost is the number of bits required to download M_i. M_i can be represented as a list of edges between others, and each edge can be identified with two indices (user IDs), i.e., $2 \log n$ bits. There are $\frac{(n-1)(n-2)}{2} \approx \frac{n^2}{2}$ edges between others. $ARR_{\varepsilon_1,\mu}$ outputs 1 with probability at most μ. In addition, each noisy triangle must have 1, 2, and 3 noisy edges in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively, as shown in Figure 2.3.

Thus, the download cost in Algorithm 4 can be written as:

$$Cost_{DL} \leq \mu^* n^2 \log n, \tag{2.10}$$

where $\mu^* = \mu$, μ^2, and μ^3 in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively. In (2.10), we upper-bounded Cost_{DL} by using the fact that $ARR_{\varepsilon_1,\mu}$ outputs 1 with probability at most μ. However, when $d_{max} \ll n$, $ARR_{\varepsilon_1,\mu}$ outputs 1 with probability $\mu e^{-\varepsilon_1}$ in most cases. In that case, we can roughly approximate Cost_{DL} by replacing μ with $\mu e^{-\varepsilon_1}$ in (2.10).

Upload Cost: The upload cost comes from the number of bits required to upload $\mathscr{R}_i^1(\mathbf{a}_i)$ and $\mathscr{R}_i^2(M_i)(\mathbf{a}_i)$. Uploading $\mathscr{R}_i^1(\mathbf{a}_i)$ involves uploading $\mathbf{r}_i$ (line 5), which is a list of up to n noisy neighbors. By sending just the indices (user IDs) of the 1s in $\mathbf{r}_i$, each user sends $\|\mathbf{r}_i\|_1 \log n$ bits, where $\|\mathbf{r}_i\|_1$ is the number of 1s in $\mathbf{r}_i$. When we use $ARR_{\varepsilon_1,\mu}$, we have $\mathbb{E}[\|\mathbf{r}_i\|_1] \leq \mu n$. Uploading $\mathscr{R}_i^2(M_i)$ involves uploading a single real number $\hat{w}_i$ (line 15), which is negligibly small (e.g., 64 bits when we use a double-precision floating-point).

Thus, the upload cost in Algorithm 4 can be written as:

$$Cost_{UL} \leq \mu n \log n. \tag{2.11}$$

Clearly, Cost_{UL} is much smaller than Cost_{DL} for large n.

Utility. Analyzing the expected l_2 loss $l_2^2(f_\triangle(G), \hat{f}_\triangle(G))$ of the algorithms involves first proving that the estimator $\hat{f}_\triangle$ is unbiased and then analyzing the variance $\mathbb{V}[\hat{f}_\triangle(G)]$ to obtain an upper-bound on $l_2^2(f_\triangle(G), \hat{f}_\triangle(G))$. This is given in the following:

Theorem 9. *Let $G \in \mathscr{G}$, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$, and $\mu \in [0, \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1}]$. Let $\hat{f}_\triangle^F(G), \hat{f}_\triangle^O(G)$, and $\hat{f}_\triangle^T(G)$ be the estimates output respectively by $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$ in Algorithm 4.*

Then, $\mathbb{E}[\hat{f}^F_\triangle(G)] = \mathbb{E}[\hat{f}^O_\triangle(G)] = \mathbb{E}[\hat{f}^T_\triangle(G)] = f_\triangle(G)$ *(i.e., estimates are unbiased) and*

$$
\begin{aligned}
l_2^2(f_\triangle(G), \hat{f}^F_\triangle(G)) &\leq \tfrac{2C_4(G)+S_2(G)}{\mu(1-e^{\varepsilon_1})^2} + \tfrac{2nd_{max}^2}{\mu^2(1-e^{\varepsilon_1})^2\varepsilon_2^2} \\
l_2^2(f_\triangle(G), \hat{f}^O_\triangle(G)) &\leq \tfrac{\mu(2C_4(G)+6S_3(G))+S_2(G)}{\mu^2(1-e^{\varepsilon_1})^2} + \tfrac{2nd_{max}^2}{\mu^4(1-e^{\varepsilon_1})^2\varepsilon_2^2} \\
l_2^2(f_\triangle(G), \hat{f}^T_\triangle(G)) &\leq \tfrac{\mu^2(2C_4(G)+6S_3(G))+S_2(G)}{\mu^3(1-e^{\varepsilon_1})^2} + \tfrac{2nd_{max}^2}{\mu^6(1-e^{\varepsilon_1})^2\varepsilon_2^2},
\end{aligned}
$$

where $C_4(G)$ *is the number of* 4*-cycles in* G *and* $S_k(G)$ *is the number of* k*-stars in* G*.*

For each of the three upper-bounds in Theorem 9, the first and second terms are the estimation errors caused by empirical estimation and the Laplacian noise, respectively. We also note that $C_4(G) = S_3(G) = O(nd_{max}^3)$ and $S_2(G) = O(nd_{max}^2)$. Thus, for small μ, the l_2 loss of empirical estimation can be expressed as $O(nd_{max}^3)$, $O(nd_{max}^2)$, and $O(nd_{max}^2)$ in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, $\mathsf{ARRTwoNS}_\triangle$, respectively (as the factors of $C_4(G)$ and $S_3(G)$ diminish for small μ).

This highlights our 4-cycle trick. The large l_2 loss of $\mathsf{ARRFull}_\triangle$ is caused by the number $C_4(G) = O(nd_{max}^3)$ of 4-cycles. $\mathsf{ARROneNS}_\triangle$ and $\mathsf{ARRTwoNS}_\triangle$ addresses this issue by increasing independent noise, as shown in Figure 2.4.

2.5 Double Clipping

In Section 2.4, we showed that the estimation error caused by empirical estimation (i.e., the first term in Theorem 9) is significantly reduced by the 4-cycle trick. However, the estimation error is still very large in our algorithms presented in Section 2.4, as shown in our experiments. This is because the estimation error by the Laplacian noise (i.e., the second term in Theorem 9) is very large, especially for small ε_2 or μ. This error term is tight and unavoidable as long as we use d_{max} as a global sensitivity, which suggests that we need a better global sensitivity analysis. To significantly reduce the global sensitivity, we propose a novel *double clipping* technique.

We describe the overview and details of our double clipping in Sections 2.5.1 and 2.5.2, respectively. Then we perform theoretical analysis in Section 2.5.3.

2.5.1 Overview

Motivation. Figure 2.5 shows noisy triangles involving edge (v_i, v_j) counted by user v_i in our three algorithms. Our algorithms in Section 2.4 use the fact that the number of such noisy triangles (hence the global sensitivity) is upper-bounded by the maximum degree d_{max} because adding one edge increases the triangle count by at most d_{max}. Unfortunately, this upper-bound is too large, as shown in our experiments.

In this paper, we significantly reduce this upper-bound by using the parameter μ in the ARR and user v_i's degree $d_i \in \mathbb{Z}_{\geq 0}$ for users with smaller IDs. For example, the number of noisy triangles involving (v_i, v_j) in $\mathsf{ARRFull}_{\triangle}$is expected to be around μd_i because one noisy edge is included in each noisy triangle (as shown in Figure 2.5) and all noisy edges are independent. μd_i is very small, especially when we set $\mu \ll 1$ to reduce the communication cost.

However, we cannot directly use μd_i as an upper-bound of the global sensitivity in $\mathsf{ARRFull}_{\triangle}$for two reasons. First, μd_i leaks the exact value of user v_i's degree d_i and violates edge LDP. Second, the number of noisy triangles involving (v_i, v_j) exceeds μd_i with high probability (about 0.5). Thus, the noisy triangle count cannot be upper-bounded by μd_i.

To address these two issues, we propose a double clipping technique, which is explained below.

Algorithm Overview. Figure 2.6 shows the overview of our double clipping, which consists of an *edge clipping* and *noisy triangle clipping*. The edge clipping addresses the first issue (i.e., leakage of d_i) as follows. It privately computes a noisy version of d_i (denoted by $\tilde{d_i}$) with edge LDP. Then it removes some neighbors from a neighbor list $\mathbf{a}_i$ so that the degree of v_i never exceeds the noisy degree $\tilde{d_i}$. This removal process is also known as graph projection [58, 64, 122, 167]. Edge clipping is used in [102] to obtain a noisy version of d_{max}.

The main novelty in our double clipping lies at the *noisy triangle clipping* to address the second issue (i.e., excess of the noisy triangle count). This issue appears when we attempt to

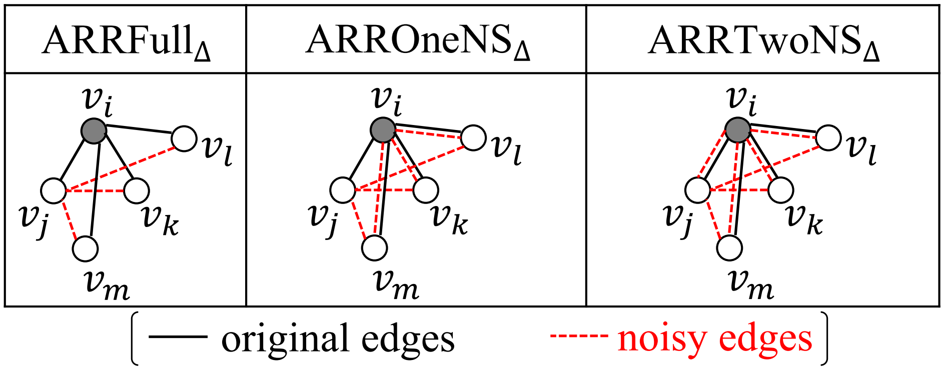


Figure 2.5. Noisy triangles involving edge (v_i, v_j) counted by user v_i ($j < k, l, m < i$).

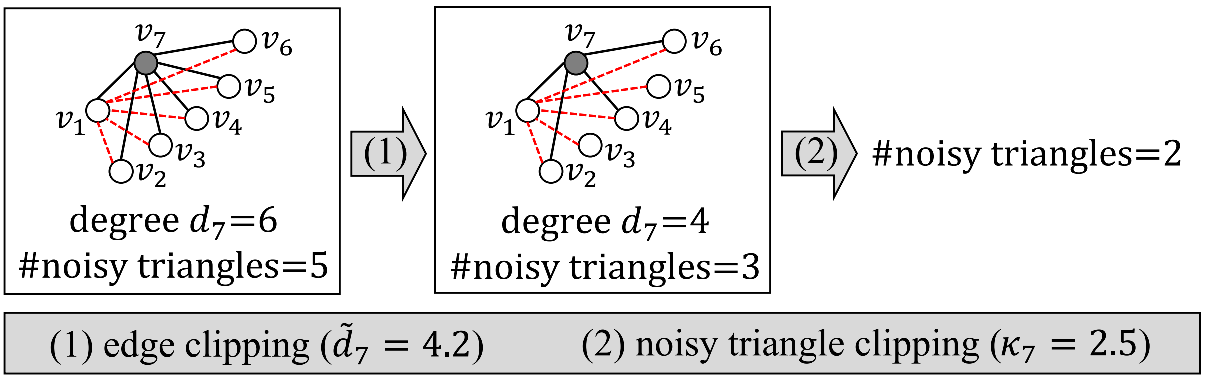


Figure 2.6. Overview of double clipping applied to edge (v_1, v_7).

reduce the global sensitivity by using a very small sampling probability for each edge. Therefore, the noisy triangle clipping has not been studied in the existing works on private triangle counting [64, 102, 120, 122, 186, 218, 219, 223], because they do not apply a sampling technique.

Our noisy triangle clipping reduces the noisy triangle count so that it never exceeds a user-dependent clipping threshold $\kappa_i \in \mathbb{R}_{\geq 0}$. Then a crucial issue is how to set an appropriate threshold κ_i. We theoretically analyze the probability that the noisy triangle count exceeds κ_i (referred to as the *triangle excess probability*) as a function of the ARR parameter μ and the noisy degree $\tilde{d}_i$. Then we set κ_i so that the triangle excess probability is very small ($= 10^{-6}$ in our experiments).

We use the clipping threshold κ_i as a global sensitivity. Note that κ_i provides edge LDP because $\tilde{d}_i$ provides edge LDP, i.e., immunity to post-processing [73]. κ_i is also very small when

Data: Neighbor list $\mathbf{a}_i \in \{0,1\}^n$, privacy budget $\varepsilon_0 \in \mathbb{R}_{\geq 0}$ $\mu \in [0, \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1}]$, $\alpha \in \mathbb{R}_{\geq 0}$, $\beta \in \mathbb{R}_{\geq 0}$.
Result: $\hat{w}_i$.
1 $\mu^* \leftarrow \mu$, μ^2, and μ^3 in F, O, and T, respectively;
/* Edge clipping. */
2 $\tilde{d}_i = \max\{d_i + \mathrm{Lap}(\frac{1}{\varepsilon_0}) + \alpha, 0\}$;
/* Remove $d_i - \lfloor \tilde{d}_i \rfloor$ neighbors if $d_i > \tilde{d}_i$. */
3 $\mathbf{a}_i \leftarrow \mathtt{GraphProjection}(\mathbf{a}_i, \tilde{d}_i)$;
/* Noisy triangle clipping. */
4 **for** j such that $a_{i,j} = 1$ and $j < i$ **do**
5 | $t_{i,j} \leftarrow |\{(v_i, v_j, v_k) : a_{i,k} = 1, (v_j, v_k) \in M_i, j < k < i\}|$;
6 **end**
/* Calculate $\kappa_i \in [\mu^* \tilde{d}_i, \tilde{d}_i]$ s.t. the triangle excess probability is β or less. */
7 $\kappa_i \leftarrow \mathtt{ClippingThreshold}(\mu, \tilde{d}_i, \beta)$;
8 $t_i \leftarrow \sum_{a_{i,j}=1, j<i} \min\{t_{i,j}, \kappa_i\}$;
9 $s_i \leftarrow |\{(v_i, v_j, v_k) : a_{i,j} = a_{i,k} = 1, j < k < i\}|$;
10 $w_i \leftarrow t_i - \mu^* \rho s_i$;
11 $\hat{w}_i \leftarrow w_i + \mathrm{Lap}(\frac{\kappa_i}{\varepsilon_2})$;
12 **return** $\hat{w}_i$

Algorithm 5: Our double clipping algorithm. "F", "O", "T" are shorthands for $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively. All the processes are run by user v_i.

$\mu \ll 1$, as it is determined based on μ.

2.5.2 Algorithms

Algorithm 5 shows our double clipping algorithm. All the processes are run by user v_i at the second round. Thus, there is no interaction with the server in Algorithm 5.

Edge Clipping. The edge clipping appears in lines 2-3 of Algorithm 5. It uses a privacy budget $\varepsilon_0 \in \mathbb{R}_{\geq 0}$.

In line 2, user v_i adds the Laplacian noise $\mathrm{Lap}(\frac{1}{\varepsilon_0})$ to her degree d_i. Since adding/removing one edge changes d_i by at most 1, this process provides ε_0-edge LDP. v_i also adds some non-negative constant $\alpha \in \mathbb{R}_{\geq 0}$ to d_i. We add this value so that edge removal (in line 3) occurs with a very small probability; e.g., in our experiments, we set $\alpha = 150$, where edge removal occurs with probability 1.5×10^{-7} when $\varepsilon_0 = 0.1$. A similar technique is introduced in [186] to provide (ε, δ)-DP [73] with small δ. The difference between ours and [186] is that we perform edge clipping to always provide ε-DP; i.e., $\delta = 0$. Let $\tilde{d_i} \in \mathbb{R}_{\geq 0}$ be the noisy degree of v_i.

In line 3, user v_i calls the function `GraphProjection`, which performs graph projection as follows; if $d_i > \tilde{d_i}$, randomly remove $d_i - \lfloor \tilde{d_i} \rfloor$ neighbors from $\mathbf{a}_i$; otherwise, do nothing. Consequently, the degree of v_i never exceeds $\tilde{d_i}$.

Noisy Triangle Clipping. The noisy triangle clipping appears in lines 4-11 of Algorithm 5.

In lines 4-6, user v_i calculates the number $t_{i,j} \in \mathbb{Z}_{\geq 0}$ of noisy triangles (v_i, v_j, v_k) $(j < k < i)$ involving (v_i, v_j) (as shown in Figure 2.5). Note that the total number t_i of noisy triangles of v_i can be expressed as: $t_i = \sum_{a_{i,j}=1, j<i} t_{i,j}$. In line 7, v_i calls the function `ClippingThreshold`, which calculates a clipping threshold $\kappa_i \in [\mu^* \tilde{d_i}, \tilde{d_i}]$ ($\mu^* = \mu$, μ^2, and μ^3 in "F", "O", and "T", respectively) based on the ARR parameter μ and the noisy degree $\tilde{d_i}$ so that the triangle excess probability does not exceed some constant $\beta \in \mathbb{R}_{\geq 0}$. We explain how to calculate the triangle excess probability in Section 2.5.3. In line 8, v_i calculates the total number t_i of noisy triangles by summing up $t_{i,j}$, with the exception that v_i adds κ_i if $t_{i,j} > \kappa_i$. In other words, triangle removal occurs if $t_{i,j} > \kappa_i$. Then, the number of noisy triangles involving (v_i, v_j) never exceeds κ_i.

Lines 9-11 in Algorithm 5 are the same as lines 12-14 in Algorithm 4, except that the global sensitivity in the former (resp. latter) is κ_i (resp. d_{max}). Line 11 in Algorithm 5 provides ε_2-edge LDP because the number of triangles involving (v_i, v_j) is now upper-bounded by κ_i.

Our Entire Algorithms with Double Clipping. We can run our algorithms $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, $\mathsf{ARRTwoNS}_\triangle$ with double clipping just by replacing lines 11-14 in Algorithm 4 with lines 2-11 in Algorithm 5. That is, after calculating $\hat{w}_i$ by Algorithm 5, v_i uploads $\hat{w}_i$ to the server. Then the server calculates an estimate of $f_\triangle(G)$ as $\hat{f}_\triangle(G) = \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^n \hat{w}_i$.

We also note that the input d_{max} in Algorithm 4 is no longer necessary thanks to the edge clipping; i.e., our entire algorithms with double clipping do not assume that d_{max} is public.

2.5.3 Theoretical Analysis

We now perform a theoretical analysis on the privacy and utility of our double clipping.

Privacy. We begin with the privacy guarantees:

Theorem 10. *For $i \in [n]$, let $\mathscr{R}_i^1, \mathscr{R}_i^2(M_i)$ be the randomizers used by user v_i in rounds 1 and 2 of our algorithms with double clipping (Algorithms 4 and 5). Let $\mathscr{R}_i(\mathbf{a}_i) = (\mathscr{R}_i^1(\mathbf{a}_i), \mathscr{R}_i^2(M_i)(\mathbf{a}_i))$ be the composition of the two randomizers. Then, $\mathscr{R}_i$ satisfies $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$-edge LDP, and $(\mathscr{R}_1, \dots, \mathscr{R}_n)$ satisfies $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$-relationship DP.*

Utility. Next, we show the triangle excess probability:

Theorem 11. *In Algorithm 5, the triangle excess probability (i.e., probability that the number of noisy triangles $t_{i,j}$ involving edge (v_i, v_j) exceeds a clipping threshold κ_i) is:*

$$\Pr(t_{i,j} > \kappa_i) \leq \exp\left[-\tilde{d}_i D\left(\tfrac{\kappa_i}{\tilde{d}_i} \parallel \mu\right)\right] \tag{2.12}$$

$$\Pr(t_{i,j} > \kappa_i) \leq \exp\left[-\tilde{d}_i D\left(\tfrac{\kappa_i}{\tilde{d}_i} \parallel \mu^2\right)\right] \tag{2.13}$$

$$\Pr(t_{i,j} > \kappa_i) \leq \mu \exp\left[-\tilde{d}_i D\left(\tfrac{\max\{\kappa_i, \mu^2 \tilde{d}_i\}}{\tilde{d}_i} \parallel \mu^2\right)\right] \tag{2.14}$$

in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively, where $D(p_1 \parallel p_2)$ is the Kullback-Leibler divergence between two Bernoulli distributions; i.e.,

$$D(p_1 \parallel p_2) = p_1 \log \tfrac{p_1}{p_2} + (1 - p_1) \log \tfrac{1-p_1}{1-p_2}.$$

In all of (2.12), (2.13), and (2.14), we use the Chernoff bound, which is known to be reasonably tight [12].

Setting κ_i. The function `ClippingThreshold` in Algorithm 5 sets a clipping threshold κ_i of user v_i based on Theorem 11. Specifically, we set $\kappa_i = \lambda_i \mu^* \tilde{d}_i$, where $\lambda_i \in \mathbb{N}$, and calculate λ_i as follows. We initially set $\lambda_i = 1$ and keep increasing λ_i by 1 until the upper-bound (i.e., right-hand side of (2.12), (2.13), or (2.14)) is smaller than or equal to the triangle excess probability β. In our experiments, we set $\beta = 10^{-6}$.

Large κ_i of ARRTwoNS$_\triangle$. By (2.12) and (2.13), the upper-bound on the triangle excess probability is the same between ARRFull$_\triangle$ and ARROneNS$_\triangle$. In contrast, ARRTwoNS$_\triangle$ has a larger upper-bound. For example, when $\kappa_i = 15\mu^*\tilde{d}_i$, $\mu^* = 10^{-3}$, and $\tilde{d}_i = 1000$, the right-hand sides of (2.12), (2.13), and (2.14) are 2.5×10^{-12}, 2.5×10^{-12}, and 3.3×10^{-2}, respectively. Consequently, ARRTwoNS$_\triangle$ has a larger global sensitivity κ_i for the same value of β.

We can explain a large global sensitivity κ_i of ARRTwoNS$_\triangle$ as follows. The number $t_{i,j}$ of noisy triangles involving (v_i, v_j) in ARRFull$_\triangle$ is expected to be around μd_i because one noisy edge is in each noisy triangle (as in Figure 2.5) and all noisy edges are independent. For the same reason, $t_{i,j}$ in ARROneNS$_\triangle$ is expected to be around $\mu^2 d_i$. However, $t_{i,j}$ in ARRTwoNS$_\triangle$ is *not* expected to be around $\mu^3 d_i$, because all the noisy triangles have noisy edge (v_i, v_j) in common (as in Figure 2.5). Then, the expectation of $t_{i,j}$ largely depends on the presence/absence of the noisy edge (v_i, v_j); i.e., if noisy edge (v_i, v_j) exists, it is $\mu^2 d_i$; otherwise, 0. Thus, κ_i cannot be effectively reduced by double clipping.

Summary. The performance guarantees of our three algorithms with double clipping can be summarized in Table 2.1.

The first and second terms of the expected l_2 loss are the l_2 loss of empirical estimation and that of the Laplacian noise, respectively. For small μ, the l_2 loss of empirical estimation can be expressed as $O(nd_{max}^3)$, $O(nd_{max}^2)$, and $O(nd_{max}^2)$ in ARRFull$_\triangle$, ARROneNS$_\triangle$, ARRTwoNS$_\triangle$, respectively, as explained in Section 2.4.3. The l_2 loss of the Laplacian noise is $O(\sum_{i=1}^n \kappa_i^2)$, which is much smaller than $O(nd_{max}^2)$. Thus, our ARROneNS$_\triangle$ that effectively reduces κ_i provides the smallest error, as shown in our experiments.

We also note that both the space and the time complexity to compute and send M_i in our algorithms are $O(\mu^* n^2)$ (as $|E'| = O(\mu^* n^2)$), which is much smaller than [102] ($= O(n^2)$).

2.6 Experiments

To evaluate each component of our algorithms in Sections 2.4 and 2.5 as well as our entire algorithms (i.e., $\mathsf{ARRFull}_{\triangle}$, $\mathsf{ARROneNS}_{\triangle}$, $\mathsf{ARRTwoNS}_{\triangle}$with double clipping), we pose the following three research questions:

RQ1. How do our three triangle counting algorithms (i.e., $\mathsf{ARRFull}_{\triangle}$, $\mathsf{ARROneNS}_{\triangle}$, $\mathsf{ARRTwoNS}_{\triangle}$) in Section 2.4 compare with each other in terms of accuracy?

RQ2. How much does our double clipping technique in Section 2.5 decrease the estimation error?

RQ3. How much do our entire algorithms reduce the communication cost, compared to the existing algorithm [102], while keeping high utility (e.g., relative error $\ll 1$)?

In Appendix B.2, we also compare our entire algorithms with one-round algorithms.

2.6.1 Experimental Set-up

In our experiments, we used two real graph datasets:

Gplus. The Google+ dataset [142] (denoted by Gplus) was collected from users who had shared circles. From the dataset, we constructed a social graph $G = (V,E)$ with 107614 nodes (users) and 12238285 edges, where edge $(v_i, v_j) \in E$ represents that v_i follows or is followed by v_j. The average (resp. maximum) degree in G is 113.7 (resp. 20127).

IMDB. The IMDB (Internet Movie Database) [1] (denoted by IMDB) includes a bipartite graph between 896308 actors and 428440 movies. From this, we constructed a graph $G = (V,E)$ with 896308 nodes (actors) and 57064358 edges, where edge $(v_i, v_j) \in E$ represents that v_i and v_j have played in the same movie. The average (resp. maximum) degree in G is 63.7 (resp. 15451). Thus, IMDB is more sparse than Gplus.

In Appendix B.4, we also evaluate our algorithms using a synthetic graph based on the Barabási-Albert model [16], which has a power-law degree distribution.

We evaluated our algorithms while changing μ^*, where $\mu^* = \mu$, μ^2, and μ^3 in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively. Cost_{DL} is the same between the three algorithms. We typically set the total privacy budget ε to $\varepsilon = 1$ (at most 2) because it is acceptable in many practical scenarios [131].

In our double clipping, we set $\alpha = 150$ and $\beta = 10^{-6}$ so that both edge removal and triangle removal occur with a very small probability ($\leq 10^{-6}$ when $\varepsilon_0 = 0.1$). Then for each algorithm, we evaluated the relative error between the true triangle count $f_\triangle(G)$ and its estimate $\hat{f}_\triangle(G)$. Since the estimate $\hat{f}_\triangle(G)$ varies depending on the randomness of LDP mechanisms, we ran each algorithm $\tau \in \mathbb{N}$ times ($\tau = 20$ and 10 for Gplus and IMDB, respectively) and averaged the relative error over the τ cases.

2.6.2 Experimental Results

Performance Comparison. First, we evaluated our algorithms with the Laplacian noise. Specifically, we evaluated all possible combinations of our three algorithms with and without our double clipping (six combinations in total) and compared them with the existing two-rounds algorithm in [102]. For algorithms with double clipping, we divided the total privacy budget ε as: $\varepsilon_0 = \frac{\varepsilon}{10}$ and $\varepsilon_1 = \varepsilon_2 = \frac{9\varepsilon}{20}$. Here, we set a very small budget ($\varepsilon_0 = \frac{\varepsilon}{10}$) for edge clipping because the degree has a small sensitivity (sensitivity$= 1$). For algorithms without double clipping, we divided ε as $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$ and used the maximum degree d_{max} as the global sensitivity.

Figures 2.7 and 2.8 show the results. Figure 2.7 highlights the relative error of our three algorithms with double clipping when $\varepsilon = 1$ or 2 and $\mu^* = 10^{-3}$. "DC" (resp. "d_{max}") represents algorithms with (resp. without) double clipping. $\mathsf{RRFull}_\triangle(d_{max})$ (marked with purple square) in Figure 2.8 (c) and (d) represents the two-rounds algorithm in [102]. Note that this is a special case of our $\mathsf{ARRFull}_\triangle$ without sampling ($\mu = \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1} = 0.62$). Figure 2.8 (c) and (d) also show the download cost Cost_{DL} calculated by (2.10). Note that when $\mu^* \geq 0.1$ (marked with squares), Cost_{DL} can be 6Gbits and 400Gbits in Gplus and IMDB, respectively, by downloading only 0/1 for each pair of users (v_j, v_k); $\mathrm{Cost}_{DL} = \frac{(n-1)(n-2)}{2}$ in this case.

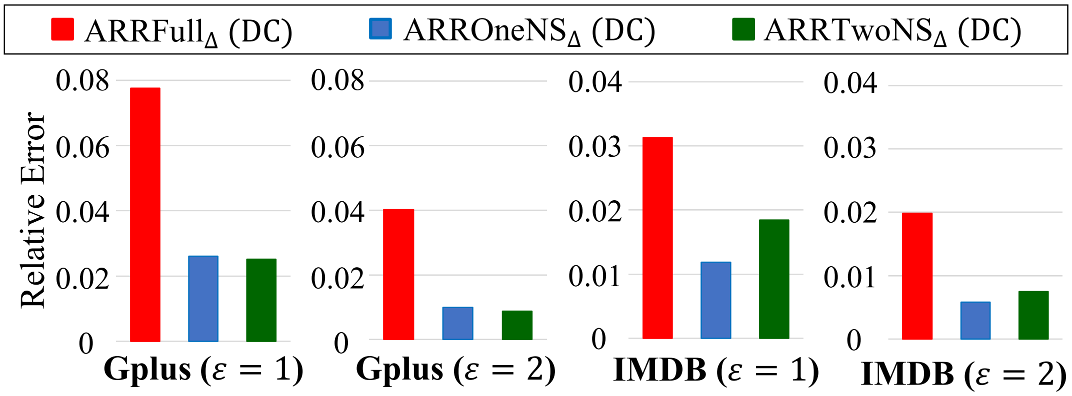


Figure 2.7. Relative error of our three algorithms with double clipping ("DC") when $\varepsilon = 1$ or 2 and $\mu^* = 10^{-3}$ ($n = 107614$ in Gplus, $n = 896308$ in IMDB).

Figures 2.7 and 2.8 show that our ARROneNS$_\triangle$ (DC) provides the best (or almost the best) performance in all cases. This is because ARROneNS$_\triangle$ (DC) introduces the 4-cycle trick and effectively reduces the global sensitivity of the Laplacian noise by double clipping. Later, we will investigate the effectiveness of the 4-cycle trick in detail by not adding the Laplacian noise. We will also investigate the impact of the Laplacian noise while changing n.

Figure 2.8 also shows that the relative error is almost the same between our three algorithms without double clipping ("d_{max}") and that it is too large. This is because Lap($\frac{d_{max}}{\varepsilon_2}$) is too large and dominant. The relative error is significantly reduced by introducing our double clipping in all cases. For example, when $\mu^* = 10^{-3}$, our double clipping reduces the relative error of ARROneNS$_\triangle$ by two or three orders of magnitude. The improvement is larger for smaller μ^*.

In Appendix B.5, we also evaluate the effect of edge clipping and noisy triangle clipping independently and show that each component significantly reduces the relative error.

Communication Cost. From Figure 2.8 (c) and (d), we can explain how much our algorithms can reduce the download cost while keeping high utility, e.g., relative error $\ll 1$.

For example, when we use the algorithm in [102], the download cost is $\text{Cost}_{DL} = 400$ Gbits in IMDB. Thus, when the download speed is 20 Mbps (recommended speed in YouTube [220]), every user v_i needs 6 hours to download the message M_i, which is far from practical. In contrast, our $\mathsf{ARROneNS}_{\triangle}$ (DC) can reduce it to 160 Mbits (8 seconds when 20 Mbps download rate) or less while keeping relative error $= 0.21$, which is practical and a dramatic improvement over [102].

We also note that since $d_{max} \ll n$ in IMDB, Cost_{DL} of our $\mathsf{ARROneNS}_{\triangle}$ (DC) can also be roughly approximated by 60 Mbits (3 seconds) by replacing μ with $\mu e^{-\varepsilon_1}$ in (2.10).

4-Cycle Trick. We also investigated the effectiveness of our 4-cycle trick in $\mathsf{ARROneNS}_{\triangle}$ and $\mathsf{ARRTwoNS}_{\triangle}$ in detail. To this end, we evaluated our three algorithms when we did *not* add the Laplacian noise at the second round. Note that they do not provide edge LDP, as $\varepsilon_2 = \infty$. The purpose here is to purely investigate the effectiveness of the 4-cycle trick related to our first research question RQ1.

Figure 2.9 shows the results, where ε_1 and μ^* are changed to various values. Figure 2.9 shows that $\mathsf{ARROneNS}_{\triangle}$ and $\mathsf{ARRTwoNS}_{\triangle}$ significantly outperform $\mathsf{ARRFull}_{\triangle}$ when μ^* is small. This is because in both $\mathsf{ARROneNS}_{\triangle}$ and $\mathsf{ARRTwoNS}_{\triangle}$, the factors of C_4 (#4-cycles) and S_3 (#3-stars) in the expected l_2 loss diminish for small μ, as explained in Section 2.4.3. In other words, $\mathsf{ARROneNS}_{\triangle}$ and $\mathsf{ARRTwoNS}_{\triangle}$ effectively address the 4-cycle issue. Figure 2.9 also shows that $\mathsf{ARRTwoNS}_{\triangle}$ slightly outperforms $\mathsf{ARROneNS}_{\triangle}$ when μ^* is small. This is because the factors of C_4 and S_3 diminish more rapidly; i.e., $\mathsf{ARRTwoNS}_{\triangle}$ addresses the 4-cycle issue more aggressively.

However, when we add the Laplacian noise, $\mathsf{ARRTwoNS}_{\triangle}$ (DC) is outperformed by $\mathsf{ARROneNS}_{\triangle}$ (DC), as shown in Figure 2.8. This is because $\mathsf{ARRTwoNS}_{\triangle}$ cannot effectively reduce the global sensitivity by double clipping. In Figure 2.8, the difference between $\mathsf{ARROneNS}_{\triangle}$ (DC) and $\mathsf{ARRFull}_{\triangle}$ (DC) is also small for very small ε or μ^* (e.g., $\varepsilon = 0.1$, $\mu^* = 10^{-6}$) because the Laplacian noise is dominant in this case.

Changing n. We finally evaluated our three algorithms with double clipping while changing the number n of users. In both Gplus and IMDB, we randomly selected n users from all users and extracted a graph with n users. Then we evaluated the relative error while changing n to various values.

Figure 2.10 shows the results, where $\varepsilon = 1$ ($\varepsilon_0 = 0.1$, $\varepsilon_1 = \varepsilon_2 = 0.45$) and $\mu^* = 10^{-3}$. In all three algorithms, the relative error decreases with increase in n. This is because the expected l_2 loss can be expressed as $O(nd_{max}^3)$ or $O(nd_{max}^2)$ in these algorithms as shown in Section 2.5.3 and the square of the true triangle count can be expressed as $\Omega(n^2)$. In other words, when $d_{max} \ll n$, the relative error becomes smaller for larger n. Figure 2.10 also shows that for small n, $\mathsf{ARRTwoNS}_\triangle$ provides the worst performance and $\mathsf{ARROneNS}_\triangle$ performs almost the same as $\mathsf{ARRFull}_\triangle$. For large n, $\mathsf{ARRFull}_\triangle$ performs the worst and $\mathsf{ARROneNS}_\triangle$ performs the best.

To investigate the reason for this, we decomposed the estimation error into two components – the first error caused by empirical estimation and the second error caused by the Laplacian noise. Specifically, for each algorithm, we evaluated the first error by calculating the relative error when we did not add the Laplacian noise ($\varepsilon_1 = 0.45$). Then we evaluated the second error by subtracting the first error from the relative error when we used double clipping ($\varepsilon_0 = 0.1$, $\varepsilon_1 = \varepsilon_2 = 0.45$).

Figure 2.11 shows the results for some values of n, where "emp" represents the first error by empirical estimation and "Lap" represents the second error by the Laplacian noise. We observe that the second error rapidly decreases with increase in n. In addition, the first error of $\mathsf{ARRFull}_\triangle$ is much larger than those of $\mathsf{ARROneNS}_\triangle$ and $\mathsf{ARRTwoNS}_\triangle$ when n is large.

We also examined the number C_4 of 4-cycles as a function of n. Figure 2.12 shows the results. We observe that C_4 (which is $O(nd_{max}^3)$) is quartic in n; e.g., C_4 is increased by $2^4 \approx 10$ and $6^4 \approx 10^3$ when n is multiplied by 2 and 6, respectively. This is because we randomly selected n users from all users and d_{max} is almost proportional to n (though $d_{max} \ll n$).

Based on Figures 2.11 and 2.12, we can explain Figure 2.10 as follows. As shown in Section 2.5.3, the l_2 loss of empirical estimation can be expressed as $O(nd_{max}^3)$, $O(nd_{max}^2)$, and $O(nd_{max}^2)$ in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively. The large l_2 loss of $\mathsf{ARRFull}_\triangle$ is caused by a large value of C_4. The expected l_2 loss of the Laplacian noise is $O(\sum_{i=1}^n \kappa_i^2)$, which is much smaller than $O(nd_{max}^2)$. Thus, as n increases, the Laplacian noise becomes relatively very small, as shown in Figure 2.11. Consequently, $\mathsf{ARROneNS}_\triangle$ provides the best performance for large n because it addresses the 4-cycle issue and effectively reduces the global sensitivity. This explains the results in Figure 2.10. It is also interesting that when $n \approx 10^5$, $\mathsf{ARRFull}_\triangle$ performs the worst in Gplus and almost the same as $\mathsf{ARROneNS}_\triangle$ in IMDB (see Figure 2.10). This is because Gplus is more dense than IMDB and C_4 is much larger in Gplus when $n \approx 10^5$, as in Figure 2.12.

In other words, Figures 2.10, 2.11, and 2.12 are consistent with our theoretical results in Section 2.5.3. From these results, we conclude that $\mathsf{ARROneNS}_\triangle$ is effective especially for a large graph (e.g., $n \approx 10^6$) or dense graph (e.g., Gplus) where the number C_4 of 4-cycles is large.

Summary. In summary, our answers to our three research questions RQ1-3 are as follows. [RQ1]: Our $\mathsf{ARROneNS}_\triangle$ achieves almost the smallest estimation error in all cases and outperforms the other two, especially for a large graph or dense graph where C_4 is large. [RQ2]: Our double clipping reduces the estimation error by two or three orders of magnitude. [RQ3]: Our entire algorithm ($\mathsf{ARROneNS}_\triangle$ with double clipping) dramatically reduces the communication cost, e.g., from 6 hours to 8 seconds or less (relative error $= 0.21$) in IMDB at a 20 Mbps download rate [220].

Thus, triangle counting under edge LDP is now much more practical. In Appendix B.3, we show that the clustering coefficient can also be accurately estimated using our algorithms.

2.7 Conclusions

We proposed triangle counting algorithms under edge LDP with a small estimation error and small communication cost. We showed that our entire algorithms with the 4-cycle trick and double clipping dramatically reduce the download cost of [102], e.g., from 6 hours to 8 seconds or less.

We assumed that each user v_i honestly inputs her neighbor list $\mathbf{a}_i$, as in most previous work on LDP. However, recent studies [34, 46] show that the estimate in LDP can be skewed by data poisoning attacks. As future work, we would like to analyze the impact of data poisoning on our algorithms and develop defenses (e.g., detection) against it.

Chapter 2, in full, is a reprint of the material as it appears in Jacob Imola, Takao Murakami, and Kamalika Chaudhuri, 31st USENIX Security Symposium, 2022. "Communication-Efficient Triangle Counting under Local Differential Privacy." The dissertation author was a joint first author of this paper.

	$\mathsf{ARRFull}_{\triangle}$	$\mathsf{ARROneNS}_{\triangle}$	$\mathsf{ARRTwoNS}_{\triangle}$
Privacy	$(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-edge LDP and $(\varepsilon_0+\varepsilon_1+\varepsilon_2)$-relationship DP		
Expected l_2 loss	$O\left(\frac{nd_{max}^3}{\mu(1-e^{-\varepsilon_1})^2}+\frac{2\sum_{i=1}^n \kappa_i^2}{\mu^2(1-e^{-\varepsilon_1})^2\varepsilon_2^2}\right)$	$O\left(\frac{nd_{max}^2}{\mu^2(1-e^{-\varepsilon_1})^2}+\frac{2\sum_{i=1}^n \kappa_i^2}{\mu^4(1-e^{-\varepsilon_1})^2\varepsilon_2^2}\right)$	$O\left(\frac{nd_{max}^2}{\mu^3(1-e^{-\varepsilon_1})^2}+\frac{2\sum_{i=1}^n \kappa_i^2}{\mu^6(1-e^{-\varepsilon_1})^2\varepsilon_2^2}\right)$
Cost_{DL}	$\mu n^2 \log n$	$\mu^2 n^2 \log n$	$\mu^3 n^2 \log n$
Cost_{UL}	$\mu n \log n$	$\mu n \log n$	$\mu n \log n$

Table 2.1. Performance guarantees of our three algorithms with double clipping when the edge removal and triangle removal do not occur. The expected l_2 loss assumes that μ is small. The download (resp. upload) cost is an upper-bound in (2.10) (resp. (2.11)).

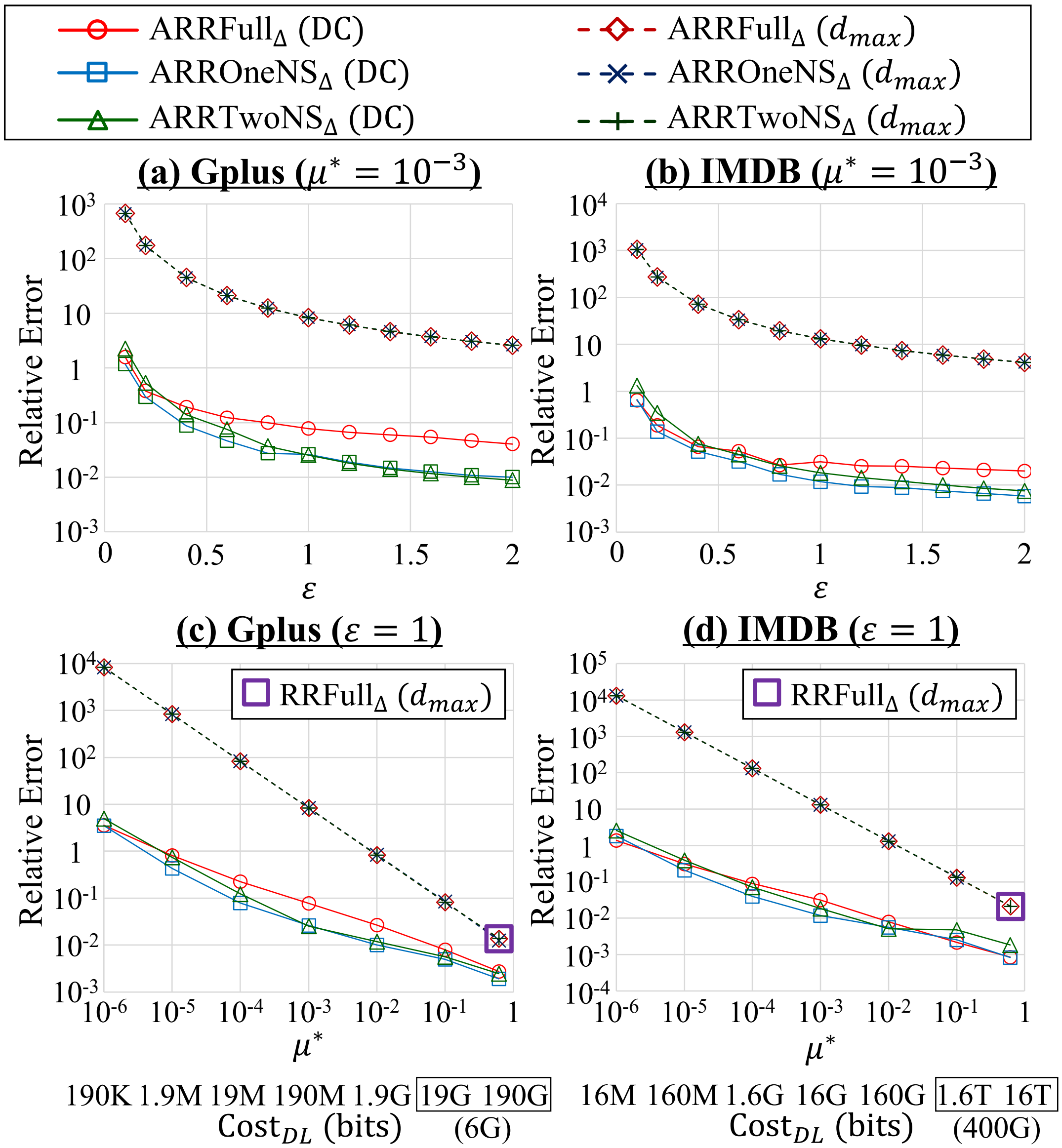


Figure 2.8. Relative error of our three algorithms with ("DC") or without ("d_{max}") double clipping ($n = 107614$ in Gplus, $n = 896308$ in IMDB). RRFull$_\triangle(d_{max})$ is the algorithm in [102]. Cost$_{DL}$ is an upper-bound in (2.10). When $\mu^* \geq 0.1$, Cost$_{DL}$ can be 6 Gbits and 400 Gbits in Gplus and IMDB, respectively, by downloading only 0/1 for each pair of users (v_j, v_k).

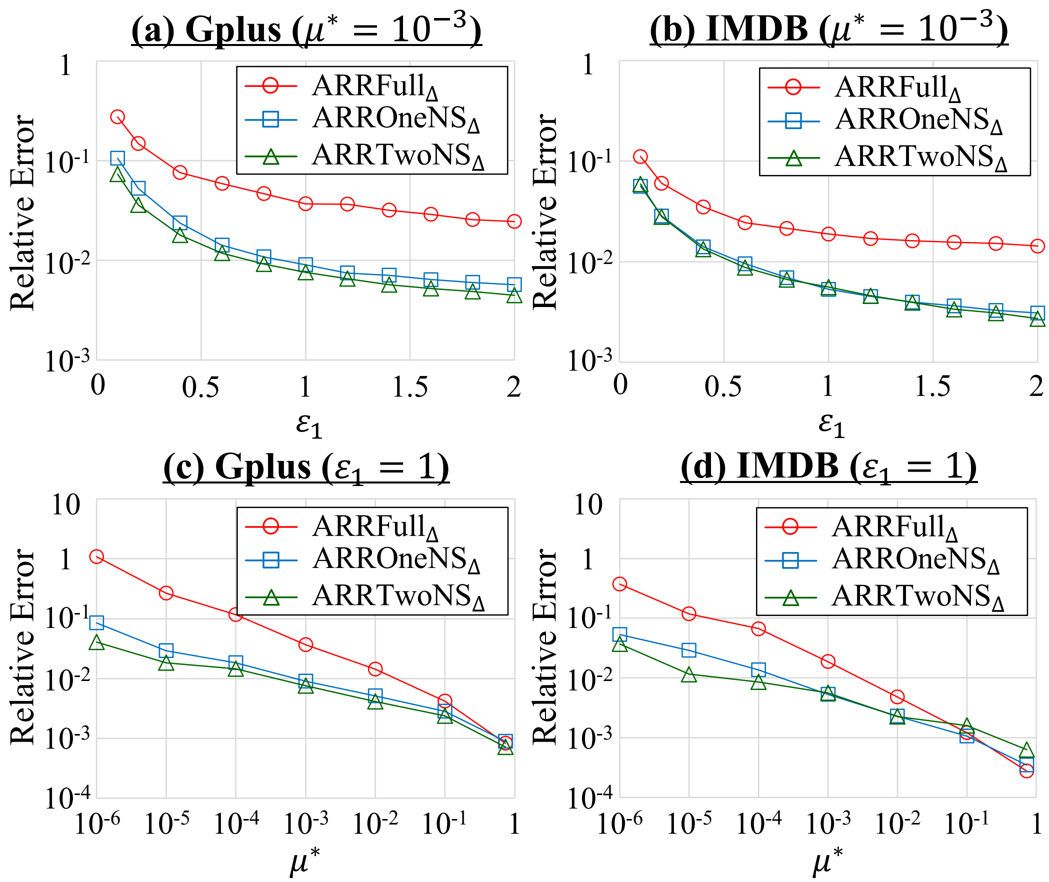


Figure 2.9. Relative error of our three algorithms without the Laplacian noise ($n = 107614$ in Gplus, $n = 896308$ in IMDB).

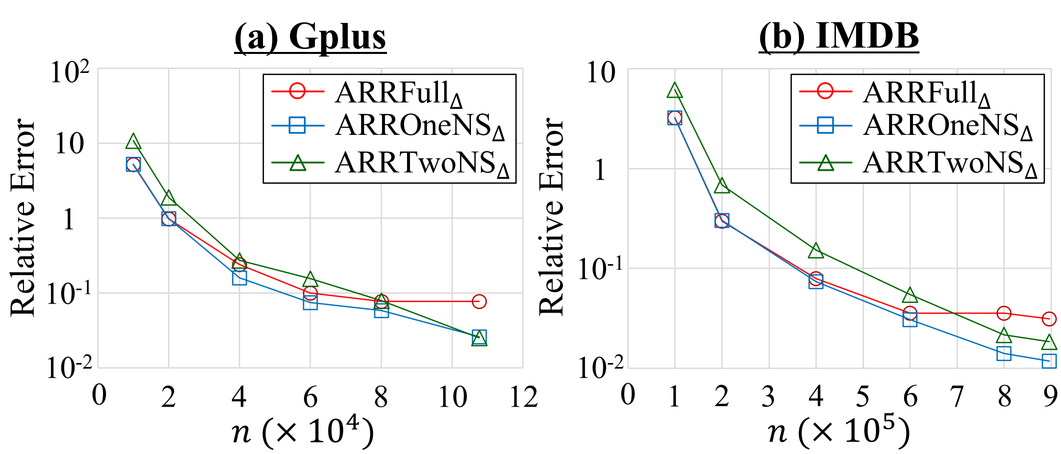


Figure 2.10. Relative error of our three algorithms with double clipping for various values of n ($\varepsilon = 1$, $\mu^* = 10^{-3}$).

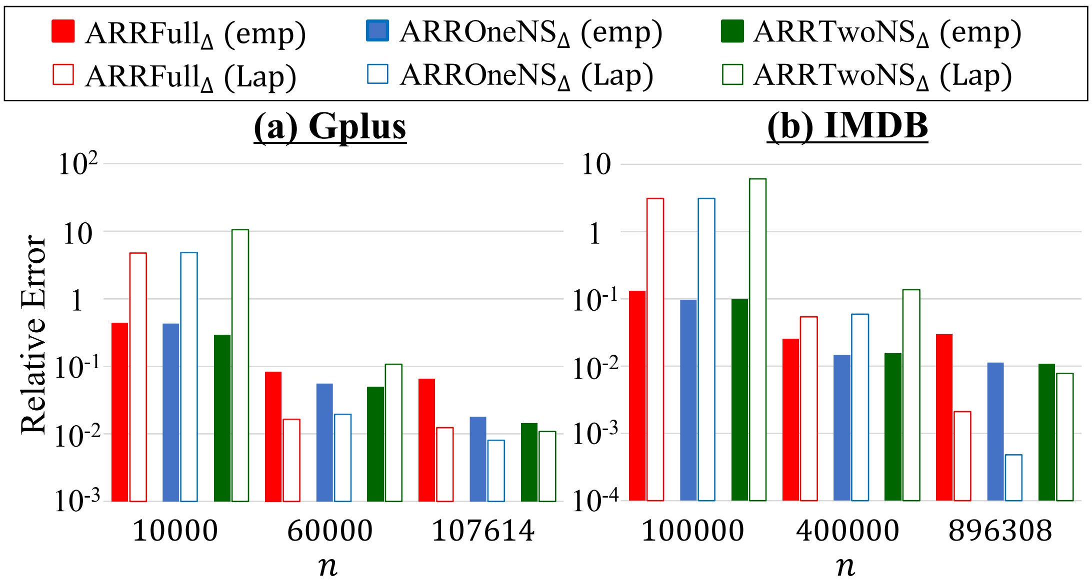


Figure 2.11. Relative error of empirical estimation and the Laplacian noise in our three algorithms with double clipping ($\varepsilon = 1$, $\mu^* = 10^{-3}$).

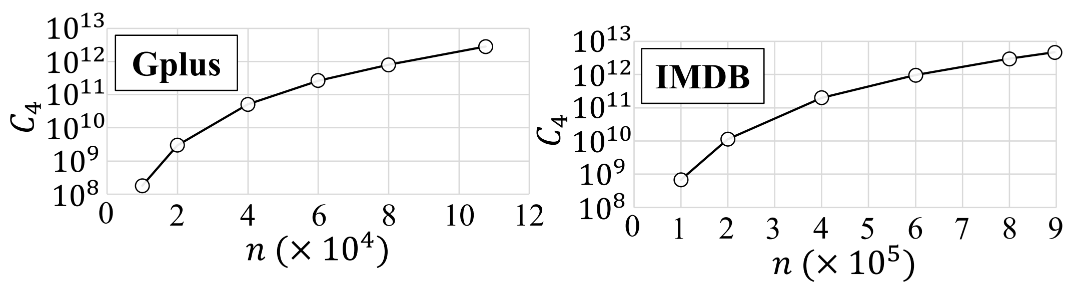


Figure 2.12. #4-cycles C_4.

Chapter 3

Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model

3.1 Introduction

Graph statistics is useful for finding meaningful connection patterns in network data, and subgraph counting is known as a fundamental task in graph analysis. For example, a triangle is a cycle of size three, and a k-star consists of a central node connected to k other nodes. These subgraphs can be used to calculate a clustering coefficient ($= \frac{3\times\#\text{triangles}}{\#\text{2-stars}}$). In a social graph, the clustering coefficient measures the tendency of nodes (users) to form a cluster with each other. It also represents the average probability that a friend's friend is also a friend [158]. Therefore, the clustering coefficient is useful for analyzing the effectiveness of friend suggestions. Another example of the subgraph is a 4-cycle, a cycle of size four. The 4-cycle count is useful for measuring the clustering ability in bipartite graphs (e.g., online dating networks, mentor-student networks [127]) where a triangle never appears [134, 171, 177]. Figure 3.1 shows examples of triangles, 2-stars, and 4-cycles. Although these subgraphs are important for analyzing the connection patterns or clustering tendencies, their exact numbers can leak sensitive edges (friendships) [102].

DP (Differential Privacy) [70, 73] – the gold standard of privacy notions – has been widely used to strongly protect edges in graph data [58, 64, 98, 102, 104, 120, 122, 165, 186, 218, 219]. In particular, recent studies [102, 104, 165, 218, 219] have applied LDP (Local DP) [121] to

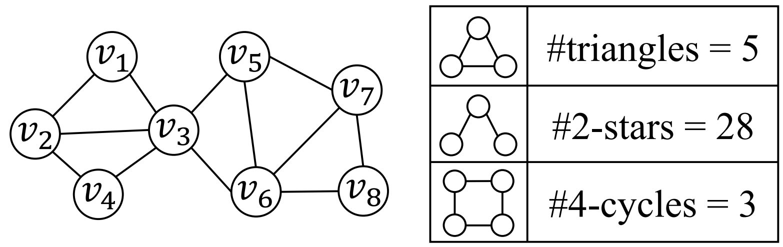


Figure 3.1. Examples of subgraph counts.

graph data. In the graph LDP model, each user obfuscates her neighbor list (friends list) by herself and sends the obfuscated neighbor list to a data collector. Then, the data collector estimates graph statistics, such as subgraph counts. Compared to central DP where a central server has personal data of all users (i.e., the entire graph), LDP does not have a risk that all personal data are leaked from the server by cyberattacks [100] or insider attacks [124]. Moreover, LDP can be applied to decentralized social networks [161, 176] (e.g., diaspora* [60], Mastodon [140]) where no server can access the entire graph; e.g., the entire graph is distributed across many servers, or no server has any original edges. It is reported in [102] that k-star counts can be accurately estimated in this model.

However, it is much more challenging to accurately count more complicated subgraphs such as triangles and 4-cycles under LDP. The root cause of this is its local property – a user cannot see edges between others. For example, user v_1 cannot count triangles or 4-cycles including v_1, as she cannot see edges between others, e.g., (v_2, v_3), (v_2, v_4), and (v_3, v_4). Therefore, the existing algorithms [102, 104, 218, 219] obfuscate each bit of the neighbor list rather than the subgraph count by the RR (Randomized Response) [208], which randomly flips 0/1. As a result, their algorithms suffer from extremely large estimation errors because it makes all edges noisy. Some studies [102, 104] significantly improve the accuracy by introducing an additional round of interaction between users and the data collector. However, multi-round interaction may be impractical in many applications, as it requires a lot of user effort and synchronization; in [102, 104], every user must respond twice, and the data collector must wait for responses from all users in each round.

In this work, we focus on a **one-round** of interaction between users and the data collector and propose accurate subgraph counting algorithms by introducing a recently studied privacy model: the *shuffle model* [82, 88]. In the shuffle model, each user sends her (encrypted) obfuscated data to an intermediate server called the shuffler. Then, the shuffler randomly shuffles the obfuscated data of all users and sends the shuffled data to the data collector (who decrypts them). The shuffling amplifies DP guarantees of the obfuscated data under the assumption that the shuffler and the data collector do not collude with each other. Specifically, it is known that DP strongly protects user privacy when a parameter (a.k.a. privacy budget) ε is small, e.g., $\varepsilon \leq 1$ [131]. The shuffling significantly reduces ε and therefore significantly improves utility at the same value of ε. To date, the shuffle model has been successfully applied to tabular data [146, 202] and gradients [92, 136] in federated learning. We apply the shuffle model to graph data to accurately count subgraphs within one round.

The main challenge in subgraph counting in the shuffle model is that each user's neighbor list is *high-dimensional data*, i.e., n-dim binary string where n is the number of users. Consequently, applying the RR to each bit of the neighbor list, as in the existing work [102, 104, 218, 219], results in an extremely large privacy budget ε even after applying the shuffling (see Section 3.4.1 for more details).

We address this issue by introducing a new, basic technique called *wedge shuffling*. In graphs, a wedge between v_i and v_j is defined by a 2-hop path with endpoints v_i and v_j. For example, in Figure 3.1, there are two wedges between v_2 and v_3: v_2-v_1-v_3 and v_2-v_4-v_3. In other words, users v_1 and v_4 have a wedge between v_2 and v_3, whereas $v_5, \dots, v_8$ do not. Each user obfuscates such wedge information by the RR, and the shuffler randomly shuffles them. Because the wedge information (i.e., whether there is a wedge between a specific user-pair) is *one-dimensional binary data*, it can be sent with small noise and small ε. In addition, the wedge is the main component of several subgraphs, such as triangles, 4-cycles, and 3-hop paths [186]. Since the wedge has little noise, we can accurately count these subgraphs based on wedge shuffling.

We apply wedge shuffling to triangle and 4-cycle counting tasks with several additional techniques. For triangles, we first propose an algorithm that counts triangles involving the user-pair at the endpoints of the wedges by locally sending an edge between the user-pair to the data collector. Then we propose an algorithm to count triangles in the entire graph by sampling disjoint user-pairs, which share no common users (i.e., no user falls in two pairs). We also propose a technique to reduce the variance of the estimate by ignoring sparse user-pairs, where either of the two users has a very small degree. For 4-cycles, we propose an algorithm to calculate an unbiased estimate of the 4-cycle count from that of the wedge count via bias correction.

We provide upper bounds on the estimation error for our triangle and 4-cycles counting algorithms. Through comprehensive evaluation, we show that our algorithms accurately estimate these subgraph counts within one round under the shuffle model.

Our Contributions. Our contributions are as follows:

- We propose a wedge shuffle technique to enable privacy amplification of graph data. To our knowledge, we are the first to shuffle graph data (see Section 3.2 for more details).

- We propose one-round triangle and 4-cycle counting algorithms based on our wedge shuffle technique. For triangles, we propose three additional techniques: sending local edges, sampling disjoint user-pairs, and variance reduction by ignoring sparse user-pairs. For 4-cycles, we propose a bias correction technique. We show upper bounds on the estimation error for each algorithm.

- We evaluate our algorithms using two real graph datasets. Our experimental results show that our one-round shuffle algorithms significantly outperform one-round local algorithms in terms of accuracy and achieve a small estimation error (relative error $\ll 1$) with a reasonable privacy budget, e.g., smaller than 1 in edge DP.

In Appendix C.1, we show that our triangle algorithm is also useful for accurately estimating the clustering coefficient within one round. We can use our algorithms to analyze the clustering tendency or the effectiveness of friend suggestions in decentralized social networks by introducing a shuffler. We implemented our algorithms in C/C++. Our code is available on GitHub [191]. The proofs of all statements in the main body are given in Appendices C.8 and C.9.

3.2 Related Work

Non-private Subgraph Counting. Subgraph counting has been extensively studied in a non-private setting (see [169] for a recent survey). Examples of subgraphs include triangles [24, 76, 126, 211], 4-cycles [23, 118, 138, 143], k-stars [6, 94], and k-hop paths [30, 119].

Here, the main challenge is to reduce the computational time of counting these subgraphs in large-scale graph data. One of the simplest approaches is edge sampling [24, 76, 211], which randomly samples edges in a graph. Edge sampling outperforms other sampling methods (e.g., node sampling, triangle sampling) [211] and is also adopted in [104] for private triangle counting.

Although our triangle algorithm also samples user-pairs, ours is different from edge sampling in two ways. First, our algorithm does not sample an edge but samples a pair of users who may or may not be a friend. Second, our algorithm samples user-pairs that share no common users to avoid the increase of the privacy budget ε as well as to reduce the time complexity (see Section 3.5 for details).

Private Subgraph Counting. Differentially private subgraph counting has been widely studied, and the previous work assumes either the central [64, 120, 122] or local [102, 104, 186, 218, 219] models. The central model assumes a centralized social network and has a data breach issue, as explained in Section 3.1.

Subgraph counting in the local model has recently attracted attention. Sun *et al.* [186] propose subgraph counting algorithms assuming that each user knows all friends' friends. However, this assumption does not hold in many social networks; e.g., Facebook users can change their settings so that anyone cannot see their friend lists. Therefore, we make a minimal assumption – each user knows only her friends.

In this setting, recent studies propose triangle [102, 104, 218, 219] and k-star [102] counting algorithms. For k-stars, Imola *et al.* [102] propose a one-round algorithm that is order optimal and show that it provides a very small estimation error. For triangles, they propose a one-round algorithm that applies the RR to each bit of the neighbor list and then calculates an unbiased estimate of triangles from the noisy graph. We call this algorithm $\mathsf{RR}_{\triangle}$. Imola *et al.* [104] show that $\mathsf{RR}_{\triangle}$ provides a much smaller estimation error than the one-round triangle algorithms in [218, 219]. In [104], they also reduce the time complexity of $\mathsf{RR}_{\triangle}$ by using the ARR (Asymmetric RR), which samples each 1 (edge) after applying the RR. We call this algorithm $\mathsf{ARR}_{\triangle}$. In this paper, we use $\mathsf{RR}_{\triangle}$ and $\mathsf{ARR}_{\triangle}$ as baselines in triangle counting. For 4-cycles, there is no existing algorithm under LDP, to our knowledge. Thus, we compare our shuffle algorithm with its local version, which does not shuffle the obfuscated data.

For triangles, Imola *et al.* also propose a two-round local algorithm in [102] and significantly reduce its download cost in [104]. Although we focus on one-round algorithms, we show in Appendix C.2 that our one-round algorithm is comparable to the two-round algorithm in [104], which requires a lot of user effort and synchronization, in terms of accuracy.

Shuffle Model. The privacy amplification by shuffling has been recently studied in [15, 47, 82, 88]. Among them, the privacy amplification bound by Feldman *et al.* [88] is the state-of-the-art – it provides a smaller ε than other bounds, such as [15, 47, 82]. Girgis *et al.* [93] consider multiple interactions between users and the data collector and show a better bound than the bound in [88] when used with composition. However, the bound in [88] outperforms the bound in [93] when used without composition. Because our work focuses on a single interaction and does not use the composition, we use the bound in [88].

The shuffle model has been applied to tabular data [146, 202] and gradients [92, 136] in federated learning. Meehan *et al.* [146] construct a graph from public auxiliary information and determine a permutation of obfuscated data using the graph to reduce re-identification risks. Liew *et al.* [133] propose network shuffling, which shuffles obfuscated data via random walks on a graph. Note that both [146] and [133] use graph data to shuffle another type of data. To our knowledge, our work is the first to shuffle graph data itself.

3.3 Preliminaries

In this section, we describe some preliminaries for our work. Section 3.3.1 defines the basic notation used in this paper. Sections 3.3.2 and 3.3.3 introduce DP on graphs and the shuffle model, respectively. Section 3.3.4 explains utility metrics.

3.3.1 Notation

Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{N}$, and $\mathbb{Z}_{\geq 0}$ be the sets of real numbers, non-negative real numbers, natural numbers, and non-negative integers, respectively. For $a \in \mathbb{N}$, let $[a]$ be the set of natural numbers that do not exceed a, i.e., $[a] = \{1, 2, \ldots, a\}$.

We consider an undirected social graph $G = (V, E)$, where V represents a set of nodes (users) and $E \subseteq V \times V$ represents a set of edges (friendships). Let $n \in \mathbb{N}$ be the number of nodes in V, and $v_i \in V$ be the i-th node, i.e., $V = \{v_1, \ldots, v_n\}$. Let $I_{-(i,j)}$ be the set of indices of users other than v_i and v_j, i.e., $I_{-(i,j)} = [n] \setminus \{i, j\}$. Let $d_i \in \mathbb{Z}_{\geq 0}$ be a degree of v_i, $d_{avg} \in \mathbb{R}_{\geq 0}$ be the average degree of G, and $d_{max} \in \mathbb{N}$ be the maximum degree of G. In most real graphs, $d_{avg} \ll d_{max} \ll n$ holds. We denote a set of graphs with n nodes by $\mathscr{G}$. Let $f^{\triangle} : \mathscr{G} \to \mathbb{Z}_{\geq 0}$ and $f^{\square} : \mathscr{G} \to \mathbb{Z}_{\geq 0}$ be triangle and 4-cycle count functions, respectively. The triangle count function takes $G \in \mathscr{G}$ as input and outputs the number $f^{\triangle}(G)$ of triangles in G, whereas the 4-cycle count function takes G as input and outputs the number $f^{\square}(G)$ of 4-cycles.

Let $\mathbf{A} = (a_{i,j}) \in \{0,1\}^{n \times n}$ be an adjacency matrix corresponding to G. If $(v_i, v_j) \in E$, then $a_{i,j} = 1$; otherwise, $a_{i,j} = 0$. We call $a_{i,j}$ an *edge indicator*. Let $\mathbf{a}_i \in \{0,1\}^n$ be a neighbor list of user v_i, i.e., the i-th row of $\mathbf{A}$. Table 3.1 shows the basic notation in this paper.

Table 3.1. Basic notation in this paper.

Symbol	Description
$G=(V,E)$	Undirected social graph.
n	Number of nodes (users).
v_i	i-th user in V, i.e., $V=\{v_1,\dots,v_n\}$.
$I_{-(i,j)}$	$=[n]\setminus\{i,j\}$.
d_i	Degree of v_i.
d_{avg}	Average degree in G.
d_{max}	Maximum degree in G.
$\mathscr{G}$	Set of possible graphs with n nodes.
$f^{\triangle}(G)$	Triangle count in graph G.
$f^{\square}(G)$	4-cycle count in graph G.
$\mathbf{A}=(a_{i,j})$	Adjacency matrix.
$\mathbf{a}_i$	Neighbor list of v_i, i.e., the i-th row of $\mathbf{A}$.

3.3.2 Differential Privacy

DP and LDP. We use differential privacy, and more specifically (ε,δ)-DP [73], as a privacy metric:

Definition 9 ((ε,δ)-DP [73])**.** *Let $n\in\mathbb{N}$ be the number of users. Let $\varepsilon\in\mathbb{R}_{\geq 0}$ and $\delta\in[0,1]$. Let $\mathscr{X}$ be the set of input data for each user. A randomized algorithm $\mathscr{M}$ with domain $\mathscr{X}^n$ provides* (ε,δ)-DP *if for any neighboring databases $D,D'\in\mathscr{X}^n$ that differ in a single user's data and any $S\subseteq \mathrm{Range}(\mathscr{M})$,*

$$\Pr[\mathscr{M}(D)\in S]\leq e^{\varepsilon}\Pr[\mathscr{M}(D')\in S]+\delta.$$

(ε,δ)-DP guarantees that two neighboring datasets D and D' are almost equally likely when ε and δ are close to 0. The parameter ε is called the privacy budget. It is well known that $\varepsilon\leq 1$ is acceptable and $\varepsilon\geq 5$ is unsuitable in many practical scenarios [131]. In addition, the parameter δ needs to be much smaller than $\frac{1}{n}$ [17, 73].

LDP [121] is a special case of DP where $n=1$. In this case, a randomized algorithm is called a *local randomizer*. We denote the local randomizer by $\mathscr{R}$ to distinguish it from the randomized algorithm $\mathscr{M}$ in the central model. Formally, LDP is defined as follows:

Definition 10 (ε-LDP [121])**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. Let $\mathscr{X}$ be the set of input data for each user. A local randomizer $\mathscr{R}$ with domain $\mathscr{X}$ provides* ε-LDP *if for any $x, x' \in \mathscr{X}$ and any $S \subseteq$* Range($\mathscr{R}$)*,*

$$\Pr[\mathscr{R}(x) \in S] \leq e^{\varepsilon} \Pr[\mathscr{R}(x') \in S]. \tag{3.1}$$

Randomized Response. We use Warner's RR (Randomized Response) [208] to provide LDP. Given $\varepsilon \in \mathbb{R}_{\geq 0}$, Warner's RR $\mathscr{R}_{\varepsilon}^{W} : \{0,1\} \rightarrow \{0,1\}$ maps $x \in \{0,1\}$ to $y \in \{0,1\}$ with the probability:

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

$\mathscr{R}_{\varepsilon}^{W}$ provides ε-LDP in Definition 10, where $\mathscr{X} = \{0,1\}$. We refer to Warner's RR $\mathscr{R}_{\varepsilon}^{W}$ with parameter ε as ε-*RR*.

DP on Graphs. For graphs, we can consider two types of DP: *edge DP* and *node DP* [98, 168]. Edge DP hides the existence of one edge, whereas node DP hides the existence of one node along with its adjacent edges. In this paper, we focus on edge DP because existing one-round local triangle counting algorithms [102, 104, 218, 219] use edge DP. In other words, we are interested in how much the estimation error is reduced at the same value of ε in edge DP by shuffling. Although node DP is much stronger than edge DP, it is much harder to attain and often results in a much larger ε [42, 175]. Thus, we leave an algorithm for shuffle node DP with small ε (e.g., $\varepsilon \leq 1$) for future work. Another interesting avenue of future work is establishing a lower bound on the estimation error for node DP.

Edge DP assumes that anyone (except for user v_i) can be an adversary who infers edges of user v_i and that the adversary can obtain all edges except for edges of v_i as background knowledge. Note that the central and local models have different definitions of neighboring data in edge DP. Specifically, edge DP in the central model [168] considers two graphs that differ in one edge. In contrast, edge LDP [165] considers two neighbor lists that differ in one bit:

Definition 11 ((ε, δ)-edge DP [168])**.** *Let $n \in \mathbb{N}$, $\varepsilon \in \mathbb{R}_{\geq 0}$, and $\delta \in [0,1]$. A randomized algorithm $\mathscr{M}$ with domain $\mathscr{G}$ provides* (ε, δ)-edge DP *if for any two neighboring graphs $G, G' \in \mathscr{G}$ that differ in **one edge** and any $S \subseteq \mathrm{Range}(\mathscr{M})$,*

$$\Pr[\mathscr{M}(G) \in S] \leq e^{\varepsilon} \Pr[\mathscr{M}(G') \in S] + \delta.$$

Definition 12 (ε-edge LDP [165])**.** *Let $\varepsilon \in \mathbb{R}_{\geq 0}$. A local randomizer $\mathscr{R}$ with domain $\{0,1\}$ 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 $S \subseteq \mathrm{Range}(\mathscr{R})$,*

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

As with edge LDP, we define *element DP*, which considers two adjacency matrices that differ in one bit, in the central model:

Definition 13 ((ε, δ)-element DP)**.** *Let $n \in \mathbb{N}$, $\varepsilon \in \mathbb{R}_{\geq 0}$, and $\delta \in [0,1]$. A randomized algorithm $\mathscr{M}$ with domain $\mathscr{G}$ provides* (ε, δ)-element DP *if for any two neighboring graphs $G, G' \in \mathscr{G}$ that differ in **one bit** in the corresponding adjacency matrices $\mathbf{A}, \mathbf{A}' \in \{0,1\}^{n \times n}$ and any $S \subseteq \mathrm{Range}(\mathscr{M})$,*

$$\Pr[\mathscr{M}(G) \in S] \leq e^{\varepsilon} \Pr[\mathscr{M}(G') \in S] + \delta.$$

Although element DP and edge DP have different definitions of neighboring data, they are closely related to each other:

Proposition 6. *If a randomized algorithm $\mathscr{M}$ provides (ε, δ)-element DP, it also provides $(2\varepsilon, 2\delta)$-edge DP.*

Proof. Adding or removing one edge affects two bits in an adjacency matrix. Thus, by group privacy [73], any (ε,δ)-element DP algorithm $\mathscr{M}$ provides $(2\varepsilon,2\delta)$-edge DP. □

Similarly, if a randomized algorithm $\mathscr{M}$ in the central model applies a local randomizer $\mathscr{R}$ providing ε-edge LDP to each neighbor list $\mathbf{a}_i$ $(1 \leq i \leq n)$, it provides 2ε-edge DP [102].

In this work, we use the shuffling technique to provide (ε,δ)-element DP and then Proposition 6 to provide $(2\varepsilon,2\delta)$-edge DP. We also compare our shuffle algorithms providing (ε,δ)-element DP and $(2\varepsilon,2\delta)$-edge DP with local algorithms providing ε-edge LDP and 2ε-edge DP to see how much the estimation error is reduced by introducing the shuffle model and a very small δ ($\ll \frac{1}{n}$).

3.3.3 Shuffle Model

We consider the following shuffle model. Each user $v_i \in V$ obfuscates her personal data using a local randomizer $\mathscr{R}$ providing ε_L-LDP for $\varepsilon_L \in \mathbb{R}_{\geq 0}$. Note that $\mathscr{R}$ is common to all users. User v_i encrypts the obfuscated data and sends it to a shuffler. Then, the shuffler randomly shuffles the encrypted data and sends the results to a data collector. Finally, the data collector decrypts them. The common assumption in the shuffle model is that the shuffler and the data collector do not collude with each other. Under this assumption, the shuffler cannot access the obfuscated data, and the data collector cannot link the obfuscated data to the users. Hereinafter, we omit the encryption/decryption process because it is clear from the context.

We use the privacy amplification result by Feldman *et al.* [88]:

Theorem 12 (Privacy amplification by shuffling [88])**.** *Let $n \in \mathbb{N}$ and $\varepsilon_L \in \mathbb{R}_{\geq 0}$. Let $\mathscr{X}$ be the set of input data for each user. Let $x_i \in \mathscr{X}$ be input data of the i-th user, and $x_{1:n} = (x_1, \cdots, x_n) \in \mathscr{X}^n$. Let $\mathscr{R} : \mathscr{X} \to \mathscr{Y}$ be a local randomizer providing ε_L-LDP. Let $\mathscr{M}_S : \mathscr{X}^n \to \mathscr{Y}^n$ be an algorithm that given a dataset $x_{1:n}$, computes $y_i = \mathscr{R}(x_i)$ for $i \in [n]$, samples a uniform random permutation π over $[n]$, and outputs $y_{\pi(1)}, \ldots, y_{\pi(n)}$. Then for any $\delta \in [0,1]$ such that $\varepsilon_L \leq \log(\frac{n}{16\log(2/\delta)})$,*

$\mathscr{M}_S$ *provides* (ε, δ)*-DP, where*

$$\varepsilon = f(n, \varepsilon_L, \delta) \tag{3.2}$$

and

$$f(n, \varepsilon_L, \delta) = \log\left(1 + \frac{e^{\varepsilon_L} - 1}{e^{\varepsilon_L} + 1}\left(\frac{8\sqrt{e^{\varepsilon_L}\log(4/\delta)}}{\sqrt{n}} + \frac{8e^{\varepsilon_L}}{n}\right)\right). \tag{3.3}$$

Thanks to the shuffling, the shuffled data $y_{\pi(1)}, \ldots, y_{\pi(n)}$ available to the data collector provides (ε, δ)-DP, where $\varepsilon \ll \varepsilon_L$.

Feldman *et al.* [88] also propose an efficient method to numerically compute a tighter upper bound than the closed-form upper bound in Theorem 12. We use both the closed-form and numerical upper bounds in our experiments. Specifically, we use the numerical upper bounds in Section 3.7 and compare the numerical bound with the closed-form bound in Appendix C.3.

Assume that ε and δ in (3.3) are constants. Then, by solving for ε_L and changing to big O notation, we obtain $\varepsilon_L = \log(n) + O(1)$. This is consistent with the upper bound $\varepsilon = O(e^{\varepsilon_L/2}/\sqrt{n})$ in [88], from which we obtain $\varepsilon_L = \log(n) + O(1)$. Similarly, the privacy amplification bound in [47] can also be expressed as $\varepsilon_L = \log(n) + O(1)$. We use the bound in [88] because it is the state-of-the-art, as described in Section 3.2.

3.3.4 Utility Metrics

We use the MSE (Mean Squared Error) in our theoretical analysis and the relative error in our experiments. The MSE is the expectation of the squared error between a true value and its estimate. Let $f : \mathscr{G} \to \mathbb{Z}_{\geq 0}$ be a subgraph count function that can be instantiated by $f^{\triangle}$ or $f^{\square}$. Let $\hat{f} : \mathscr{G} \to \mathbb{R}$ be the corresponding estimator. Let $\mathsf{MSE} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be the MSE function, which maps the estimate $\hat{f}(G)$ to the MSE. Then the MSE can be expressed as $\mathsf{MSE}(\hat{f}(G)) = \mathbb{E}[(f(G) - \hat{f}(G))^2]$, where the expectation is taken over the randomness in the estimator $\hat{f}$. By the bias-variance decomposition [156], the MSE can be expressed as a summation of the squared bias $(\mathbb{E}[\hat{f}(G)] - f(G))^2$ and the variance $\mathbb{V}[\hat{f}(G)] = \mathbb{E}[(\hat{f}(G) - \mathbb{E}[\hat{f}(G)])]^2$. Thus, for an unbiased estimator $\hat{f}$ satisfying $\mathbb{E}[\hat{f}(G)] = f(G)$, the MSE is equal to the variance, i.e., $\mathsf{MSE}(\hat{f}(G)) = \mathbb{V}[\hat{f}(G)]$.

Although the MSE is suitable for theoretical analysis, it tends to be large when the number n of users is large. This is because the true triangle and 4-cycle counts are very large when n is large – $f^{\triangle}(G) = O(nd_{max}^2)$ and $f^{\square}(G) = O(nd_{max}^3)$. Therefore, we use the relative error in our experiments. The relative error is an absolute error divided by the true value and is given by $\frac{|f^{\triangle}(G) - \hat{f}^{\triangle}(G)|}{\min\{f^{\triangle}(G), \eta\}}$, where $\eta \in \mathbb{R}_{\geq 0}$ is a small positive value. Following the convention [29, 41, 215], we set $\eta = \frac{n}{1000}$.

When the relative error is well below 1, the estimate is accurate. Note that the absolute error smaller than 1 would be impossible under DP with meaningful ε (e.g., $\varepsilon \leq 1$), as we consider counting queries. However, the relative error (= absolute error / true count) much smaller than 1 is possible under DP with meaningful ε.

3.4 Shuffle Model for Graphs

In this work, we apply the shuffle model to graph data to accurately estimate subgraph counts, such as triangles and 4-cycles. Section 3.4.1 explains our technical motivation. In particular, we explain why it is challenging to apply the shuffle model to graph data. Section 3.4.2 proposes a wedge shuffle technique to overcome the technical challenge.

3.4.1 Our Technical Motivation

The shuffle model has been introduced to dramatically reduce the privacy budget ε (hence the estimation error at the same ε) in tabular data [146, 202] or gradients [92, 136]. However, it is very challenging to apply the shuffle model to graph data, as explained below.

Figure 3.2 shows the shuffle model for graph data, where each user v_i has her neighbor list $\mathbf{a}_i \in \{0,1\}^n$. The main challenge here is that the shuffle model uses a *standard* definition

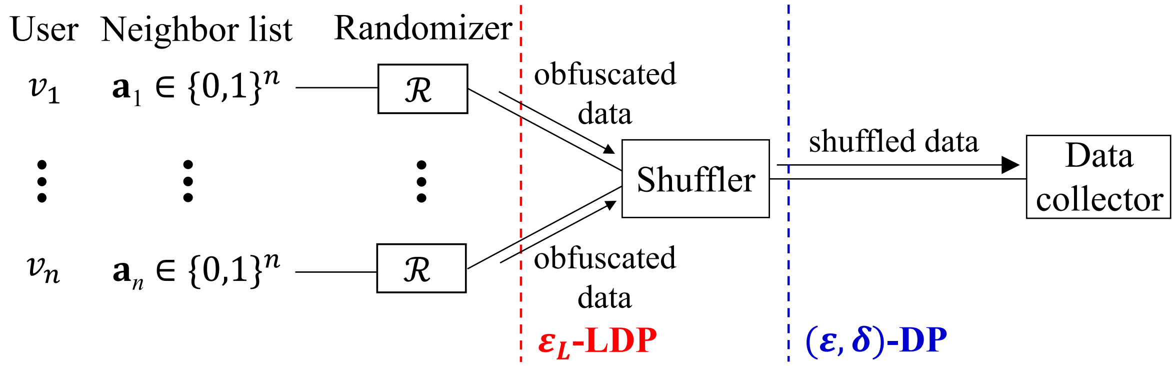


Figure 3.2. Shuffle model for graphs.

of LDP for the local randomizer and that a neighbor list is *high-dimensional data*, i.e., n-dim binary string. Specifically, LDP in Definition 10 requires any pair of inputs x and x' to be indistinguishable; i.e., the inequality (3.1) must hold for all pairs of possible inputs. Thus, if we use the entire neighbor list as input data (i.e., $\mathbf{a}_i = x_i$ in Theorem 12), either privacy or utility is destroyed for large n.

To illustrate this, consider the following example. Assume that $n = 10^5$ and $\delta = 10^{-8}$. Each user v_i applies ε_0-RR with $\varepsilon_0 = 1$ to each bit of her neighbor list $\mathbf{a}_i$. This mechanism is called the randomized neighbor list [165] and provides ε_0-edge LDP. However, the privacy budget ε_L in the standard LDP (Definition 10) is extremely large – by group privacy [73], $\varepsilon_L = n\varepsilon_0 = 10^5$. Because ε_L is much larger than $\log(\frac{n}{16\log(2/\delta)}) = 8.09$, we cannot use the privacy amplification result in Theorem 12. This is evident from the fact that the shuffled data $y_{\pi(1)}, \dots, y_{\pi(n)}$ are easily re-identified when n is large. If we use ε_0-RR with $\varepsilon_0 = \frac{1}{n}$, we can use the amplification result (as $\varepsilon_L = n\varepsilon_0 = 1$). However, it makes obfuscated data almost a random string and destroys the utility because ε_0 is too small.

In this work, we address this issue by introducing a basic technique, which we call *wedge shuffling*.

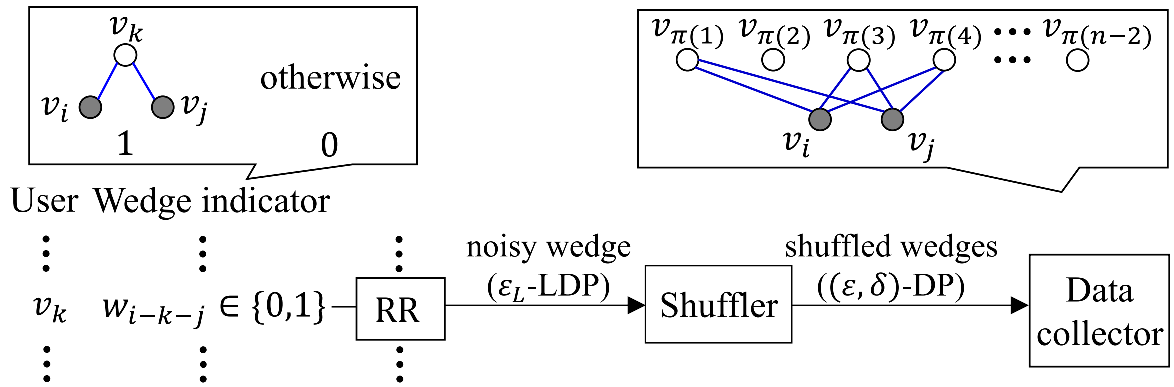


Figure 3.3. Overview of wedge shuffling with inputs v_i and v_j.

3.4.2 Our Approach: Wedge Shuffling

Figure 3.3 shows the overview of our wedge shuffle technique. This technique calculates the number of wedges (2-hop paths) between a specific pair of users v_i and v_j.

Algorithm 6 shows our wedge shuffle algorithm, which we call WS. Given users v_i and v_j, each of the remaining users v_k ($k \neq i, j$) calculates a *wedge indicator* $w_{i-k-j} = a_{k,i}a_{k,j}$, which takes 1 if a wedge v_i-v_k-v_j exists and 0 otherwise (line 2). Then, v_k obfuscates w_{i-k-j} using ε_L-RR and sends it to the shuffler (line 3). The shuffler randomly shuffles the noisy wedges using a random permutation π over $I_{-(i,j)}$ $(= [n] \setminus \{i,j\})$ to provide (ε,δ)-DP with $\varepsilon \ll \varepsilon_L$ (line 5). Finally, the shuffler sends the shuffled wedges to the data collector (line 6). The only information available to the data collector is the number of wedges from v_i to v_j, i.e., common friends of v_i and v_j.

Our wedge shuffling has two main features. First, the wedge indicator w_{i-k-j} is *one-dimensional binary data*. Therefore, it can be sent with small noise and small ε, unlike the n-dimensional neighbor list. For example, when $n = 10^5$, $\delta = 10^{-8}$, and $\varepsilon = 1$, the value of ε_L in (3.2) and (3.3) is $\varepsilon_L = 5.44$. In this case, ε_L-RR rarely flips w_{i-k-j} – the flip probability is 0.0043. In other words, the shuffled wedges are almost free of noise.

Second, the wedge is the main component of many subgraphs such as triangles, k-triangles [120], 3-hop paths [186], and 4-cycles. For example, a triangle consists of one wedge

Data: Adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, $\varepsilon_L \in \mathbb{R}_{\geq 0}$, user-pair (v_i, v_j).
Result: Shuffled wedges $\{y_{\pi(k)} | k \in I_{-(i,j)}\}$.
foreach $k \in I_{-(i,j)}$ **do**
    $[v_k]$ $w_{i-k-j} \leftarrow a_{k,i}a_{k,j}$;
    $[v_k]$ $y_k \leftarrow \mathscr{R}^W_{\varepsilon_L}(x)(w_{i-k-j})$; Send y_k to the shuffler;
end
[s] Sample a random permutation π over $I_{-(i,j)}$;
[s] Send $\{y_{\pi(k)} | k \in I_{-(i,j)}\}$ to the data collector;
[d] **return** $\{y_{\pi(k)} | k \in I_{-(i,j)}\}$

Algorithm 6: Our wedge shuffle algorithm WS. $[v_k]$, [s], and [d] represent that the process is run by user v_i, the shuffler, and the data collector, respectively.

and one edge, e.g., $v_i - v_{\pi(3)} - v_j$ and (v_i, v_j) in Figure 3.3. More generally, a k-triangle consists of k triangles sharing one edge. Thus, it can be decomposed into k wedges and one edge. A 3-hop path consists of one wedge and one edge. A 4-cycle consists of two wedges, e.g., $v_i - v_{\pi(1)} - v_j$ and $v_i - v_{\pi(3)} - v_j$ in Figure 3.3. Because the shuffled wedges have little noise, we can accurately count these subgraphs based on wedge shuffling, compared to local algorithms in which all edges are noisy.

In this work, we focus on triangles and 4-cycles and present algorithms with upper bounds on the estimation error based on our wedge shuffle technique.

3.5 Triangle Counting Based on Wedge Shuffling

Based on our wedge shuffle technique, we first propose a one-round triangle counting algorithm. Section 3.5.1 describes the overview of our algorithms. Section 3.5.2 proposes an algorithm for counting triangles involving a specific user-pair as a building block of our triangle counting algorithm. Section 3.5.3 proposes our triangle counting algorithm. Section 3.5.4 proposes a technique to significantly reduce the variance in our triangle counting algorithm. Section 3.5.5 summarizes the performance guarantees of our triangle algorithms.

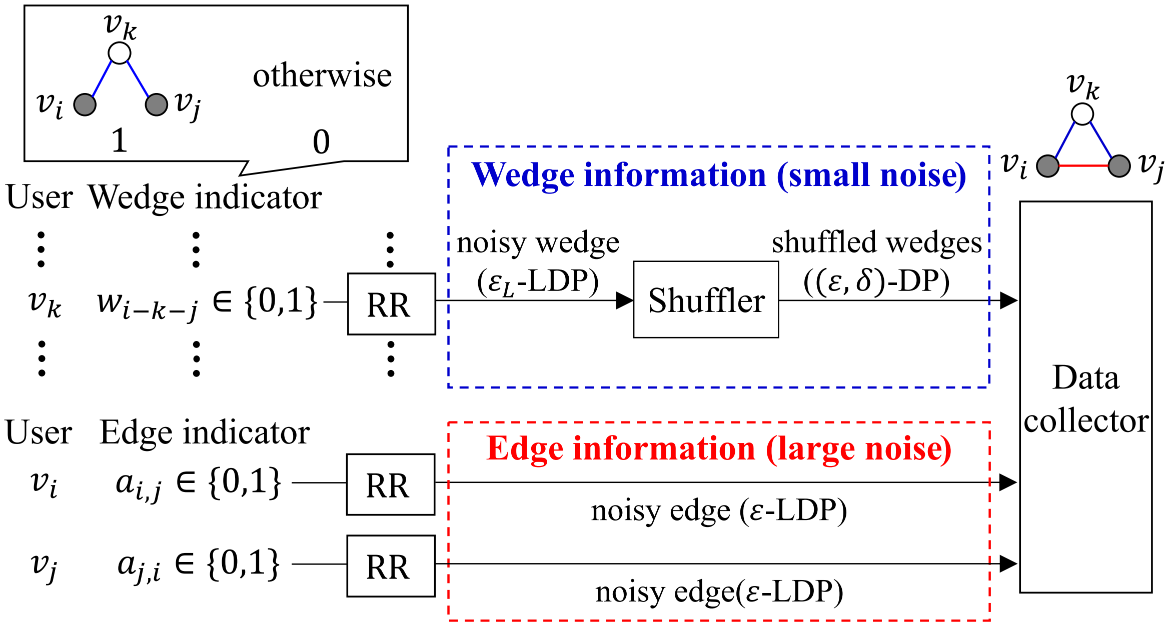


Figure 3.4. Overview of our WSLE (Wedge Shuffling with Local Edges) algorithm with inputs v_i and v_j.

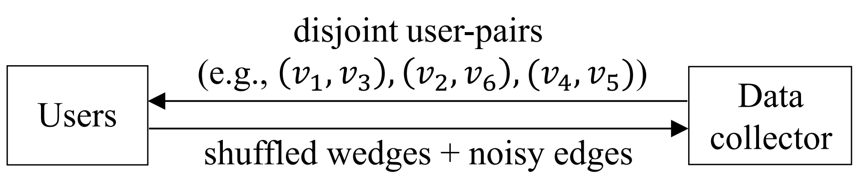


Figure 3.5. Overview of our triangle counting algorithm. We use our WSLE algorithm with each user-pair.

3.5.1 Overview

Our wedge shuffle technique tells the data collector the number of common friends of v_i and v_j. However, this information is not sufficient to count triangles in the entire graph. Therefore, we introduce three additional techniques: *(i) sending local edges*, *(ii) sampling disjoint user-pairs*, and *(iii) variance reduction by ignoring sparse user-pairs*. Below, we briefly explain each technique.

Sending Local Edges. First, we consider the problem of counting triangles involving a specific user-pair (v_i, v_j) and propose an algorithm to send *local edges* between v_i and v_j, along with shuffled wedges, to the data collector. We call this the WSLE (Wedge Shuffling with Local

Edges) algorithm.

Figure 3.4 shows the overview of WSLE. In this algorithm, users v_i and v_j obfuscate edge indicators $a_{i,j}$ and $a_{j,i}$, respectively, using ε-RR and send them to the data collector directly (or through the shuffler without shuffling). Then, the data collector calculates an unbiased estimate of the triangle count from the shuffled wedges and the noisy edges. Because ε is small, a large amount of noise is added to the edge indicators. However, *only one edge* is noisy (the other two have little noise) in any triangle the data collector sees. This brings us an advantage over the one-round local algorithms in which all three edges are noisy.

Sampling Disjoint User-Pairs. Next, we consider the problem of counting triangles in the entire graph G. A naive solution to this problem is to use our WSLE algorithm with all $\binom{n}{2}$ user-pairs as input. However, it results in very large ε and δ because it uses each element of the adjacency matrix $\mathbf{A}$ many times. To address this issue, we propose a triangle counting algorithm that samples disjoint user-pairs, ensuring that no user falls in two pairs.

Figure 3.5 shows the overview of our triangle algorithm. The data collector sends the sampled user-pairs to users. Then, users apply WSLE with each user-pair and send the results to the data collector. Finally, the data collector calculates an unbiased estimate of the triangle count from the results. Because our triangle algorithm uses each element of $\mathbf{A}$ *at most once*, it provides (ε, δ)-element DP hence $(2\varepsilon, 2\delta)$-edge DP. In addition, our triangle algorithm reduces the time complexity from $O(n^3)$ to $O(n^2)$ by sampling user-pairs rather than using all user-pairs.

We prove that the MSE of our triangle counting algorithm is $O(n^3)$ when we ignore the factor of d_{max}. When we do not shuffle wedges, the MSE is $O(n^4)$. In addition, the MSE of the existing one-round local algorithm [104] with the same time complexity is $O(n^6)$, as proved in Appendix C.7. Thus, our algorithm provides a dramatic improvement over the local algorithms.

Variance Reduction. Although our algorithm dramatically improves the MSE, the factor of n^3 may still be large. Therefore, we propose a variance reduction technique that ignores sparse user-pairs, where either of the two users has a very small degree. Our basic idea is that the

Data: Adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, $\varepsilon \in \mathbb{R}_{\geq 0}$, $\delta \in [0,1]$, user-pair (v_i, v_j).
Result: Estimate $\hat{f}^{\triangle}_{i,j}(G)$ of the number $f^{\triangle}_{i,j}(G)$ of triangles involving (v_i, v_j).
1 $\varepsilon_L \leftarrow$ `LocalPrivacyBudget`(n, ε, δ);
`/* Wedge shuffling */`
2 $\{y_{\pi(k)} | k \in I_{-(i,j)}\} \leftarrow \mathsf{WS}(\mathbf{A}, \varepsilon_L, (v_i, v_j))$;
`/* Send local edges */`
3 $[v_i]$ $z_i \leftarrow \mathscr{R}^W_\varepsilon(x)(a_{i,j})$; Send z_i to the data collector;
4 $[v_j]$ $z_j \leftarrow \mathscr{R}^W_\varepsilon(x)(a_{j,i})$; Send z_j to the data collector;
`/* Calculate an unbiased estimate */`
5 [d] $q_L \leftarrow \frac{1}{e^{\varepsilon_L}+1}$; $q \leftarrow \frac{1}{e^{\varepsilon}+1}$;
6 [d] $\hat{f}^{\triangle}_{i,j}(G) \leftarrow \frac{(z_i+z_j-2q)\sum_{k\in I_{-(i,j)}}(y_k - q_L)}{2(1-2q)(1-2q_L)}$;
7 [d] **return** $\hat{f}^{\triangle}_{i,j}(G)$

Algorithm 7: WSLE (Wedge Shuffling with Local Edges). WS is shown in Algorithm 6.

number of triangles involving such a user-pair is very small and can be approximated by 0. By ignoring the sparse user-pairs, we can significantly reduce the variance at the cost of introducing a small bias. We prove that our variance reduction technique reduces the MSE from $O(n^3)$ to $O(n^\gamma)$ where $\gamma \in [2,3)$ and makes one-round triangle counting more accurate.

3.5.2 WSLE (Wedge Shuffling with Local Edges)

Algorithm. We first propose the WSLE algorithm as a building block of our triangle counting algorithm. WSLE counts triangles involving a specific user-pair (v_i, v_j).

Algorithm 7 shows WSLE. Let $f^{\triangle}_{i,j} : \mathscr{G} \rightarrow \mathbb{Z}_{\geq 0}$ be a function that takes $G \in \mathscr{G}$ as input and outputs the number $f^{\triangle}_{i,j}(G)$ of triangles involving (v_i, v_j) in G. Let $\hat{f}^{\triangle}_{i,j}(G) \in \mathbb{R}$ be an estimate of $f^{\triangle}_{i,j}(G)$.

We first call the function `LocalPrivacyBudget`, which calculates a local privacy budget ε_L from n, ε, and δ (line 1). Specifically, this function calculates ε_L such that ε is a closed-form upper bound (i.e., $\varepsilon = f(n-2, \varepsilon_L, \delta)$ in (3.2)) or numerical upper bound in the shuffle model with $n-2$ users. Given ε_L, we can easily calculate the closed-form or numerical upper bound ε by (3.3) and the open source code in [88][1], respectively. Thus, we can also easily calculate ε_L

[1]https://github.com/apple/ml-shuffling-amplification.

from ε by calculating a lookup table for pairs $(\varepsilon, \varepsilon_L)$ in advance.

Then, we run our wedge shuffle algorithm WS in Algorithm 6 (line 2); i.e., each user $v_k \in I_{-(i,j)}$ sends her obfuscated wedge indicator $y_k = \mathscr{R}^W_{\varepsilon_L}(w_{i-k-j})$ to the shuffler, and the shuffler sends shuffled wedge indicators $\{y_{\pi(k)} | k \in I_{-(i,j)}\}$ to the data collector. Meanwhile, user v_i obfuscates her edge indicator $a_{i,j}$ using ε-RR $\mathscr{R}^W_{\varepsilon}$ and sends the result $z_i = \mathscr{R}^W_{\varepsilon}(a_{i,j})$ to the data collector (line 3). Similarly, v_j sends $z_j = \mathscr{R}^W_{\varepsilon}(a_{j,i})$ to the data collector (line 4).

Finally, the data collector estimates $f^{\triangle}_{i,j}(G)$ from $\{y_{\pi(k)} | k \in I_{-(i,j)}\}$, z_i, and z_j. Specifically, the data collector calculates the estimate $\hat{f}^{\triangle}_{i,j}(G)$ as follows:

$$\hat{f}^{\triangle}_{i,j}(G) = \frac{(z_i + z_j - 2q) \sum_{k \in I_{-(i,j)}} (y_k - q_L)}{2(1-2q)(1-2q_L)}, \tag{3.4}$$

where $q_L = \frac{1}{e^{\varepsilon_L}+1}$ and $q = \frac{1}{e^{\varepsilon}+1}$ (lines 5-6). Note that this estimate involves simply summing over the set $\{y_{\pi(k)}\}$ and does not require knowing the value of π. This is consistent with the shuffle model. As we prove later, $\hat{f}^{\triangle}_{i,j}(G)$ in (3.4) is an unbiased estimate of $f^{\triangle}_{i,j}(G)$.

Theoretical Properties. Below, we show some theoretical properties of WSLE. First, we prove that the estimate $\hat{f}^{\triangle}_{i,j}(G)$ is unbiased:

Theorem 13. *For any indices $i, j \in [n]$, the estimate produced by WSLE satisfies $\mathbb{E}[\hat{f}^{\triangle}_{i,j}(G)] = f^{\triangle}_{i,j}(G)$.*

Next, we show the MSE (= variance). Recall that in the shuffle model, $\varepsilon_L = \log n + O(1)$ when ε and δ are constants. We show the MSE for a general case and for the shuffle model:

Theorem 14. *For any indices $i, j \in [n]$, the estimate produced by WSLE provides the following utility guarantee:*

$$\begin{aligned} & MSE(\hat{f}^{\triangle}_{i,j}(G)) = \mathbb{V}[\hat{f}^{\triangle}_{i,j}(G)] \\ & \leq \frac{nq_L + q(1-2q_L)^2 d_{max}^2}{(1-2q)^2(1-2q_L)^2} \triangleq err_{WSLE}(n, d_{max}, q, q_L). \end{aligned} \tag{3.5}$$

When ε and δ are constants and $\varepsilon_L = \log n + O(1)$, we have

$$err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) = O(d_{max}^2). \tag{3.6}$$

The equation (3.6) follows from (3.5) because $q_L = \frac{1}{e^{\varepsilon_L}+1} = \frac{1}{ne^{O(1)}+1}$. Because WSLE is a building block for our triangle counting algorithms, we introduce the notation

$$err_{\mathsf{WSLE}}(n, d_{max}, q, q_L)$$

for our upper bound in (3.5). Observing (3.5), if we do not use the shuffling technique (i.e., $\varepsilon_L = \varepsilon$), then $err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) = O(n + d_{max}^2)$ when we treat ε and δ as constants. In contrast, in the shuffle model where we have $\varepsilon_L = \log n + O(1)$, then $err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) = O(d_{max}^2)$. This means that wedge shuffling reduces the MSE from $O(n + d_{max}^2)$ to $O(d_{max}^2)$, which is significant when $d_{max} \ll n$.

3.5.3 Triangle Counting

Algorithm. Based on WSLE, we propose an algorithm that counts triangles in the entire graph G. We denote this algorithm by $\mathsf{WShuffle}_\triangle$, as it applies wedge shuffling to triangle counting.

Algorithm 8 shows $\mathsf{WShuffle}_\triangle$. First, the data collector samples disjoint user-pairs, ensuring that no user falls in two pairs. Specifically, it calls the function `RandomPermutation`, which samples a uniform random permutation σ over $[n]$ (line 1). Then, it samples disjoint user-pairs as $(v_{\sigma(1)}, v_{\sigma(2)}), (v_{\sigma(3)}, v_{\sigma(4)}), \dots, (v_{\sigma(2t-1)}, v_{\sigma(2t)})$, where $t \in [\lfloor \frac{n}{2} \rfloor]$. The parameter t represents the number of user-pairs and controls the trade-off between the MSE and the time complexity; when $t = \lfloor \frac{n}{2} \rfloor$, the MSE is minimized and the time complexity is maximized. The data collector sends the sampled user-pairs to users (line 2).

Then, we run our wedge algorithm WSLE in Algorithm 7 with each sampled user-pair as

Data: Adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, $\varepsilon \in \mathbb{R}_{\geq 0}$, $\delta \in [0,1]$, $t \in [\lfloor \frac{n}{2} \rfloor]$.
Result: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
`/* Sample disjoint user-pairs */`
1 [d] $\sigma \leftarrow$`RandomPermutation`(n);
2 [d] Send $(v_{\sigma(1)}, v_{\sigma(2)}), \dots, (v_{\sigma(2t-1)}, v_{\sigma(2t)})$ to users;
3 **foreach** $i \in \{1,3,\dots,2t-1\}$ **do**
4     $\hat{f}^{\triangle}_{\sigma(i),\sigma(i+1)}(G) \leftarrow \mathsf{WSLE}(\mathbf{A}, \varepsilon, \delta, (v_{\sigma(i)}, v_{\sigma(i+1)}))$;
5 **end**
`/* Calculate an unbiased estimate */`
6 [d] $\hat{f}^{\triangle}(G) \leftarrow \frac{n(n-1)}{6t} \sum_{i=1,3,\dots,2t-1} \hat{f}^{\triangle}_{\sigma(i),\sigma(i+1)}(G)$;
7 [d] **return** $\hat{f}^{\triangle}(G)$

Algorithm 8: Our triangle counting algorithm $\mathsf{WShuffle}_{\triangle}$. WSLE is shown in Algorithm 7.

input (lines 3-5). Finally, the data collector estimates the triangle count $f^{\triangle}(G)$ as follows:

$$\hat{f}^{\triangle}(G) = \frac{n(n-1)}{6t} \sum_{i=1,3,\dots,2t-1} \hat{f}^{\triangle}_{\sigma(i),\sigma(i+1)}(G) \tag{3.7}$$

(line 6). Note that a single triangle is never counted by more than one user-pair, as the user-pairs never overlap. Later, we prove that $\hat{f}^{\triangle}(G)$ in (3.7) is unbiased.

Theoretical Properties. We prove that $\mathsf{WShuffle}_{\triangle}$ provides DP:

Theorem 15. *$\mathsf{WShuffle}_{\triangle}$ provides (ε, δ)-element DP and $(2\varepsilon, 2\delta)$-edge DP.*

Theorem 15 comes from the fact that WSLE with a user-pair (v_i, v_j) provides (ε, δ)-DP for each element in the *i*-th and *j*-th columns of the adjacency matrix $\mathbf{A}$ and that $\mathsf{WShuffle}_{\triangle}$ samples disjoint user-pairs, i.e., it uses each element of $\mathbf{A}$ at most once.

Note that running WSLE with all $\binom{n}{2}$ user-pairs provides $((n-2)\varepsilon, (n-2)\delta)$-DP, as it uses each element of $\mathbf{A}$ at most $n-2$ times. The privacy budget is very large, even using the advanced composition [73, 116]. We avoid this issue by sampling user-pairs that share no common users.

We also prove that $\mathsf{WShuffle}_{\triangle}$ provides an unbiased estimate:

Theorem 16. *The estimate produced by $\mathsf{WShuffle}_{\triangle}$ satisfies $\mathbb{E}[\hat{f}^{\triangle}(G)] = f^{\triangle}(G)$.*

Next, we analyze the MSE (= variance) of $\mathsf{WShuffle}_{\triangle}$. This analysis is non-trivial because $\mathsf{WShuffle}_{\triangle}$ samples each user-pair *without replacement*. In this case, the sampled user-pairs are not independent. However, we can prove that t estimates in (3.7) are negatively correlated with each other (Lemma 5 in Appendix C.8.5). Thus, the variance of the sum of t estimates in (3.7) is upper bounded by the sum of their variances, each of which is given by Theorem 14. This brings us to the following result:

Theorem 17. *The estimate produced by $\mathsf{WShuffle}_{\triangle}$ provides the following utility guarantee:*

$$\begin{aligned} \mathsf{MSE}(\hat{f}^{\triangle}(G)) &= \mathbb{V}[\hat{f}^{\triangle}(G)] \\ &\leq \frac{n^4}{36t} err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) + \frac{n^3}{36t} d_{max}^3, \end{aligned} \tag{3.8}$$

where $err_{\mathsf{WSLE}}(n, d_{max}, q, q_L)$ is given by (3.5). When ε and δ are constants, $\varepsilon_L = \log(n) + O(1)$, and $t = \lfloor \frac{n}{2} \rfloor$, we have

$$\mathsf{MSE}(\hat{f}^{\triangle}(G)) \leq O(n^3 d_{max}^2). \tag{3.9}$$

The inequality (3.9) follows from (3.6) and (3.8). The first and second terms in (3.8) are caused by Warner's RR and the sampling of disjoint user-pairs, respectively. In other words, the MSE of $\mathsf{WShuffle}_{\triangle}$ can be decomposed into two factors: the RR and user-pair sampling.

For example, assume that $t = \lfloor \frac{n}{2} \rfloor$. When we do not shuffle wedges (i.e., $\varepsilon_L = \varepsilon$), then $err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) = O(n + d_{max}^2)$, and MSE in (3.8) is $O(n^4 + n^3 d_{max}^2)$. When we shuffle wedges, the MSE is $O(n^3 d_{max}^2)$. Thus, when we ignore the factor of d_{max}, our wedge shuffle technique reduces the MSE from $O(n^4)$ to $O(n^3)$ in triangle counting. The factor of n^3 is caused by the RR for local edges. This is intuitive because a large amount of noise is added to the local edges.

Finally, we analyze the time complexity of $\mathsf{WShuffle}_{\triangle}$. The time complexity of running WSLE with all $\binom{n}{2}$ user-pairs is $O(n^3)$, as there are $O(n^2)$ user-pairs in total and WSLE requires the time complexity of $O(n)$. In contrast, the time complexity of $\mathsf{WShuffle}_{\triangle}$ with $t = \lfloor \frac{n}{2} \rfloor$ is $O(n^2)$ because it samples $O(n)$ user-pairs. Thus, $\mathsf{WShuffle}_{\triangle}$ reduces the time complexity from $O(n^3)$ to $O(n^2)$ by user-pair sampling. We can further reduce the time complexity at the cost of increasing the MSE by setting t small, i.e., $t \ll \lfloor \frac{n}{2} \rfloor$.

3.5.4 Variance Reduction

Algorithm. $\mathsf{WShuffle}_{\triangle}$ achieves the MSE of $O(n^3)$ when we ignore the factor of d_{max}. To provide a smaller estimation error, we propose a variance reduction technique that ignores sparse user-pairs. We denote our triangle counting algorithm with the variance reduction technique by $\mathsf{WShuffle}_{\triangle}^*$.

As explained in Section 3.5.3, the factor of n^3 is caused by the RR for local edges. However, most user-pairs v_i and v_j have a very small minimum degree $\min\{d_i, d_j\} \ll d_{max}$, and there is no edge (v_i, v_j) between them in almost all cases. In addition, even if there is an edge (v_i, v_j), the number of triangles involving the sparse user-pair is very small (at most $\min\{d_i, d_j\}$) and can be approximated by 0. By ignoring such sparse user-pairs, we can dramatically reduce the variance of the RR for local edges at the cost of a small bias. This is an intuition behind our variance reduction technique.

Algorithm 9 shows $\mathsf{WShuffle}_{\triangle}^*$. This algorithm detects sparse user-pairs based on the degree information. However, user v_i's degree d_i can leak the information about edges of v_i. Thus, $\mathsf{WShuffle}_{\triangle}^*$ calculates a differentially private estimate of d_i within one round.

Specifically, $\mathsf{WShuffle}_{\triangle}^*$ uses two privacy budgets: $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$. The first budget ε_1 is for privately estimating d_i, whereas the second budget ε_2 is for WSLE. Lines 1 to 5 in Algorithm 9 are the same as those in Algorithm 8, except that Algorithm 9 uses ε_2 to provide (ε_2, δ)-element DP. After these processes, each user v_i adds the Laplacian noise $\mathrm{Lap}(\frac{1}{\varepsilon_1})$ with mean 0 and scale $\frac{1}{\varepsilon_1}$ to her degree d_i and sends the noisy degree $\tilde{d}_i$ $(= d_i + \mathrm{Lap}(\frac{1}{\varepsilon_1}))$ to the data collector (lines 6-8). Because the sensitivity [73] of d_i (the maximum distance of d_i between two neighbor lists that differ in one bit) is 1, adding $\mathrm{Lap}(\frac{1}{\varepsilon_1})$ to d_i provides ε_1-element DP.

Data: Adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, $\varepsilon_1, \varepsilon_2 \in \mathbb{R}_{\geq 0}$, $\delta \in [0,1]$, $t \in [\lfloor \frac{n}{2} \rfloor]$, $c \in \mathbb{R}_{\geq 0}$.
Result: Estimate $\hat{f}^{\triangle}(G)$ of $f^{\triangle}(G)$.
/* Sample disjoint user-pairs */
1 [d] $\sigma \leftarrow$RandomPermutation(n);
2 [d] Send $(v_{\sigma(1)}, v_{\sigma(2)}), \dots, (v_{\sigma(2t-1)}, v_{\sigma(2t)})$ to users;
3 **foreach** $i \in \{1, 3, \dots, 2t-1\}$ **do**
4     $\hat{f}^{\triangle}_{\sigma(i),\sigma(i+1)}(G) \leftarrow \mathsf{WSLE}(\mathbf{A}, \varepsilon_2, \delta, (v_{\sigma(i)}, v_{\sigma(i+1)}))$;
5 **end**
/* Send noisy degrees */
6 **for** $i = 1$ **to** n **do**
7     $[v_i]$ $\tilde{d}_i \leftarrow d_i + \mathrm{Lap}(\frac{1}{\varepsilon_1})$; Send $\tilde{d}_i$ to the data collector;
8 **end**
/* Calculate a variance-reduced estimate */
9 [d] $\tilde{d}_{avg} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \tilde{d}_i$; $d_{th} \leftarrow c\tilde{d}_{avg}$;
10 [d] $D \leftarrow \{i | i = 1, 3, \dots, 2t-1, \min\{\tilde{d}_{\sigma(i)}, \tilde{d}_{\sigma(i+1)}\} > d_{th}\}$;
11 [d] $\hat{f}^{\triangle}(G) \leftarrow \frac{n(n-1)}{6t} \sum_{i \in D} \hat{f}^{\triangle}_{\sigma(i),\sigma(i+1)}(G)$;
12 [d] **return** $\hat{f}^{\triangle}(G)$

Algorithm 9: Our triangle counting algorithm with variance reduction $\mathsf{WShuffle}^{*}_{\triangle}$. WSLE is shown in Algorithm 7.

Then, the data collector estimates the average degree d_{avg} as $\tilde{d}_{avg} = \frac{1}{n}\sum_{i=1}^{n} \tilde{d}_i$ and sets a threshold d_{th} of the minimum degree to $d_{th} = c\tilde{d}_{avg}$, where $c \in \mathbb{R}_{\geq 0}$ is a small positive number, e.g., $c \in [1, 10]$ (line 9). Finally, the data collector estimates $f^{\triangle}(G)$ as

$$\hat{f}^{\triangle}(G) = \frac{n(n-1)}{6t} \sum_{i \in D} \hat{f}^{\triangle}_{\sigma(i),\sigma(i+1)}(G), \tag{3.10}$$

where

$$D = \{i | i = 1, 3, \dots, 2t-1, \min\{\tilde{d}_{\sigma(i)}, \tilde{d}_{\sigma(i+1)}\} > d_{th}\}$$

(lines 10-11). The difference between (3.7) and (3.10) is that (3.10) ignores sparse user-pairs $v_{\sigma(i)}$ and $v_{\sigma(i+1)}$ such that $\min\{\tilde{d}_{\sigma(i)}, \tilde{d}_{\sigma(i+1)}\} \leq d_{th}$. Since $d_{avg} \ll d_{max}$ in practice, $d_{th} \ll d_{max}$ holds for small c.

The parameter c controls the trade-off between the bias and variance of the estimate

$\hat{f}^{\triangle}(G)$. The larger c is, the more user-pairs are ignored. Thus, as c increases, the bias is increased, and the variance is reduced. In practice, a small c not less than 1 results in a small MSE because most real graphs are scale-free networks that have a power-law degree distribution [16]. In the scale-free networks, most users' degrees are smaller than the average degree d_{avg}. For example, in the BA (Barabási-Albert) graph model [16, 96], most users' degrees are $\frac{d_{avg}}{2}$. Thus, if we set $c \in [1, 10]$, for example, then most user-pairs are ignored (i.e., $|D| \ll t$), which leads to a significant reduction of the variance at the cost of a small bias.

Recall that the parameter t in $\mathsf{WShuffle}_{\triangle}^{*}$ controls the trade-off between the MSE and the time complexity. Although $\mathsf{WShuffle}_{\triangle}^{*}$ always samples t disjoint user-pairs, we can modify $\mathsf{WShuffle}_{\triangle}^{*}$ so that it stops sampling user-pairs right after the estimate $\hat{f}^{\triangle}(G)$ in (3.10) is converged. We can also sample dense user-pairs (v_i, v_j) with large noisy degrees $\tilde{d}_i$ and $\tilde{d}_j$ at the beginning (e.g., by sorting users in descending order of noisy degrees) to improve the MSE for small t. Evaluating such improved algorithms is left for future work.

Theoretical Properties. As with $\mathsf{WShuffle}_{\triangle}$, $\mathsf{WShuffle}_{\triangle}^{*}$ provides the following privacy guarantee:

Theorem 18. *$\mathsf{WShuffle}_{\triangle}^{*}$ provides $(\varepsilon_1 + \varepsilon_2, \delta)$-element DP and $(2(\varepsilon_1 + \varepsilon_2), 2\delta)$-edge DP.*

Next, we analyze the bias of $\mathsf{WShuffle}_{\triangle}^{*}$. Here, we assume most users have a small degree using parameters $\lambda \in \mathbb{R}_{\geq 0}$ and $\alpha \in [0, 1)$:

Theorem 19. *Suppose that in G, there exist $\lambda \in \mathbb{R}_{\geq 0}$ and $\alpha \in [0, 1)$ such that at most n^{α} users have a degree larger than λd_{avg}. Suppose $\mathsf{WShuffle}_{\triangle}^{*}$ is run with $c \geq \lambda$. Then, the estimator produced by $\mathsf{WShuffle}_{\triangle}^{*}$ provides the following bias guarantee:*

$$Bias[\hat{f}^{\triangle}(G)] = |\mathbb{E}[\hat{f}^{\triangle}(G)] - f^{\triangle}(G)| \leq \frac{nc^2 d_{avg}^2}{3} + \frac{4n^{\alpha}}{3\varepsilon_1^2}. \tag{3.11}$$

The values of λ and α depend on the original graph G. In the scale-free networks, α is small for a moderate value of λ. For example, in the BA graph with $n = 107614$ and $d_{avg} = 200$

used in Appendix C.4, $\alpha = 0.5$, 0.6, 0.8, and 0.9 when $\lambda = 10.1$, 5.4, 1.6, and 0.9, respectively. When c and ε_1 are constants, the bias can be expressed as $O(nd_{avg}^2)$.

Finally, we show the variance of $\mathsf{WShuffle}^*_\triangle$. This result assumes that c is bigger $(= \frac{(1-\alpha)\log n}{\varepsilon_1 d_{avg}})$ than λ. We assume this because otherwise, many sparse users (with $d_i \leq \lambda d_{avg}$) have a noisy degree $\tilde{d}_i \geq c\tilde{d}_{avg}$, causing the set D to be noisy. In practice, the gap between c and λ is small because $\log n$ is much smaller than d_{avg}.

Theorem 20. *Suppose that in G, there exist $\lambda \in \mathbb{R}_{\geq 0}$ and $\alpha \in [0,1)$ such that at most n^α users have a degree larger than λd_{avg}. Suppose $\mathsf{WShuffle}^*_\triangle$ is run with $c \geq \lambda + \frac{(1-\alpha)\log n}{\varepsilon_1 d_{avg}}$. Then, the estimator produced by $\mathsf{WShuffle}^*_\triangle$ provides the following variance guarantee:*

$$\mathbb{V}[\hat{f}^\triangle(G)] \leq \\ \frac{n^2 d_{max}^4}{9} + \frac{2n^{2+2\alpha}}{9t} err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) + \frac{n^{2+\alpha} d_{max}^3}{36t}. \tag{3.12}$$

When ε_1, ε_2, and δ are constants, $\varepsilon_L = \log n + O(1)$, and $t = \lfloor \frac{n}{2} \rfloor$,

$$\mathbb{V}[\hat{f}^\triangle(G)] \leq O(n^2 d_{max}^4 + n^{1+2\alpha} d_{max}^2). \tag{3.13}$$

The first[2], second, and third terms in (3.12) are caused by the randomness in the choice of D, the RR, and user-pair sampling, respectively. By (3.13), our variance reduction technique reduces the variance from $O(n^3)$ to $O(n^\gamma)$ where $\gamma \in [2,3)$ when we ignore the factor of d_{max}. Because the MSE is the sum of the squared bias and the variance, it is also $O(n^\gamma)$.

The value of γ in our bound $O(n^\gamma)$ depends on the parameter c in $\mathsf{WShuffle}^*_\triangle$. For example, in the BA graph ($n = 107614$, $d_{avg} = 200$), $\gamma = 2$, 2.2, 2.6, and 2.8 ($\alpha = 0.5$, 0.6, 0.8, and 0.9) when $c = 10.4$, 5.6, 1.7, and 1.0, respectively, and $\varepsilon_1 = 0.1$. Thus, the variance decreases with increase in c. However, by (3.11), a larger c results in a larger bias. In our

[2]The first term in (3.12) is actually $\frac{(\sum_{i=1}^n d_i^2)^2}{9}$ and is much smaller than $\frac{n^2 d_{max}^4}{9}$. We express it as $O(n^2 d_{max}^4)$ in (3.13) for simplicity. See Appendix C.8.8 for details.

Table 3.2. Performance guarantees of one-round triangle counting algorithms providing edge DP. $\alpha \in [0,1)$. See also footnote 2 for the variance of $\mathsf{WShuffle}^*_\triangle$.

Algorithm	Model	Variance	Bias	Time
$\mathsf{WShuffle}^*_\triangle$	shuffle	$O(n^2 d_{max}^4 + n^{1+2\alpha} d_{max}^2)$	$O(nd_{avg}^2)$	$O(n^2)$
$\mathsf{WShuffle}_\triangle$	shuffle	$O(n^3 d_{max}^2)$	0	$O(n^2)$
$\mathsf{WLocal}_\triangle$	local	$O(n^4 + n^3 d_{max}^2)$	0	$O(n^2)$
$\mathsf{ARR}_\triangle$ [104]	local	$O(n^6)$	0	$O(n^2)$
$\mathsf{RR}_\triangle$ [102]	local	$O(n^4)$	0	$O(n^3)$

experiments, we show that $\mathsf{WShuffle}^*_\triangle$ provides a small estimation error when $c = 1$ to 4. When $c = 1$, $\mathsf{WShuffle}^*_\triangle$ empirically works well despite a large γ because most users' degrees are smaller than d_{avg} in practice, as explained above. This indicates that our upper bound in (3.13) might not be tight when c is around 1. Improving the bound is left for future work.

3.5.5 Summary

Table 3.2 summarizes the performance guarantees of one-round triangle algorithms providing edge DP. Here, we consider a variant of $\mathsf{WShuffle}_\triangle$ that does not shuffle wedges (i.e., $\varepsilon_L = \varepsilon$) as a one-round local algorithm. We call this variant $\mathsf{WLocal}_\triangle$ (Wedge Local). We also show the variance of $\mathsf{ARR}_\triangle$ [104] and $\mathsf{RR}_\triangle$ [102]. The time complexity of $\mathsf{RR}_\triangle$ is $O(n^3)$[3], and that of $\mathsf{ARR}_\triangle$ is $O(n^2)$ when we set the sampling probability $p_0 \in [0,1]$ of the ARR to $p_0 = O(n^{-1/3})$. We prove the variance of $\mathsf{ARR}_\triangle$ in this case and $\mathsf{RR}_\triangle$ in Appendix C.7. We do not show the other one-round local algorithms [218, 219] in Table 3.2 for two reasons: (i) they have the time complexity of $O(n^3)$ and suffer from a larger estimation error than $\mathsf{RR}_\triangle$ [104]; (ii) their upper-bounds on the variance and bias are unclear.

Table 3.2 shows that our $\mathsf{WShuffle}^*_\triangle$ dramatically outperforms the three local algorithms – when we ignore d_{max}, the MSE of $\mathsf{WShuffle}^*_\triangle$ is $O(n^\gamma)$ where $\gamma \in [2,3)$, whereas that of the local algorithms is $O(n^4)$ or $O(n^6)$. We also show this through experiments.

[3]Technically speaking, the algorithms of $\mathsf{RR}_\triangle$ and the one-round local algorithms in [218, 219] involve counting the number of triangles in a dense graph. This can be done in time $O(n^\omega)$, where $\omega \in [2,3)$ and $O(n^\omega)$ is the time required for matrix multiplication. However, these algorithms are of theoretical interest, and they do not outperform naive matrix multiplication except for very large matrices [7]. Thus, we assume implementations that use naive matrix multiplication in $O(n^3)$ time.

Note that both $\mathsf{ARR}_\triangle$ and $\mathsf{RR}_\triangle$ provide pure DP ($\delta = 0$), whereas our shuffle algorithms provide approximate DP ($\delta > 0$). However, it would not make a noticeable difference, as δ is sufficiently small (e.g., $\delta = 10^{-8} \ll \frac{1}{n}$ in our experiments).

Comparison with the Central Model. Finally, we note that our $\mathsf{WShuffle}^*_\triangle$ is worse than algorithms in the central model in terms of the estimation error.

Specifically, Imola *et al.* [102] consider a central algorithm that adds the Laplacian noise $\mathrm{Lap}(\frac{d_{max}}{\varepsilon})$ to the true count $f^\triangle(G)$ and outputs $f^\triangle(G) + \mathrm{Lap}(\frac{d_{max}}{\varepsilon})$[4]. This central algorithm provides $(\varepsilon, 0)$-edge DP. In addition, the estimate is unbiased, and the variance is $\frac{2d_{max}^2}{\varepsilon^2} = O(d_{max}^2)$. Thus, the central algorithm provides a much smaller MSE (= variance) than $\mathsf{WShuffle}^*_\triangle$.

However, our $\mathsf{WShuffle}^*_\triangle$ is preferable to central algorithms in terms of the trust model – the central model assumes that a single party accesses personal data of all users and therefore has a risk that the entire graph is leaked from the party. $\mathsf{WShuffle}^*_\triangle$ can also be applied to decentralized social networks, as described in Section 3.1.

3.6 4-Cycle Counting Based on Wedge Shuffling

Next, we propose a one-round 4-cycle counting algorithm in the shuffle model. Section 3.6.1 explains its overview. Section 3.6.2 proposes our 4-cycle counting algorithm and shows its theoretical properties. Section 3.6.3 summarizes the performance guarantees of our 4-cycle algorithms.

3.6.1 Overview

We apply our wedge shuffling technique to 4-cycle counting with two additional techniques: *(i) bias correction* and *(ii) sampling disjoint user-pairs*. Below, we briefly explain each of them.

Bias Correction. As with triangles, we begin with the problem of counting 4-cycles involving

[4]Here, we assume that d_{max} is publicly available; e.g., $d_{max} = 5000$ in Facebook [210]. When d_{max} is not public, the algorithm in [102] outputs $f(G) + \mathrm{Lap}(\frac{\tilde{d}_{max}}{\varepsilon})$, where $\tilde{d}_{max} = \max_{i=1,\dots,n} \tilde{d}_i$, i.e., the maximum of noisy degrees.

specific users v_i and v_j. We can leverage the noisy wedges output by our wedge shuffle algorithm WS to estimate such a 4-cycle count. Specifically, let $f_{i,j}^{\square} : \mathscr{G} \rightarrow \mathbb{Z}_{\geq 0}$ be a function that, given $G \in \mathscr{G}$, outputs the number $f_{i,j}^{\square}(G)$ of 4-cycles for which users v_i and v_j are opposite nodes, i.e. the number of unordered pairs (k,k') such that $v_i - v_k - v_j - v_{k'} - v_i$ is a path in G. Each pair (k,k') satisfies the above requirement if and only if $v_i - v_k - v_j$ and $v_i - v_{k'} - v_j$ are wedges in G. Thus, we have $f_{i,j}^{\square}(G) = \binom{f_{i,j}^{\wedge}}{2}$, where $f_{i,j}^{\wedge}$ is the number of wedges between v_i and v_j. Based on this, we calculate an unbiased estimate $\hat{f}_{i,j}^{\wedge}$ of the wedge count using WS. Then, we calculate an estimate of the 4-cycle count as $\binom{\hat{f}_{i,j}^{\wedge}}{2}$. Here, it should be noted that the estimate $\binom{\hat{f}_{i,j}^{\wedge}}{2}$ is *biased*, as proved later. Therefore, we perform bias correction – we subtract a positive value from the estimate to obtain an unbiased estimate $\hat{f}_{i,j}^{\square}(G)$ of the 4-cycle count.

Note that unlike WSLE, no edge between (v_i, v_j) needs to be sent. In addition, thanks to the privacy amplification by shuffling, all wedges can be sent with small noise.

Sampling Disjoint User-Pairs. Having an estimate $\hat{f}_{i,j}^{\square}(G)$, we turn our attention to estimating 4-cycle count $f^{\square}(G)$ in the entire graph G. As with triangles, a naive solution using estimates $\hat{f}_{i,j}^{\square}(G)$ for all $\binom{n}{2}$ user-pairs (v_i, v_j) results in very large ε and δ. To avoid this, we sample disjoint user-pairs and obtain an unbiased estimate of $f^{\square}(G)$ from them.

3.6.2 4-Cycle Counting

Algorithm. Algorithm 10 shows our 4-cycle counting algorithm. We denote it by $\mathsf{WShuffle}_{\square}$. First, we set a local privacy budget ε_L from n, ε, and δ in the same way as WSLE (line 1). Then, we sample t disjoint pairs of users using the permutation σ (lines 3-4). Each pair is given by $(v_{\sigma(i),\sigma(i+1)})$ for $i \in \{1,3,\ldots,2t-1\}$.

For each $i \in \{1,3,\ldots,2t-1\}$, we compute an unbiased estimate $\hat{f}_{\sigma(i),\sigma(i+1)}^{\square}(G)$ of the 4-cycle count involving $v_{\sigma(i)}$ and $v_{\sigma(i+1)}$ (lines 5-9). To do this, we call WS on $(v_{\sigma(i)} v_{\sigma(i+1)})$ to obtain an unbiased estimate $\hat{f}_{\sigma(i),\sigma(i+1)}^{\wedge}(G)$ of the wedge count (lines 6-7). We calculate an

Data: Adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$, $\varepsilon \in \mathbb{R}_{\geq 0}$, $\delta \in [0,1]$, $t \in [\lfloor \frac{n}{2} \rfloor]$.
Result: Estimate $\hat{f}^{\square}(G)$ of $f^{\square}(G)$.

```
1  ε_L ← LocalPrivacyBudget(n, ε, δ);
2  [d] q_L ← 1/(e^{ε_L}+1);
   /* Sample disjoint user-pairs */
3  [d] σ ← RandomPermutation(n);
4  [d] Send (v_σ(1), v_σ(2)), ..., (v_σ(2t−1), v_σ(2t)) to users;
5  foreach i ∈ {1, 3, ..., 2t − 1} do
6      {y_{π_i(k)} | k ∈ I_{−(σ(i),σ(i+1))}} ← WS(A, ε_L, (v_σ(i), v_σ(i+1)));
7      [d] f̂^∧_{σ(i),σ(i+1)}(G) ← Σ_{k∈I_{−(σ(i),σ(i+1))}} (y_k − q_L)/(1 − 2q_L);
8      [d] f̂^□_{σ(i),σ(i+1)}(G) ← f̂^∧_{σ(i),σ(i+1)}(G)(f̂^∧_{σ(i),σ(i+1)}(G) − 1)/2 − (n−2)/2 · q_L(1−q_L)/(1−2q_L)^2;
9  end
   /* Calculate an unbiased estimate */
10 [d] f̂^□(G) ← n(n−1)/(4t) Σ_{i=1,3,...,2t−1} f̂^□_{σ(i),σ(i+1)}(G);
11 [d] return f̂^□(G)
```

Algorithm 10: Our 4-cycle counting algorithm $\mathsf{WShuffle}_{\square}$. WS is shown in Algorithm 6.

estimate $\hat{f}^{\wedge}_{i,j}(G)$ of the number $f^{\wedge}_{i,j}(G)$ of wedges between v_i and v_j in G as follows:

$$\hat{f}^{\wedge}_{i,j}(G) = \textstyle\sum_{k \in I_{-(i,j)}} \frac{y_k - q_L}{1 - 2q_L}. \tag{3.14}$$

Later, we will prove that $\hat{f}^{\wedge}_{i,j}(G)$ is an unbiased estimator. As with (3.4), this estimate involves the sum over the set $\{y_{\pi(k)}\}$ and does not require knowing the permutation π produced by the shuffler. Then, we obtain an unbiased estimator of $f^{\square}_{\sigma(i),\sigma(i+1)}$ as follows:

$$\hat{f}^{\square}_{i,j}(G) = \frac{\hat{f}^{\wedge}_{i,j}(G)(\hat{f}^{\wedge}_{i,j}(G)-1)}{2} - \frac{n-2}{2} \frac{q_L(1-q_L)}{(1-2q_L)^2} \tag{3.15}$$

(line 8). Note that there is a quadratic relationship between $f^{\square}_{i,j}(G)$ and $f^{\wedge}_{i,j}(G)$, i.e., $f^{\square}_{i,j}(G) = \binom{f^{\wedge}_{i,j}(G)}{2}$. Thus, even though $\hat{f}^{\wedge}_{i,j}(G)$ is unbiased, we must subtract a term from $\binom{\hat{f}^{\wedge}_{i,j}(G)}{2}$ (i.e., bias correction) to obtain an unbiased estimator $\hat{f}^{\square}_{i,j}(G)$. This forms the righthand side of (3.15) and ensures that $\hat{f}^{\square}_{i,j}(G)$ is unbiased.

Finally, we sum and scale $\hat{f}^{\square}_{\sigma(i),\sigma(i+1)}(G)$ for each i to obtain an estimate $\hat{f}^{\square}(G)$ of the

4-cycle count $f^{\square}(G)$ in the entire graph G:

$$\hat{f}^{\square}(G) = \tfrac{n(n-1)}{4t} \textstyle\sum_{i=1,3,\dots,2t-1} \hat{f}^{\square}_{\sigma(i),\sigma(i+1)}(G) \tag{3.16}$$

(line 10). Note that it is possible that a single 4-cycle is counted twice; e.g., a 4-cycle v_i-v_j-v_k-v_l-v_i is possibly counted by (v_i, v_k) and (v_j, v_l) if these user-pairs are selected. However, this is not an issue, because all 4-cycles are equally likely to be counted zero times, once, or twice. We also prove later that $\hat{f}^{\square}(G)$ in (3.16) is an unbiased estimate of $f^{\square}(G)$.

Theoretical Properties. First, $\mathsf{WShuffle}_{\square}$ guarantees DP:

Theorem 21. *$\mathsf{WShuffle}_{\square}$ provides (ε, δ)-element DP and $(2\varepsilon, 2\delta)$-edge DP.*

In addition, thanks to the design of (3.15), we can show that $\mathsf{WShuffle}_{\square}$ produces an unbiased estimate of $f^{\square}(G)$:

Theorem 22. *The estimate produced by $\mathsf{WShuffle}_{\square}$ satisfies $\mathbb{E}[\hat{f}^{\square}(G)] = f^{\square}(G)$.*

Finally, we show the MSE (= variance) of $f^{\square}(G)$:

Theorem 23. *The estimate produced by $\mathsf{WShuffle}_{\square}$ satisfies*

$$\begin{aligned} \mathsf{MSE}(\hat{f}^{\square}(G)) &= \mathbb{V}[\hat{f}^{\square}(G)] \\ &\leq \frac{9n^5 q_L (d_{max} + nq_L)^2}{16t(1-2q_L)^4} + \frac{n^3 d_{max}^6}{64t}. \end{aligned} \tag{3.17}$$

When ε and δ are constants, $\varepsilon_L = \log n + O(1)$, and $t = \lfloor \frac{n}{2} \rfloor$, we have

$$\mathsf{MSE}(\hat{f}^{\square}(G)) = \mathbb{V}[\hat{f}^{\square}(G)] = O\left(n^3 d_{max}^2 + n^2 d_{max}^6\right). \tag{3.18}$$

The first and second terms in (3.17) are caused by the RR and the sampling of disjoint user-pairs, respectively.

Table 3.3. Performance guarantees of one-round 4-cycle counting algorithms providing edge DP.

Algorithm	Model	Variance	Bias	Time
$\mathsf{WShuffle}_{\square}$	shuffle	$O(n^3 d_{max}^2 + n^2 d_{max}^6)$	0	$O(n^2)$
$\mathsf{WLocal}_{\square}$	local	$O(n^6 + n^2 d_{max}^6)$	0	$O(n^2)$

3.6.3 Summary

Table 3.3 summarizes the performance guarantees of the 4-cycle counting algorithms. As a one-round local algorithm, we consider a local model version of $\mathsf{WShuffle}_{\square}$ that does not shuffle wedges (i.e., $\varepsilon_L = \varepsilon$). We denote it by $\mathsf{WLocal}_{\square}$. To our knowledge, $\mathsf{WLocal}_{\square}$ is the first local 4-cycle counting algorithm.

By (3.17), when $t = \lfloor \frac{n}{2} \rfloor$, the MSE of $\mathsf{WLocal}_{\square}$ can be expressed as $O(n^6 + n^2 d_{max}^6)$. Thus, our wedge shuffle technique dramatically reduces the MSE from $O(n^6 + n^2 d_{max}^6)$ to $O(n^3 d_{max}^2 + n^2 d_{max}^6)$. Note that the square of the true count $f^{\square}(G)$ is $O(n^2 d_{max}^6)$. This indicates that our $\mathsf{WShuffle}_{\square}$ may not work well in an extremely sparse graph where $d_{max} < n^{\frac{1}{4}}$. However, $d_{max} \gg n^{\frac{1}{4}}$ holds in most social graphs; e.g., the maximum number d_{max} of friends is much larger than 100 when $n = 10^8$. In this case, $\mathsf{WShuffle}_{\square}$ can accurately estimate the 4-cycle count, as shown in our experiments.

Comparison with the Central Model. As with triangles, our $\mathsf{WShuffle}_{\square}$ is worse than algorithms in the central model in terms of the estimation error.

Specifically, analogously to the central algorithm for triangles [102], we can consider a central algorithm that outputs $f^{\square}(G) + \mathrm{Lap}(\frac{d_{max}^2}{\varepsilon})$. This algorithm provides $(\varepsilon, 0)$-edge DP and the variance of $\frac{2d_{max}^4}{\varepsilon^2} = O(d_{max}^4)$. Because d_{max} is much smaller than n, this central algorithm provides a much smaller MSE ($=$ variance) than $\mathsf{WShuffle}_{\square}$. This indicates that there is a trade-off between the trust model and the estimation error.

3.7 Experimental Evaluation

Based on the performance guarantees summarized in Tables 3.2 and 3.3, we pose the following research questions:

RQ1. How much do our entire algorithms ($\mathsf{WShuffle}^*_\triangle$ and $\mathsf{WShuffle}_\square$) outperform the local algorithms?

RQ2. For triangles, how much does our variance reduction technique decrease the relative error?

RQ3. How small relative errors do our entire algorithms achieve with a small privacy budget?

We designed experiments to answer these questions.

3.7.1 Experimental Set-up

We used the following two real graph datasets:

- **Gplus**: The first dataset is the Google+ dataset [142] denoted by Gplus. This dataset includes a social graph $G = (V, E)$ with $n = 107614$ users and 12238285 edges, where an edge $(v_i, v_j) \in E$ represents that a user v_i follows or is followed by v_j. The average and maximum degrees are $d_{avg} = 227.4$ and $d_{max} = 20127$, respectively.

- **IMDB**: The second dataset is the IMDB (Internet Movie Database) [1] denoted by IMDB. This dataset includes a bipartite graph between 896308 actors and 428440 movies. From this, we extracted a graph $G = (V, E)$ with $n = 896308$ actors and 57064358 edges, where an edge represents that two actors have played in the same movie. The average and maximum degrees are $d_{avg} = 127.3$ and $d_{max} = 15451$, respectively; i.e., IMDB is more sparse than Gplus.

In Appendix C.4, we also evaluate our algorithms using the Barabási-Albert graphs [16, 96], which have a power-law degree distribution. Moreover, in Appendix C.5, we evaluate our 4-cycle algorithms using bipartite graphs generated from Gplus and IMDB.

For triangle counting, we evaluated the following four one-round algorithms: $\mathsf{WShuffle}^*_\triangle$, $\mathsf{WShuffle}_\triangle$, $\mathsf{WLocal}_\triangle$, and $\mathsf{ARR}_\triangle$ [104]. We did not evaluate $\mathsf{RR}_\triangle$ [102], because it was too inefficient – it was reported in [102] that when $n = 10^6$, $\mathsf{RR}_\triangle$ would require over 30 years even on a supercomputer. The same applies to the one-round local algorithms in [218, 219] with the same time complexity ($= O(n^3)$).

For 4-cycle counting, we compared $\mathsf{WShuffle}_\square$ with $\mathsf{WLocal}_\square$. Because $\mathsf{WLocal}_\square$ is the first local 4-cycle counting algorithm (to our knowledge), we did not evaluate other algorithms.

In our shuffle algorithms $\mathsf{WShuffle}^*_\triangle$, $\mathsf{WShuffle}_\triangle$, and $\mathsf{WShuffle}_\square$, we set $\delta = 10^{-8}$ ($\ll \frac{1}{n}$) and $t = \frac{n}{2}$. We used the numerical upper bound in [88] for calculating ε in the shuffle model. In $\mathsf{WShuffle}^*_\triangle$, we set $c \in [0.1, 4]$ and divided the total privacy budget ε as $\varepsilon_1 = \frac{\varepsilon}{10}$ and $\varepsilon_2 = \frac{9\varepsilon}{10}$. Here, we assigned a small budget to ε_1 because a degree d_i has a very small sensitivity ($= 1$) and $\mathrm{Lap}(\frac{1}{\varepsilon_1})$ is very small. In $\mathsf{ARR}_\triangle$, we set the sampling probability p_0 to $p_0 = n^{-1/3}$ or $0.1n^{-1/3}$ so that the time complexity is $O(n^2)$.

We ran each algorithm 20 times and evaluated the average relative error over the 20 runs. In Appendix C.6, we show that the standard error of the average relative error is small.

3.7.2 Experimental Results

Relative Error vs. ε. We first evaluated the relation between the relative error and ε in element DP or edge LDP, i.e., 2ε in edge DP. We also measured the time to estimate the triangle/4-cycle count from the adjacency matrix $\mathbf{A}$ using a supercomputer [5] with two Intel Xeon Gold 6148 processors (2.40 GHz, 20 Cores) and 412 GB main memory.

Figure 3.6 shows the relative error ($c = 1$). Here, we show the performance of $\mathsf{WShuffle}_\triangle$ when we do not add the Laplacian noise (denoted by $\mathsf{WShuffle}_\triangle$ (w/o Lap)). In IMDB, we do not show $\mathsf{ARR}_\triangle$ with $p_0 = n^{-1/3}$, because it takes too much time (longer than one day). Table 3.4 highlights the relative error when $\varepsilon = 0.5$ or 1. It also shows the running time of counting triangles or 4-cycles when $\varepsilon = 1$ (we verified that the running time had little dependence on ε).

Figure 3.6 and Table 3.4 show that our shuffle algorithms dramatically improve the local

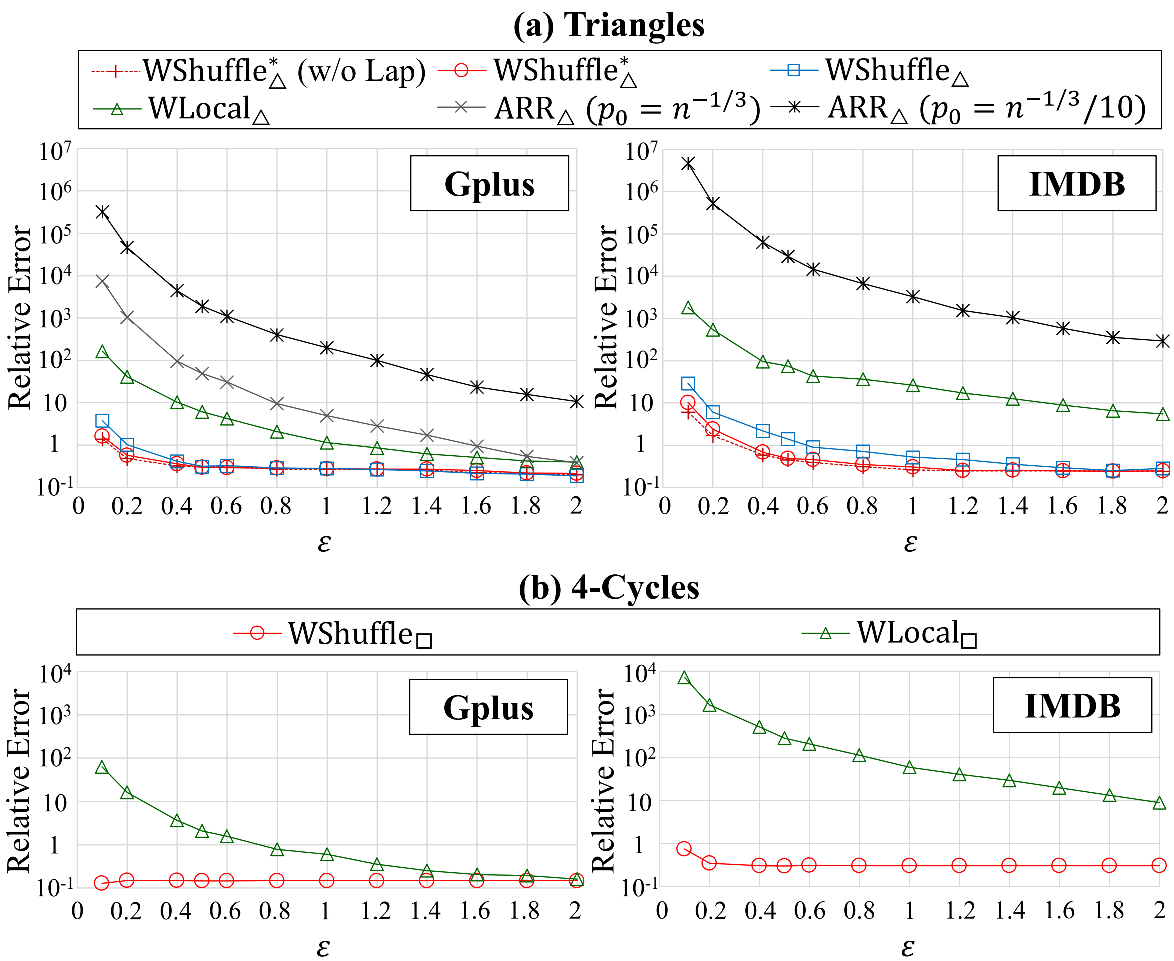


Figure 3.6. Relative error vs. ε ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$). p_0 is the sampling probability in the ARR.

algorithms. In triangle counting, WShuffle$^*_\triangle$ outperforms WLocal$_\triangle$ by one or two orders of magnitude and ARR$_\triangle$ by even more[5]. WShuffle$^*_\triangle$ also requires less running time than ARR$_\triangle$ with $p_0 = n^{-1/3}$. Although the running time of ARR$_\triangle$ can be improved by using a smaller p_0, it results in a higher relative error. In 4-cycle counting, WShuffle$_\square$ significantly outperforms WLocal$_\square$. The difference between our shuffle algorithms and the local algorithms is larger in IMDB because it is more sparse; i.e., the difference between d_{max} and n is larger in IMDB. This is consistent with our theoretical results in Tables 3.2 and 3.3.

[5]Note that ARR$_\triangle$ uses only the lower-triangular part of the adjacency matrix $\mathbf{A}$ and therefore provides ε-edge DP (rather than 2ε-edge DP); i.e., it does not suffer from the doubling issue explained in Section 3.3.2. However, Figure 3.6 shows that WShuffle$^*_\triangle$ significantly outperforms ARR$_\triangle$ even if we double ε for only WShuffle$^*_\triangle$.

Table 3.4. Relative error (RE) when $\varepsilon = 0.5$ or 1 and computational time ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$). The lowest relative error is highlighted in bold.

(a) Gplus

	RE ($\varepsilon = 0.5$)	RE ($\varepsilon = 1$)	Time (sec)
$\mathsf{WShuffle}^*_\triangle$	$\mathbf{2.98 \times 10^{-1}}$	$\mathbf{2.77 \times 10^{-1}}$	3.60×10^{1}
$\mathsf{WShuffle}_\triangle$	3.12×10^{-1}	2.79×10^{-1}	3.62×10^{1}
$\mathsf{WLocal}_\triangle$	6.10×10^{0}	1.14×10^{0}	5.83×10^{1}
$\mathsf{ARR}_\triangle$ ($p_0 = n^{-1/3}$)	4.90×10^{1}	4.93×10^{0}	7.15×10^{2}
$\mathsf{ARR}_\triangle$ ($p_0 = 0.1n^{-1/3}$)	1.88×10^{3}	1.97×10^{2}	3.48×10^{1}
$\mathsf{WShuffle}_\square$	$\mathbf{1.45 \times 10^{-1}}$	$\mathbf{1.47 \times 10^{-1}}$	3.47×10^{1}
$\mathsf{WLocal}_\square$	2.08×10^{0}	5.96×10^{-1}	5.70×10^{1}

(b) IMDB

	RE ($\varepsilon = 0.5$)	RE ($\varepsilon = 1$)	Time (sec)
$\mathsf{WShuffle}^*_\triangle$	$\mathbf{4.88 \times 10^{-1}}$	$\mathbf{3.08 \times 10^{-1}}$	2.39×10^{3}
$\mathsf{WShuffle}_\triangle$	1.41×10^{0}	5.22×10^{-1}	2.40×10^{3}
$\mathsf{WLocal}_\triangle$	7.46×10^{1}	2.63×10^{1}	3.96×10^{3}
$\mathsf{ARR}_\triangle$ ($p_0 = 0.1n^{-1/3}$)	2.98×10^{4}	3.27×10^{3}	2.81×10^{3}
$\mathsf{WShuffle}_\square$	$\mathbf{3.03 \times 10^{-1}}$	$\mathbf{3.08 \times 10^{-1}}$	2.29×10^{3}
$\mathsf{WLocal}_\square$	2.82×10^{2}	5.91×10^{1}	3.91×10^{3}

Figure 3.6 and Table 3.4 also show that $\mathsf{WShuffle}^*_\triangle$ outperforms $\mathsf{WShuffle}_\triangle$, especially when ε is small. This is because the variance is large when ε is small. In addition, $\mathsf{WShuffle}^*_\triangle$ significantly outperforms $\mathsf{WShuffle}_\triangle$ in IMDB because $\mathsf{WShuffle}^*_\triangle$ significantly reduces the variance when $d_{max} \ll n$, as shown in Table 3.2. In other words, this is also consistent with our theoretical results. For example, when $\varepsilon = 0.5$, our variance reduction technique reduces the relative error from 1.41 to 0.488 (about one-third) in IMDB.

Furthermore, Figure 3.6 shows that the relative error of $\mathsf{WShuffle}^*_\triangle$ is hardly changed by adding the Laplacian noise. This is because the sensitivity of each user's degree d_i is very small $(= 1)$. In this case, the Laplacian noise is also very small.

Our $\mathsf{WShuffle}^*_\triangle$ achieves a relative error of 0.3 ($\ll 1$) when the privacy budget is $\varepsilon = 0.5$ or 1 in element DP ($2\varepsilon = 1$ or 2 in edge DP). $\mathsf{WShuffle}_\square$ achieve a relative error of 0.15 to 0.3 with a smaller privacy budget (e.g., $\varepsilon = 0.2$) because it does not send local edges – the error of $\mathsf{WShuffle}_\square$ is mainly caused by user-pair sampling that is independent of ε.

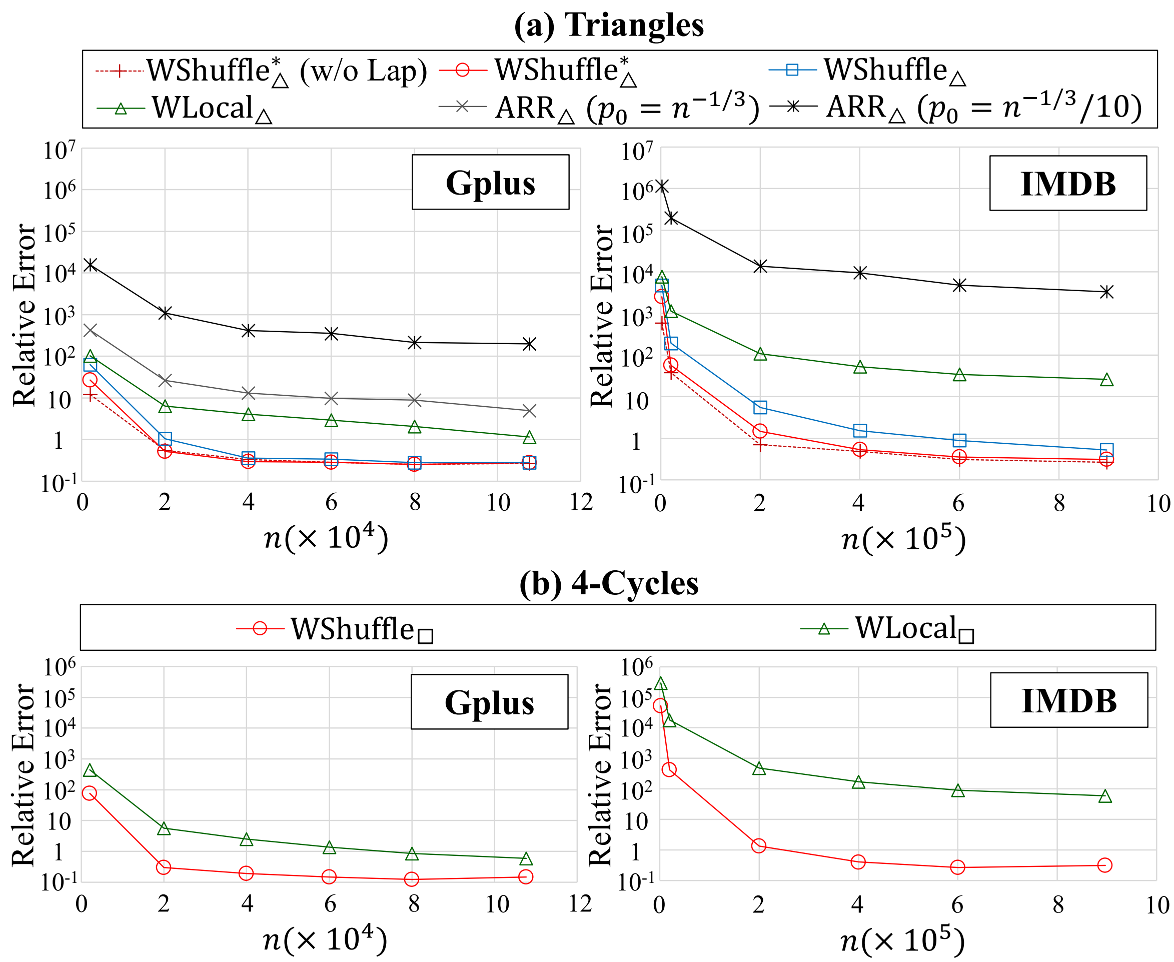


Figure 3.7. Relative error vs. n ($\varepsilon = 1$, $c = 1$).

In summary, our $\mathsf{WShuffle}^*_\triangle$ and $\mathsf{WShuffle}_\square$ significantly outperform the local algorithms and achieve a relative error much smaller than 1 with a reasonable privacy budget, i.e., $\varepsilon \leq 1$.

Relative Error vs. n. Next, we evaluated the relation between the relative error and n. Specifically, we randomly selected n users from all users and extracted a graph with n users. Then we set $\varepsilon = 1$ and changed n to various values starting from 2000.

Figure 3.7 shows the results ($c = 1$). When $n = 2000$, $\mathsf{WShuffle}_\triangle$ and $\mathsf{WShuffle}_\square$ provide relative errors close to $\mathsf{WLocal}_\triangle$ and $\mathsf{WLocal}_\square$, respectively. This is because the privacy amplification effect is limited when n is small. For example, when $n = 2000$ and $\varepsilon = 1$, the numerical bound is $\varepsilon_L = 1.88$. The value of ε_L increases with increase in n; e.g., when $n = 107614$

and 896308, the numerical bound is $\varepsilon_L = 5.86$ and 7.98, respectively. This explains the reason that our shuffle algorithms significantly outperform the local algorithms when n is large in Figure 3.7.

Parameter c in $\mathsf{WShuffle}^*_\triangle$. Finally, we evaluated our $\mathsf{WShuffle}^*_\triangle$ while changing the parameter c that controls the bias and variance. Recall that as c increases, the bias is increased, and the variance is reduced. We set $\varepsilon = 0.1$ or 1 and changed c from 0.1 to 4.

Figure 3.8 shows the results. Here, we also show the relative error of $\mathsf{WShuffle}_\triangle$. We observe that the optimal c is different for $\varepsilon = 0.1$ and $\varepsilon = 1$. The optimal c is around 3 to 4 for $\varepsilon = 0.1$, whereas the optimal c is around 0.5 to 1 for $\varepsilon = 1$. This is because the variance of $\mathsf{WShuffle}_\triangle$ is large (resp. small) when ε is small (resp. large). For a small ε, a large c is effective in significantly reducing the variance. For a large ε, a small c is effective in keeping a small bias.

We also observe that $\mathsf{WShuffle}^*_\triangle$ is always better than (or almost the same as) $\mathsf{WShuffle}_\triangle$ when $c = 1$ or 2. This is because most users' degrees are smaller than the average degree d_{avg}, as described in Section 3.5.4. When $c = 1$ or 2, most user-pairs are ignored. Therefore, we can significantly reduce the variance at the cost of a small bias.

Summary. In summary, our answers to the three questions at the beginning of Section 3.7 are as follows. RQ1: Our $\mathsf{WShuffle}^*_\triangle$ and $\mathsf{WShuffle}_\square$ outperform the one-round local algorithms by one or two orders of magnitude (or even more). RQ2: Our variance reduction technique significantly reduces the relative error (e.g., by about one-third) for a small ε in a sparse dataset. RQ3: $\mathsf{WShuffle}^*_\triangle$ achieves a relative error of 0.3 ($\ll 1$) when $\varepsilon = 0.5$ or 1 in element DP ($2\varepsilon = 1$ or 2 in edge DP). $\mathsf{WShuffle}_\square$ achieves a relative error of 0.15 to 0.3 with a smaller privacy budget: $\varepsilon = 0.2$.

3.8 Conclusion

In this paper, we made the first attempt (to our knowledge) to shuffle graph data for privacy amplification. We proposed wedge shuffling as a basic technique and then applied it to

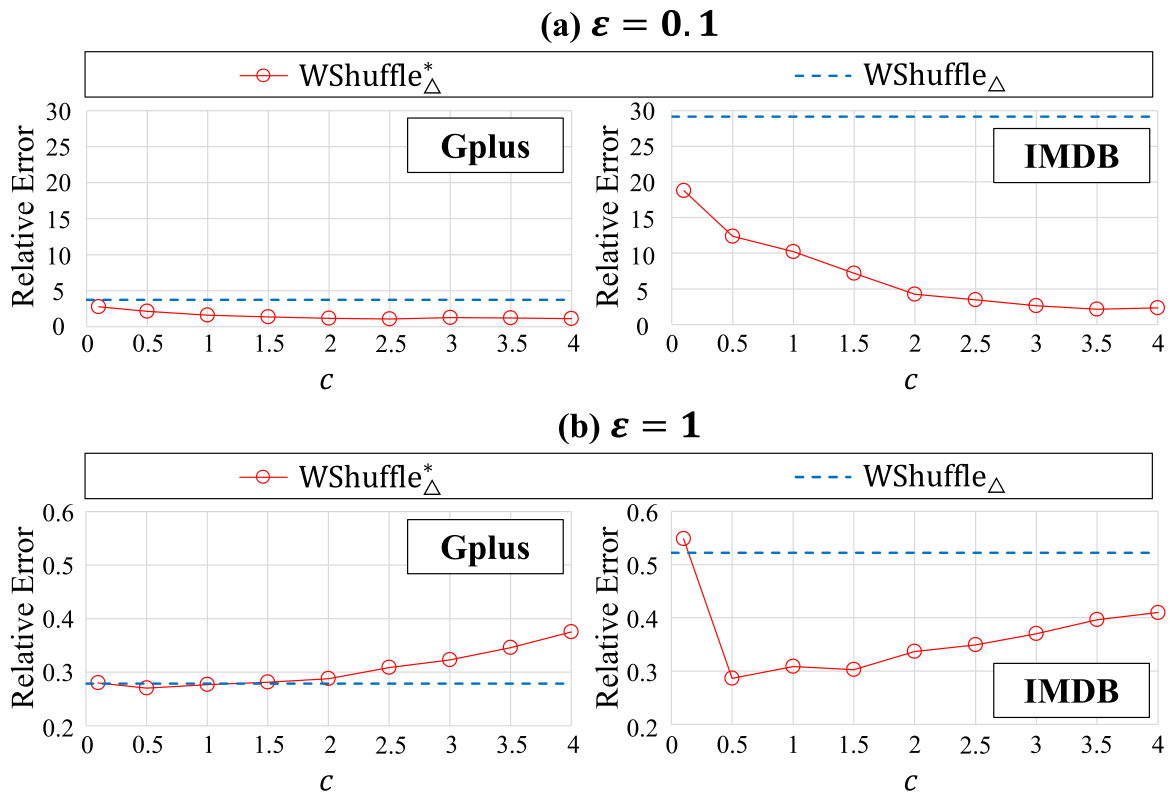


Figure 3.8. Relative error vs. parameter c in WShuffle$^*_\triangle$ ($n = 107614$ in Gplus, $n = 896308$ in IMDB).

one-round triangle and 4-cycle counting with several additional techniques. We showed upper bounds on the MSE for each algorithm. We also showed through comprehensive experiments that our one-round shuffle algorithms significantly outperform the one-round local algorithms and achieve a small relative error with a reasonable privacy budget, e.g., smaller than 1 in edge DP.

For future work, we would like to apply wedge shuffling to other subgraphs such as 3-hop paths [186] and k-triangles [120].

Chapter 3, in full, is a reprint of the material as it appears in Jacob Imola, Takao Murakami, and Kamalika Chaudhuri, Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022. "Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model." The dissertation author was a joint first author of this paper.

Chapter 4

Robustness of Locally Differentially Private Graph Analysis Against Poisoning

4.1 Introduction

Federated analytics enable a data analyst to gather useful information from data distributed across multiple users *without* centrally pooling the data. An important class of tasks in federated analytics is computing statistics over graph data, which has wide-spread applications ranging from market value prediction [141], fake news detection [22] to drug development [90]. Usually, the graph encodes sensitive user information, such as social network data, which raises privacy concerns. For instance, a mobile phone company might be interested in calculating statistics over the graph of call history which reveals an user's social interactions. To this end, local differential privacy (LDP) is currently the most popular model for achieving data privacy for federated analytics.

The distributed nature of LDP, however, leaves the door open for poisoning attacks. For example, an adversary can inject fake users into the system or compromise the accounts of real users to run untrusted applications on user devices. Consequently, there is no guarantee that these users will comply with the LDP protocols. The adversary can send carefully crafted malformed data from these non-compliant users and skew statistical estimates, including those involving only honest users.

Prior work, which focuses on tabular data [46, 34, 132], finds that poisoning attacks can

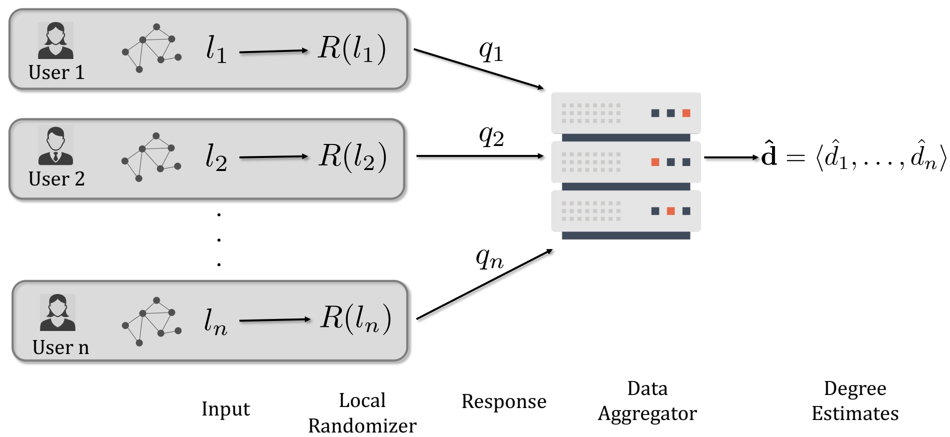


Figure 4.1. Graph analysis in the LDP setting.

be carried out against naive LDP protocols. However, the impact of poisoning under LDP for graph analysis is largely unexplored. In this paper, we initiate a formal study on the impact of such poisoning attacks on LDP protocols for graph statistics. We focus on the task of degree estimation, one of the most fundamental tasks in graph analysis.

A real-world use-case for a poisoning attack is as follows. Suppose a data analyst is interested in hiring the most influential nodes (users) of a graph for marketing and uses a node's degree as its measure of influence. Here, an adversary might want to promote a specific malicious node (user) to be selected as an influencer or prevent an honest node from being selected as an influencer. Concretely, suppose a single malicious user wants to (falsely) get themselves selected as the most influential node. If the LDP protocol used is the Laplace mechanism, where each user directly submits their (noisy) degree to the analyst (Fig. 4.1), then the malicious user can lie flagrantly and report their degree to be $n-1$. This can skew the response by as much as the number of users!

We address this challenge and design degree estimation protocols that are robust to poisoning attacks. Our algorithms are based on the key observation that graph data is naturally redundant – the information about an edge e_{ij} is shared by both users U_i and U_j. Leveraging this observation, we propose robust protocols based on two new ideas. First, we use *distributed* *information* – we collect the information about each edge from *both* users. The second idea is to *verify* – the data analyst can verify the consistency of the collected information by leveraging its redundancy. Specifically, as long as at least one of U_i or U_j is honest, the analyst can check for consistency[1] between the two bits obtained from U_i and U_j and detect malicious behavior.

If there are at most m malicious users, then with no privacy, the data analyst could flag malicious users as those with more than m edge inconsistencies in their reported edges since this is beyond a tolerable number of inconsistencies for an honest user. However, LDP requires randomization which makes the user's reports probabilistic. Consequently, the aforementioned simple consistency check cannot be applied directly to LDP protocols. We mitigate this challenge by proposing edge inconsistency confidence bounds under LDP which guarantees detection of malicious users without misclassifying the honest ones. Using our edge consistency checks we design novel degree estimation protocols under LDP which are robust to poisoning attacks.

In summary, we are the *first* to study the impact of poisoning on LDP degree estimation for graphs. Our main contributions are:

- We propose a formal framework for analyzing the robustness of a protocol. Specifically, we measure the robustness along two dimensions, *correctness* (for honest users) and *soundness* (for malicious users). Intuitively, good correctness means that the protocol is accurate for honest users, and good soundness means that it can detect/restrict malicious users.
- Based on the proposed framework, we study the impact of poisoning attacks on private degree estimation in the LDP setting. The attacks can be classified into two types: (1) input poisoning attacks where the adversary does not have access to implementation of the LDP protocol and can only falsify their input data (Fig. 4.2), and (2) response poisoning attacks where the adversary can tamper with the LDP implementation and directly manipulate the (noisy) responses of the LDP protocol (Fig. 4.3). The former is independent of LDP while the latter utilizes the characteristics of LDP. We observe that LDP makes a degree estimation

[1]For the ease of exposition, we disregard errors due to machine failures.

Table 4.1. Summary of correctness and soundness results in the paper. The $\tilde{O}$ notation asymptotically holds for $\varepsilon < 1$, and hides factors of $\log \frac{1}{\delta}$. $n-1$-tight indicates that there exists a worst-case attack that can skew the degree estimates by $n-1$. All the above results are attack-agnostic.

Protocol	Response Poisoning			Privacy Guarantee
	Correctness	Soundness		
R_{Lap}	$(\tilde{O}(\frac{1}{\varepsilon}), \delta)$	$(n-1)$-tight	Thm. 1	ε-Edge LDP
SimpleRR	$(\tilde{O}(m + \frac{m}{\varepsilon} + \frac{\sqrt{n}}{\varepsilon}), \delta)$	$(n-1)$-tight	Thm. 2	ε-Edge LDP
RRCheck	$(\tilde{O}(m + \frac{m}{\varepsilon} + \frac{\sqrt{n}}{\varepsilon}), \delta)$	$(\tilde{O}(m + \frac{m}{\varepsilon} + \frac{\sqrt{n}}{\varepsilon}), \delta)$	Thm. 3	ε-Edge LDP
Hybrid	$(\tilde{O}(\frac{1}{\varepsilon}), \delta)$	$(\tilde{O}(m + \frac{m}{\varepsilon} + \frac{\sqrt{n}}{\varepsilon}), \delta)$	Thm. 5	ε-Edge LDP

Protocol	Input Poisoning		
	Correctness	Soundness	
R_{Lap}	$(\tilde{O}(\frac{1}{\varepsilon}), \delta)$	$(n-1, \frac{1}{2})$	Thm. 6
SimpleRR	$(\tilde{O}(m + \frac{\sqrt{n}}{\varepsilon}), \delta)$	$(n-1, \frac{1}{2})$	Thm. 7
RRCheck	$(\tilde{O}(m + \frac{\sqrt{n}}{\varepsilon}), \delta)$	$(\tilde{O}(m + \frac{\sqrt{n}}{\varepsilon}), \delta)$	Thm. 8
Hybrid	$(\tilde{O}(\frac{1}{\varepsilon}), \delta)$	$(\tilde{O}(m + \frac{\sqrt{n}}{\varepsilon}), \delta)$	Thm. 9

protocol more vulnerable to poisoning – the impact of a response poisoning attack can be worse than that of any input poisoning attack.

- Leveraging the natural redundancy in graph data, we design robust degree estimation protocols under LDP that significantly reduce the impact of adversarial poisoning and compute degree estimates with high accuracy (Table 4.1).
- We evaluate our proposed robust degree estimation protocols under poisoning attacks on real-world datasets to demonstrate their efficacy in practice. We observe that even a relatively small number of malicious parties ($m = 1\%$) can stage significantly damaging poisoning attacks. This demonstrates the practical threat of such attacks. Nevertheless, our empirical results show that our proposed protocols are able to thwart poisoning attacks even with a large number of malicious users ($m = 37.5\%$).

4.2 Preliminaries

Notation. Throughout this paper, let $G = (V, E)$ be an undirected graph with V and E representing the set of nodes (vertices) and edges, respectively. We assume a graph with $n \in \mathbb{N}$

nodes, i.e., $V=[n]$ where $[n]$ denotes the set $\{1,2,\cdots,n\}$. Let $\mathscr{G}$ denote the domain of all graphs with n nodes. Each node $i \in V$ corresponds to a user U_i. Let $l_i \in \{0,1\}^n$ be the adjacency list for $U_i, i \in [n]$ where bit $l_i[j], j \in [n]$ encodes the edge e_{ij} between a pair of users U_i and U_j. Specifically, $l_i[j]=1$ if $e_{ij} \in E$ and $e_{ij}=0$ otherwise. Let $\mathbf{d}=\langle d_1,\ldots,d_n\rangle \in \mathbb{R}^n$ denote the vector of degrees in G. m denotes the number of malicious users.

4.2.1 Local Differential Privacy for Graphs

Our paper focuses on the *local* model of DP, one of the most popular models. The local model consists of a set of individual users (U) and an untrusted data aggregator (analyst); each user perturbs their data using a (local) DP algorithm and sends it to the aggregator which uses these noisy data to estimate certain statistics of the entire dataset.

The most popular privacy guarantee for graphs in the local setting is known as *edge* LDP [160, 168] which protects the existence of an edge between any two users. In other words, on observing the output, an adversary cannot distinguish between two graphs that differ in a single edge. Formally, we have:

Definition 1 (ε-Edge LDP[165])**.** *Let $R:\{0,1\}^n \mapsto \mathscr{X}$ be a randomized algorithm that takes an adajcency list $l \in \{0,1\}^n$ as input. We say $R(\cdot)$ provides ε-edge LDP if for any two neighboring lists $l,l' \in \{0,1\}^n$ that differ in one bit (i.e., one edge) and any output $s \in \mathscr{X}$,*

$$\Pr[R(l)=s] \leq e^{\varepsilon}\Pr[R(l')=s] \tag{4.1}$$

Randomized Response. Randomized Response ($RR_\rho(\cdot)$) [209] releases a bit $b \in \{0,1\}$ by flipping it with probability $\rho=\frac{1}{1+e^{\varepsilon}}$. We extend the mechanism to inputs in $\{0,1\}^n$ by flipping each bit independently with probability ρ (Alg. 18 in App. D.1) which satisfies ε-edge DP.

Laplace Mechanism. The Laplace mechanism(R_{Lap}) is a standard algorithm to achieve DP [74]. For degree estimation, each user U_i simply reports $\tilde{d_i}=d_i+\eta, \eta \sim Lap(\frac{1}{\varepsilon})$ where $Lap(b)$ represents the Laplace distribution with scale parameter b. This mechanism satisfies ε-edge DP.

Every protocol used in this paper is based on randomized response, the Laplace mechanism, or a composition thereof, and it is easy to show each one satisfies ε-edge LDP.

4.2.2 Protocol Setup

Problem Statement. We consider single round, non-interactive protocols in which each user $U_i, i \in [n]$ runs the local randomizer $R_i : \{0,1\}^n \to \mathscr{X}$ for some output space $\mathscr{X}$ on their adjacency lists l_i (Fig. 4.1). The data aggregator collects the noisy responses and applies a function $D : \mathscr{X}^n \to (\mathbb{N} \cup \{\perp\})^d$ to produce final degree estimates $\hat{\mathbf{d}} = \langle \hat{d}_1, \ldots, \hat{d}_n \rangle$). Here, $\hat{d}_i$ is the aggregator's estimate for d_i for user U_i. Note that the aggregator is allowed to output a special symbol $\perp$ for a user U_i if they believe the estimate $\hat{d}_i$ to be invalid (i.e., U_i is malicious).

Threat Model. In the execution of a protocol $\mathscr{P}$, a subset of users $\mathscr{M} \in [n]$ may be malicious. The malicious users may return arbitrary output with the goal of carrying out a poisoning attack on $\mathscr{P}$. Let $m = |\mathscr{M}|$ denote the number of malicious users. We refer to $\mathscr{H} = [n] \setminus \mathscr{M}$ as the set of honest users. We do not make any assumptions on how the malicious users are instantiated in practice – they could correspond to either fake accounts created by the adversary or a set of real accounts which have been compromised or a combination of both.

Based on the specifications of the practical implementation of the LDP, there is an important distinction between the way in which the malicious users may carry out their poisoning attacks. We outline them as follows:

- **Input Poisoning.** In this threat model, the users do not have access to the implementation of the LDP randomizer. For instance, mobile applications might run proprietary code which the users do not have permission to tamper with. Consequently, the only thing a malicious user can do is falsify their underlying input data, i.e., change their input from l_i to an arbitrary l'_i, and then report $q_i = R_i(l'_i)$ (Fig. 4.2).
- **Response Poisoning.** This is a stronger threat model where a malicious user has direct

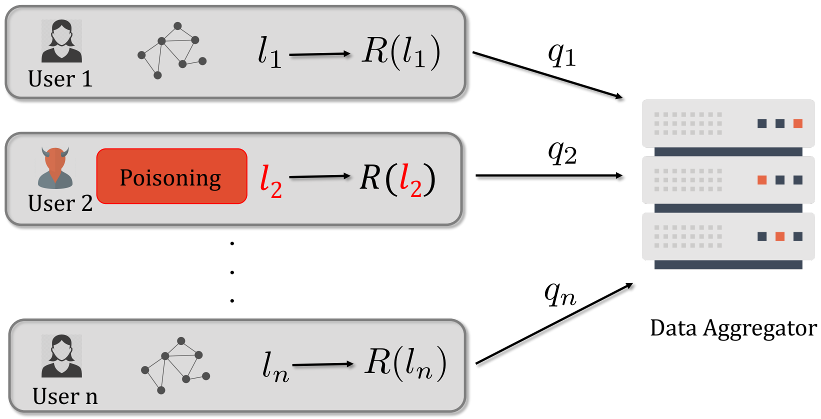


Figure 4.2. Illustration of an input poisoning attack.

control over the implementation of the LDP randomizer. For instance, the user could hack into the mobile application collecting their data. Consequently, the user can completely bypass the randomizer and submit an arbitrary response q_i (Fig. 4.3) to the aggregator.

Note that input poisoning attack applies to *any* protocol, private or not, because a user is free to change their input anytime. However, response poisoning attacks are unique to LDP – the distinction between an user's input and their response is a characteristic of LDP which we results in a separation in the efficacy of the two types of attacks (see Sec. 4.7 for details).

4.2.3 Motivating Attacks

For the Laplace mechanism, R_{Lap}, and randomized response mechanism, RR_ρ, outlined in Sec. 4.2.1, we present two concrete motivating attacks. We consider the attacks in the context of the task of influential node identification, where the goal of the data aggregator is to identify certain nodes with a high measure of "influence". This is useful for various applications where the influential nodes are selected for disseminating information/advertisement. Here, we consider the simplest measure of influence – degree; nodes with the top-k degrees are selected as the influencers. With the goal of modifying the influence of different users, a malicious user may

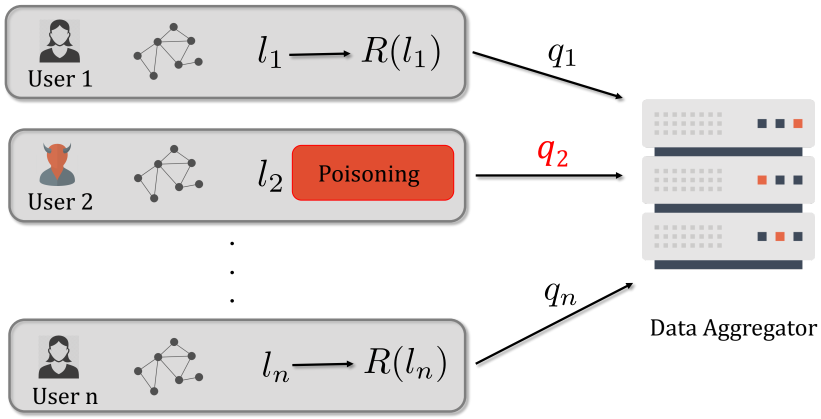


Figure 4.3. Illustration of a response poisoning attack.

choose to carry out the following attacks:

Degree Inflation Attack. In this attack, a target malicious user U_t wants to get themselves selected as an influential node. For R_{Lap}, since each user sends their degree d_i plus Laplace noise, the target user maximizes their degree estimate by simply sending $n-1$.

For RR_ρ, the target malicious user U_t colludes with a set of other users (for instance, by injecting fake users) as follows. All the non-target malicious users $U_i, i \in \mathscr{M} \setminus t$ report 1 for the edges corresponding to U_t. Additionally, U_t reports an all-one list.

Degree Deflation Attack. In this attack, the target is an honest user $U_t \in \mathscr{H}$ who is being victimized by a set of malicious users (for instance, the adversary compromises a set of real accounts with an edge to U_t) – $\mathscr{M}$ wants to ensure that U_t is *not* selected as an influential node. The attack strategy is to decrease the aggregator's degree estimate $\hat{d}_t$ for U_t. For R_{Lap}, there is no way for the malicious party to influence $\hat{d}_t$. However, for randomized response, the malicious users may report 0 for each edge in $\mathscr{M}$ connected to U_t, reducing their degree by this amount.

4.3 Quantifying Robustness

In this section, we present our formal framework for analyzing the robustness of a protocol for degree estimation. Specifically, we measure the robustness along two dimensions, *correctness* (for honest users) and *soundness* (for malicious users). Intuitively, good correctness means that the protocol is accurate for honest users, and good soundness means that it can detect/restrict malicious users. Hence, a protocol which has both properties is robust to poisoning attacks.

4.3.1 Metrics

Our notion of correctness guarantees that the protocol will produce an accurate degree estimate $\hat{d}_i$ for an honest user $U_i \in \mathscr{H}$ even under poisoning. On the other hand, good soundness prevents a malicious user $U_i \in \mathscr{M}$ from manipulating their degree estimate by too much *without* detection. We describe them in details as follows.

Correctness (For Honest Users). The *correctness* of a protocol assesses its resilience to manipulation of an honest user's estimator. Specifically, malicious users can adversely affect an honest user $U_i \in \mathscr{H}$ by

- tampering with the value of U_i's degree estimate $\hat{d}_i$ (by introducing additional error), or
- attempting to mislabel U_i as malicious (by influencing the aggregator to report $\hat{d}_i = \perp$)

Correctness protects the honest user U_i along the above dimensions and is formally defined as follows:

Definition 2. *(**Correctness**) Let $\langle R_1, \ldots, R_n \rangle$ be a non-interactive,* LDP *protocol for degree estimation producing estimates $\langle \hat{d}_1, \ldots, \hat{d}_n \rangle$. Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$ and $\mathscr{H}$ be the set of honest users. Then, the protocol is defined to be (α_1, δ_1)-correct w.r.t. an* *attack from $\mathscr{M}$ if for all input graphs $G \in \mathscr{G}$ we have:*

$$\forall U_i \in \mathscr{H}, \Pr\left[\hat{d}_i = \perp \vee |\hat{d}_i - d_i| \geq \alpha_1\right] \leq \delta_1, \tag{4.2}$$

where the above probability is taken w.r.t the randomness in the protocol and the attack.

The parameter α_1 dictates the accuracy of the estimate $\hat{d}_i$, and the parameter δ_1 dictates the chance of failure—that either of the aforementioned conditions fail to hold. Thus, if a protocol is (α_1, δ_1)-correct, it means that with probability at least $(1-\delta_1)$, the degree estimate $\hat{d}_i$ for any honest user $U_i \in \mathscr{H}$ has error *at most* α_1, *and* U_i is guaranteed to be *not* mislabeled as malicious.

Complementary to the above definition, we introduce the notion of α_1-*tight correctness*. A protocol is defined to be α_1-tight correct w.r.t an attack, if there exists a graph such that the attack is *guaranteed* to either skew the the degree estimate of at least one honest user by *exactly* α_1. We use this definition to show the existence of strong attacks that are guaranteed to be very successful in manipulating the data of an honest user, which motivates the need for robust solutions.

Lower the value of α_1 and δ_1, better is the robustness of the protocol for honest users.

Soundness (For Malicious Users). The *soundness* of a protocol assesses its ability to restrict adversarial manipulations of a malicious user's estimator. In particular, the protocol either returns an accurate estimate or returns $\hat{d}_i = \perp$ for these malicious users, regardless of the poisoning attack used. Formally, we use the following definition (which uses the complement event $\hat{d}_i \neq \perp \wedge |\hat{d}_i - d_i| \geq \alpha_2$):

Definition 3. ***(Soundness)*** *Let $\langle R_1, \ldots, R_n \rangle$ be a non-interactive,* LDP *protocol for degree estimation producing estimates $\hat{d}_1, \ldots, \hat{d}_n$. Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$ and $\mathscr{H}$ be the complement set of honest users. Then, the protocol is defined to be (α_2, δ_2)-sound w.r.t* *an attack from $\mathscr{M}$ if, for all input graphs $G \in \mathscr{G}$ we have*

$$\forall U_i \in \mathscr{M}, \Pr\left[\hat{d}_i \neq \perp \wedge |\hat{d}_i - d_i| \geq \alpha_2\right] \leq \delta_2, \tag{4.3}$$

where the above probability is taken w.r.t randomness in the protocol and attack.

Like with correctness, the parameter α_2 dictates the accuracy of the estimate $\hat{d}_i$, and δ_2 dictates the chance of failure. As an important distinction, note that the failure event is when $\hat{d}_i$ is both $\neq \perp$ *and* is inaccurate, as stated above. Thus, the $\vee$ used in the definition of correctness is replaced by a $\wedge$. In other words, a protocol is (α_2, δ_2)-sound if for any malicious user $U_i, i \in \mathscr{M}$, the protocol

- fails to identify U_i as malicious, *and*
- reports its degree estimate $\hat{d}_i$ with error *greater* than α_2

with probability at most δ_2.

Complementary to the above definition, we introduce the notion of α_1-*tight soundness*. A protocol is defined to be α_1-tight sound w.r.t an attack if there exists a graph $G \in \mathscr{G}$ such that at least one malicious user is *guaranteed* to have their degree estimate misestimated by *exactly* α_1 *without* getting detected. In other words, an $(n-1)$-tight sound/correct attack represents the strongest possible attack[2] – an user's estimate can *always* be skewed by the worst-case amount. We use this definition to motivate the need for more robust protocols.

Lower the value of α_2 and lower the value of δ_2, better is the robustness of the protocol for malicious users.

Note. Our proposed framework strives to provide a strong notion of robustness – not only are we able to guarantee accurate estimates, but also detect *and* flag individual malicious users (by reporting $\perp$).

[2] We do not consider self-edges or loops in the graph.

4.4 Impact of Poisoning on Baseline Protocols

Within our robustness framework, we analyze the two naive private mechanisms outlined in Sec. 4.2.1 – the Laplace mechanism and randomized response. The shortcomings of these mechanisms motivate the design of our robust protocols discussed later in the paper. We present all our results for the stronger threat model of response poisoning attacks first. We defer all our discussion for the input poisoning attack to Sec. 4.7.

4.4.1 Laplace Mechanism

The simplest mechanism for estimating an user's degree is the Laplace mechanism, R_{Lap}. Recall here, each user directly reports their noisy estimates. Consequently, the degree estimate of an honest user *cannot* be tampered with at all – the $\tilde{O}(\frac{1}{\varepsilon})$[3] term is due to the error of the added Laplace noise. This error is in fact optimal (matches that of central DP) for degree estimation. On the flip side, a malicious user can flagrantly lie about their estimate without detection resulting in the the worst-case soundness guarantee. Specifically, there exists a graph and an attack against R_{Lap} in which a malicious user is guaranteed to manipulate their true degree by $n-1$ – this holds for the case where the malicious user is an isolated node but lies that their degree is $n-1$. The robustness of R_{Lap} against response poisoning attacks is formalized as follows:

Theorem 1. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. The R_{Lap} protocol is $(\frac{1}{\varepsilon}\log\frac{n}{\delta}), \delta)$-correct with respect to any response poisoning attack from $\mathscr{M}$. However, there is a response poisoning attack $\mathscr{A}$ such that R_{Lap} is $(n-1)$-tight sound with respect to $\mathscr{A}$.*

The proof of the above theorem is in App. D.2.3. Thus according to our robustness framework, R_{Lap} has good correctness but not soundness. Intuitively, R_{Lap} fails to provide good soundness because there is no way to verify the malicious users' reports. It is important to note that R_{Lap} has good correctness guarantee even with $n-1$ malicious users while the worst-case soundness is inevitable even with a single malicious user.

[3] $\tilde{O}$ hides factors of $\log\frac{1}{\delta}$

Data: $\{l_1, \cdots, l_n\}$ where $l_i \in \{0,1\}^n$ is the adjacency list of user U_i
Result: $(\hat{d}_1, \cdots, \hat{d}_n)$ where $\hat{d}_i$ is the degree estimate for user U_i
Users;
foreach $i \in [n]$ **do**
    $q_i = RR_\rho(l_i)$;
end
Data Aggregator: foreach $i \in [n]$ **do**
    $count_i^1 = \sum_{j<i} q_j[i] + \sum_{i<j} q_i[j]$;
    $\hat{d}_i = \frac{1}{1-2\rho}(count_i^1 - \rho(n-1))$;
end
return $(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$

Algorithm 11: Naive degree counting algorithm based on randomized response. *SimpleRR*: $\{0,1\}^{n\times n} \mapsto \{\mathbb{N} \cup \{\perp\}\}^n$

4.4.2 Randomized Response

In this section, we look at an alternative mechanism where the users release their edges via randomized response. Recall that the information about an edge is shared between two users – the idea here is to leverage this *distributed information*. For our baseline algorithm, *SimpleRR* (described in Alg. 11), the data aggregator collects information about an edge from a *single* user. Specifically, for edge (i, j) with $i < j$, it simply uses the response from user U_i to decide if the edge exists. To estimate the degree, it counts the total number of edges to user U_i with the random variable $count_i^1$ and then computes a debiased estimate of the degree. Note that this naive approach is used by many prior works in local graph algorithms, such as [204, 165, 103, 105].

Formally for response poisoning attacks, we have:

Theorem 2. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. Then, the SimpleRR protocol is $(m\frac{e^\varepsilon+1}{e^\varepsilon-1} + \sqrt{n}\frac{\sqrt{(e^\varepsilon+1)\ln\frac{2n}{\delta}}}{e^\varepsilon-1}, \delta)$-correct with respect to any response poisoning attack from $\mathscr{M}$. However, there is a response poisoning attack $\mathscr{A}$ such that SimpleRR is $(n-1)$-tight sound with respect to $\mathscr{A}$.*

The above theorem is proved in App. D.2.4. For $\varepsilon < 1$, the correctness guarantee is $\approx m(1+\frac{1}{\varepsilon}) + \frac{\sqrt{n}}{\varepsilon}$. Intuitively, the $\frac{\sqrt{n}}{\varepsilon}$ term comes from the error introduced by randomized response. The $m(1+\frac{1}{\varepsilon})$ term comes from the adversarial behavior of the malicious users – m term is inevitable and accounts for the worst case scenario where all m malicious users are colluding (see Sec. 4.8 for more details), while the $\frac{1}{\varepsilon}$ factor corresponds to the scaling factor required for de-biasing. This observation is in line with prior work [46] that assesses the impact of poisoning attacks on tabular data. Clearly, smaller the value of ε, worse is the impact of the attack.

Similar to the Laplace mechanism, *SimpleRR* is $(n-1)$-tight sound, i.e., a malicious user can always get away with the worst-case $n-1$ error. This happens when U_n is an isolated node who acts maliciously and reports an all-one list. Thus, once again this worst-case soundness is inevitable even with a single malicious user. In the next section, we discuss how to leverage the data redundancy in graphs and verify the data collected via randomized response to improve the soundness.

4.5 Improving Soundness with Verification

In this section, we present our proposed protocol for robust degree estimation. As discussed in the previous section, the naive *SimpleRR* protocol offers poor soundness. To tackle this, we propose a new protocol, *RRCheck*, that enhances *SimpleRR* with a consistency check based on the redundancy in graph data and flags users if they fail the check. Consequently, *SimpleRR* improves the soundness guarantee significantly. We observe that with higher privacy (lower ε), the protocol is less robust. We conclude this section by analyzing *RRCheck* under no privacy constraint—the difference between the correctness and soundness guarantees are the *price of privacy*.

4.5.1 *RRCheck* Protocol

The *RRCheck* protocol is described in Alg. 12 and works as follows. *RRCheck* enhances the data collected by *SimpleRR* with verification – for edge $e_{ij} \in E$, instead of collecting a noisy response from just one of the users U_i or U_j, *RRCheck* collects a noisy response from *both* users. This creates data redundancy which can then be checked for consistency. Specifically, the estimator counts only those edges e_{ij} for which *both* U_i and U_j are consistent and report a 1. The count of noisy edges involving user U_i is then given by:

$$count_i^{11} = \sum_{j \in [n] \setminus i} q_i[j] q_j[i].$$

The unbiased degree estimate of U_i is computed as follows:

$$\hat{d_i} = \frac{count_i^{11} - \rho^2(n-1)}{1-2\rho}. \tag{4.4}$$

For robustness, *RRCheck* imposes a check on the number of instances of inconsistent reporting (U_i and U_j differ in their respective bits reported for their mutual edge e_{ij}). For every user U_i, the protocol has an additional capability of returning $\perp$ whenever the consistency check fails, indicating that the aggregator believes that user U_i is malicious. The intuition is that if the user U_i is malicious and attempts to poison a lot of the edges, then there would be a large number of inconsistent reports corresponding to the edges to honest users. *RRCheck* counts the number of inconsistent reports for user U_i as:

$$count_i^{01} = \sum_{j=1}^{n} (1 - q_i[j]) q_j[i],$$

i.e., the number of edges connected to user U_i[4] for which they reported 0 and user U_j reported 1. Intuitively, the check computes the expected number of inconsistent reports assuming user U_i to be honest and flags U_i in case the reported number is outside a confidence interval. Formally, if

$$|count_i^{01} - \rho(1-\rho)(n-1)| \leq \tau, \tag{4.5}$$

[4]It is symmetric (and doesn't give additional information) to count edges for which user U_i reports 1 and U_j reports 0.

Data: Adjacency lists $\{l_1, \cdots, l_n\}$; τ, threshold for consistency check
Result: $(\hat{d}_1, \cdots, \hat{d}_n)$ where $\hat{d}_i$ is the degree estimate for user U_i

$\rho = \frac{1}{1+e^{\varepsilon}}$;
Users;
foreach $i \in [n]$ **do**
    $q_i = RR_\rho(l_i)$
end
Data Aggregator;
foreach $i \in [n]$ **do**
    $count_i^{11} = \sum_{j \in [n] \setminus i} q_i[j] q_j[i]$;
    $count_i^{01} = \sum_{j \in [n] \setminus i} (1 - q_i[j]) q_j[i]$;
    if $|count_i^{01} - \rho(1-\rho)(n-1)| \leq \tau$ **then**
        $\hat{d}_i = \frac{1}{1-2\rho}(count_i^{11} - \rho^2(n-1))$;
    else
        $\hat{d}_i = \perp$
end
return $(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$

Algorithm 12: Degree estimation algorithm with checks, *RRCheck*: $\{0,1\}^{n \times n} \mapsto \{\mathbb{N} \cup \{\perp\}\}^n$.

then set $\hat{d}_i = \perp$, where $\tau = m + \sqrt{3n\rho \ln \frac{2}{\delta}}$ is a threshold. This check forces a malicious user to send a response with only a small number of poisoned edges (as allowed by the threshold τ), thereby significantly restricting the impact of adversarial manipulations. For example, they are not able to indicate they are connected to all users in the graph, as this would produce a large number of inconsistent edges.

Note that due to the randomization required for LDP, some honest users might also fail the check. However, we observe that for two honest users U_i and U_j, the product term $(1 - q_i[j])q_j[i]$ follows the Bernoulli$(\rho(1-\rho))$ distribution, irrespective of whether the edge e_{ij} exists. Consequently $count_i^{01}$ is tightly concentrated around its mean. This ensures that the probability of mislabeling an honest user (by returning $\perp$) is low.

Formally, we are able to show the following correctness and soundness guarantees for *RRCheck*.

Theorem 3. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. Then, the RRCheck protocol* *run with threshold* $\tau = m + \sqrt{2\rho n \ln\frac{4n}{\delta}}$ *is* $\left(2m(\frac{e^{\varepsilon}+1}{e^{\varepsilon}-1}) + 4\sqrt{n}\frac{\sqrt{(e^{\varepsilon}+1)\ln\frac{4n}{\delta}}}{e^{\varepsilon}-1}, \delta\right)$*-correct and sound with respect to any response poisoning attack from* $\mathscr{M}$*.*

This theorem is proved in App. D.2.5. The additional verification of *RRCheck* results in a clear improvement — the malicious users can now skew their degree estimates only by a limited amount (as determined by the threshold τ) or risk getting detected, which results in a better soundness guarantee. Specifically, a malicious user can now only skew their degree estimate by at most $\tilde{O}\big(m(1+\frac{1}{\varepsilon}) + \frac{\sqrt{n}}{\varepsilon}\big)$ for response poisoning attacks, respectively (as compared to $n-1$ in Thm. 2). It is important to note that the above results are completely *attack-agnostic* – they hold for *any* attack, for any number of malicious users, and all graphs.

Note that the correctness and soundness guarantees in the above theorem worsen with smaller ε. This is because at lower privacy, the collected responses are more noisy thereby making it harder to distinguish honest users from malicious ones. In particular, a protocol should not return $\perp$ for honest users (i.e., mislabel them) to ensure good correctness. Consequently, more malicious error is tolerated before a $\perp$ is returned for a malicious user. This is evident in Eq. 4.5–observe the threshold τ grows with smaller ε. We expand on this price of privacy in the next section.

It is interesting to note that for response poisoning, the degree deflation attack (Sec. 4.2.3) represents a worst-case attack for correctness – the attack can skew an honest user's degree estimate by $\Omega\big(m(1+\frac{1}{\varepsilon}) + \frac{\sqrt{n}}{\varepsilon}\big)$. Similarly, the degree inflation attack (Sec. 4.2.3) can skew a malicious user's degree estimate by $\Omega\big(m(1+\frac{1}{\varepsilon}) + \frac{\sqrt{n}}{\varepsilon})\big)$ resulting in the worst-case soundness.

4.5.2 Price of Privacy

The randomization required to achieve privacy adversely impacts a protocol's robustness to poisoning. Here, we perform an ablation study and formalize the price of privacy by comparing to the correctness and soundness of *non-private* protocols. For this, we adapt our consistency check to the non-private setting via the *DegCheck* protocol (Alg. 13) as described below. First,

Data: $\{l_1, \cdots, l_n\}$ where $l_i \in \{0,1\}^n$ is the adjacency list of user U_i
Result: $(\hat{d}_1, \cdots, \hat{d}_n)$ where $\hat{d}_i$ is the degree estimate for user U_i

Users;
foreach $i \in [n]$ **do**
  $q_i = l_i$;
end
Data Aggregator;
foreach $i \in [n]$ **do**
  $count_i^{11} = \sum_{j \in [n] \setminus i} q_i[j] q_j[i]$;
  $count_i^{01} = \sum_{j \in [n] \setminus i} (1 - q_i[j]) q_j[i]$;
  $count_i^{10} = \sum_{j \in [n] \setminus i} q_i[j] (1 - q_j[i])$;
  if $(count_i^{01} + count_i^{10}) \leq m$ **then**
    $\hat{d}_i = count_i^{11}$;
  else
    $\hat{d}_i = \perp$;
end
return $(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$

Algorithm 13: Degree checking algorithm without privacy, *DegCheck*: $\{0,1\}^{n \times n} \mapsto \{\mathbb{N} \cup \{\perp\}\}^n$.

every user reports their true adjacency list to the data aggregator. The data aggregator then employs a consistency check to identify the malicious users. Due to the absence of randomization, the check is much simpler here and involves just ensuring that the number of inconsistent reports for user U_i is bounded by m, i.e., $count_i^{01} + count_i^{10} \leq m$. In case the check goes through, the aggregator can directly use $count_i^{11}$, the count of the edges where both users have reported 1s consistently, as the degree estimate $\hat{d}_i$.

We quantify the impact of the poisoning attacks on *DegCheck* as follows.

Theorem 4. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. Then, there are poisoning attacks $\mathscr{A}_1$ and $\mathscr{A}_2$ such that the DegCheck protocol is m-tight correct with respect to $\mathscr{A}_1$ and* $(\min\{2m-1, n-1\})$*-tight sound with respect to $\mathscr{A}_2$.*

The proof of the above theorem is in App. D.2.6. Note that the robustness guarantees are tight in that there are attacks which always successfully attain m error for an honest user and $\min\{2m-1, n-1\}$ for a malicious user. Thus, the low-order manipulation term of $O(m)$ is inevitable even for non-private protocols based on consistency checks.

Comparing Thm. 4 to Thm. 3, we see an improvement in both correctness and soundness guarantees over the private protocols – the malicious users can skew the degree estimates by only $O(m)$, and the $\tilde{O}(\frac{m}{\varepsilon} + \frac{\sqrt{n}}{\varepsilon})$ terms have disappeared. This highlights the price of privacy – the private protocols incur additional error due to the inherent randomization of LDP.

Thus, our proposed *RRCheck* protocol shows that the soundness of a degree estimation protocol can be significantly improved by leveraging the redundancy in graph data. Additionally, we observe the robustness of the protocol worsens with higher privacy and we explicitly formalize price of privacy.

4.6 Improving Correctness with A Hybrid Protocol

The robustness guarantees for *RRCheck* contain a $\tilde{O}(\frac{\sqrt{n}}{\varepsilon})$ term coming from the error in randomized response. This is inherent in *any* randomized response based mechanism [21, 36, 67] since each of the n bits of the adjacency list need to be independently randomized. Unfortunately, this dependence on n has an adverse effect on the utility of the degree estimates. Typically, real-world graphs are sparse in nature with maximum degree $d_{max} \ll n$. Hence, the $\tilde{O}(\frac{\sqrt{n}}{\varepsilon})$ noise term completely dominates the degree estimates resulting in poor accuracy for the honest users (i.e. poor correctness). On the other hand, R_{Lap} provides a more accurate degree estimate for the honest users but has the worst-case $(n-1)$-tight soundness (see Thm. 1). In this section, we present a mitigation strategy. The key idea is to combine the two approaches and use a hybrid protocol, *Hybrid*, that achieves the best of both worlds:

- correctness guarantee of R_{Lap}, and
- soundness guarantee of *RRCheck*.

The *Hybrid* protocol is outlined in Alg. 14 and described as follows. Each user U_i prepares two responses – the noisy adjacency list, q_i, randomized via RR_ρ, and a noisy degree estimate, $\tilde{d}_i^{lap}$, perturbed via R_{Lap}, and sends them to the data aggregator. U_i divides the privacy budget between the two responses according to some constant $c \in (0,1)$. The data aggregator first processes each list q_i to employ the same consistency check on $count_i^{01}$ as that of the *RRCheck* protocol (Step 9). In case the check passes, the aggregator computes the unbiased degree estimate $\tilde{d}_i^{rr}$ from $count_i^{11}$, in the exact same way as *RRCheck*. Note that $\tilde{d}_i^{rr}$ and $\tilde{d}_i^{lap}$ are the noisy estimates of the *same* ground truth degree, d_i, computed via two different randomization mechanisms. To this end, the aggregator employs a second check (Step 10) to verify the consistency of the two estimates as follows:

$$|\tilde{d}_i^{rr} - \tilde{d}_i^{lap}| \leq \frac{2\tau}{1-2\rho} + \frac{1}{(1-c)\varepsilon} \ln \frac{2n}{\delta},$$

where ρ in this case is equal to $\frac{1}{1+e^{c\varepsilon}}$. This check accounts for the error from $\tilde{d}_i^{rr}$ (the $\frac{2\tau}{1-2\rho}$) term, and the error from $\tilde{d}_i^{lap}$ (the $\frac{1}{(1-c)\varepsilon} \ln \frac{2n}{\delta}$ term). Finally, the protocol returns $\perp$ if either of the checks fail. In the event that both the checks pass, the aggregator uses $\tilde{d}_i^{lap}$ (obtained via R_{Lap}) as the final degree estimate $\hat{d}_i$ for U_i.

Each $\hat{d}_i^{rr}$ estimate is computed identically to that of *RRCheck*. *Hybrid* allows a user to send an even more accurate estimate of their degree – to prevent malicious users from outright lying about this value, $\hat{d}_i^{lap}$ is compared to $\hat{d}_i^{rr}$. This allows *Hybrid* to enjoy the correctness guarantee of R_{Lap} and the soundness guarantee of *RRCheck*. Formally,

Theorem 5. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. Then, for all $c \in (0,1)$, the Hybrid protocol run with $\tau = m + \sqrt{2\rho n \ln \frac{8n}{\delta}}$ is*

$$\left(\frac{\ln \frac{2n}{\delta}}{(1-c)\varepsilon}, \delta\right) - \textit{correct and}$$

$$\left(4m\left(\frac{e^{c\varepsilon}+1}{e^{c\varepsilon}-1}\right) + 8\sqrt{n}\frac{\sqrt{(e^{c\varepsilon}+1)\ln \frac{8n}{\delta}}}{e^{c\varepsilon}-1} + \frac{\ln \frac{2n}{\delta}}{(1-c)\varepsilon}, \delta\right)\textit{-sound}$$

with respect to any response poisoning attack.

The proof is in App. D.2.7. We remark that *Hybrid* achieves the optimal correctness of

Data: Adjacency lists $\{l_1, \cdots, l_n\}$; τ, threshold for consistency check
Result: $(\hat{d}_1, \cdots, \hat{d}_n)$ where $\hat{d}_i$ is the degree estimate for user U_i

$\rho = \frac{1}{1+e^{c\varepsilon}}$;
Users;
Select $c \in (0,1)$;
foreach $i \in [n]$ **do**
  $q_i = RR_\rho(l_i)$;
  $\tilde{d}_i^{lap} = \|l_i\|_1 + Lap(\frac{1}{(1-c)\varepsilon})$;
end
Data Aggregator;
foreach $i \in [n]$ **do**
  $count_i^{11} = \sum_{j\in[n]\setminus i} q_i[j]q_j[i]$;
  $count_i^{01} = \sum_{j\in[n]\setminus i} (1-q_i[j])q_j[i]$;
  if $|count_i^{01} - \rho(1-\rho)(n-1)| \leq \tau$ **then**
    $\tilde{d}_i^{rr} = \frac{1}{1-2\rho}(count_i^{11} - \rho^2(n-1))$;
    if $|\tilde{d}_i^{rr} - \tilde{d}_i^{lap}| \leq \frac{2\tau}{1-2\rho} + \frac{1}{(1-c)\varepsilon} \ln \frac{2n}{\delta}$ **then**
      $\hat{d}_i = \tilde{d}_i^{lap}$;
    else
      $\hat{d}_i = \perp$;
  else
    $\hat{d}_i = \perp$;
end
return $(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$

Algorithm 14: Hybrid algorithm *Hybrid*: $\{0,1\}^{n\times n} \mapsto \{\mathbb{N} \cup \{\perp\}\}^n$

$\tilde{O}(\frac{1}{\varepsilon})$ that is achievable under LDP. This is due to the fact that the data aggregator uses $\tilde{d}_i^{lap}$ as its final degree estimate. The soundness guarantee can be written as $\tilde{O}(m(1+\frac{1}{\varepsilon}) + \frac{\sqrt{n}}{\varepsilon})$ which is the same as that of *RRCheck*. This is enforced by the two consistency checks. Hence, the hybrid mechanism achieves the best of both worlds.

4.7 Results for Input Poisoning Attacks

So far we have only considered response poisoning attacks where the malicious users are free to report arbitrary responses to the data aggregator. However, to carry out such an attack in practice, a user would have to bypass the LDP data collection mechanism. Concretely, if a mobile application were used to collect a user's data, a malicious user would have to hack into the software and directly report their poisoned response. On the other hand, for input poisoning the malicious user needs to just lie about their input to the application (for instance, by misreporting their list of friends) which is always possible. Hence clearly, input poisoning attacks are more easily realizable in practice. In fact, in certain cases the malicious users might be restricted to just input poisoning attacks due to the implementation of the LDP mechanism. For instance, mobile applications might have strict security features in place preventing unauthorized code tampering. Another possibility is cryptographically verifying the randomizers [123] to ensure that all the steps of the privacy protocol (such as, noise generation) is followed correctly. Given its very realistic practical threat, in this section we study the impact of input poisoning attacks.

Note that input poisoning attacks are strictly weaker than response poisoning attacks. This is because the poisoned input is randomized to satisfy LDP in the former which introduces noise in the final output, thereby weakening the adversary's signal. Hence intuitively, we hope to obtain better robustness against input poisoning attacks. In what follows, we formalize the above intuition for degree estimation. We first investigate the baseline protocols R_{Lap} and *SimpleRR* and show that while input poisoning attacks are less damaging than response poisoning attacks, the protocols still suffer from poor robustness guarantees. Next, we show that our proposed protocols, *RRCheck* and *Hybrid*, offer improved robustness against input poisoning attacks. These results demonstrate a separation between the efficacy of response and input poisoning attacks.

Recall in the Laplace mechanism, each user simply reports a private estimate of their degree. Under input poisoning attacks, Laplace noise is added to the poisoned input before it is reported to the data aggregator. Consequently, the response poisoning attack in which a malicious user could *deterministically* report their degree as $n-1$ (Thm. 1) is no longer possible – in order to manipulate their degree by $n-1$, the malicious user needs to get lucky with the sampled Laplace noise, resulting in the following theorem:

Theorem 6. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. The R_{Lap} protocol is* $(\frac{1}{\varepsilon}\ln\frac{n}{\delta}, \delta)$

correct and $(n-1,\frac{1}{2})$-sound with respect to any input poisoning attack from $\mathscr{M}$.

The proof of the above theorem is in App. D.2.8. Unsurprisingly, compared to Thm. 1 for response poisoning attacks, the correctness guarantee is unchanged because no attack is possible for honest users. However, the soundness guarantee is different. Thm. 1 delineates an $(n-1)$-tight soundness guarantee, demonstrating the feasibility of the worst-case attack in which a malicious user can *always* manipulate their degree by $n-1$. In contrast, R_{Lap} is $(n-1,\frac{1}{2})$-sound with respect to any input poisoning attack. This is because the sampled Laplace noise is negative with probability $\frac{1}{2}$. Hence, even a worst-case malicious user who sends the maximum degree estimate of $n-1$ will only get assigned a final estimate this high if the sampled noise is non-negative. Thus, the noise from the Laplace mechanism prevents the adversary from carrying out the deterministic worst-case attack.

Next, we show the result for *SimpleRR*, our second baseline protocol. There is an improvement in both correctness and soundness, because the adversary's signals in the poisoned data (such as, a malicious user indicating they share an edge with every other user, or m malicious users intentionally deleting their edges to an honest user), are noised via randomized response which weakens them.

Theorem 7. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}|=m$. The SimpleRR protocol is $(m+\sqrt{n}\frac{\sqrt{2(e^{\varepsilon}+1)\ln\frac{4n}{\delta}}}{e^{\varepsilon}-1},\delta)$-correct and $(n-1,\frac{1}{2})$-sound with respect to any input poisoning attack from $\mathscr{M}$.*

The above theorem is proved in App. D.2.9. Written asymptotically, the correctness guarantee of Thm. 7 is $\tilde{O}(m+\frac{\sqrt{n}}{\varepsilon})$, which improves the guarantee over response poisoning attacks (Thm. 2) by a factor of $\frac{m}{\varepsilon}$. This shows a separation between input and response poisoning attacks. A similar case holds for soundness – while *SimpleRR* is $(n-1)$-tight sound under response poisoning attacks, for input poisoning attacks, it is $(n-1,\frac{1}{2})$-sound. The implications of this observation are similar to those of Thm. 6 as discussed above.

Despite exhibiting improvement over response poisoning attacks, both naive protocols still fall short of providing acceptable soundness guarantees. Here, we analyze the robustness of our proposed protocols, *SimpleRR* and *Hybrid*, under input poisoning attacks. For both the mechanisms here, we are able to a set a smaller value for τ, the threshold for checking the number of inconsistent edges. This is because the number of inconsistent edges is more concentrated around its means, and hence, a tighter confidence interval with a smaller τ suffices. Thus, both the correctness and soundness of the protocols are improved. Formally for *SimpleRR*, we have:

Theorem 8. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. The protocol RRCheck run with $\tau = m(1-2\rho) + \sqrt{8\max\{\rho n, m\}\ln\frac{8n}{\delta}}$ is $(2m + 4\sqrt{\max\{n, m(e^{\varepsilon}+1)\}}\frac{\sqrt{2(e^{\varepsilon}+1)\ln\frac{8n}{\delta}}}{e^{\varepsilon}-1}, \delta)$-correct and sound with respect to any input poisoning attack from $\mathscr{M}$.*

The proof is in App. D.2.10. For typical values of ε, the correctness and soundness guarantees can be written as $(\tilde{O}(m + \frac{\sqrt{n}}{\varepsilon}), \delta)$ (because $\sqrt{m(e^{\varepsilon}+1)} \leq \sqrt{n}$). Compared to Thm. 3 for response poisoning attacks, there is an improvement of $\frac{m}{\varepsilon}$ which is a direct consequence of a smaller τ.

For *Hybrid*, we have:

Theorem 9. *Let $\mathscr{M}$ be a set of malicious users with $|\mathscr{M}| = m$. For any $c \in (0,1)$, the Hybrid protocol run with threshold $\tau = m(1-2\rho) + \sqrt{8\max\{\rho n, m\}\ln\frac{8n}{\delta}}$ is $(\frac{1}{(1-c)\varepsilon}\ln\frac{4n}{\delta}, \delta)$-correct and $(4m + 8\sqrt{\max\{n, m(e^{c\varepsilon}+1)\}}\frac{\sqrt{2(e^{c\varepsilon}-1)\ln\frac{8n}{\delta}}}{e^{c\varepsilon}+1}, \delta)$-sound with respect to any input poisoning attack from $\mathscr{M}$.*

This theorem is proved in App. D.2.11. Written asymptotically, the correctness guarantee of *Hybrid* is $(\tilde{O}(\frac{1}{\varepsilon}), \delta)$, and its soundness is $(\tilde{O}(m + \frac{\sqrt{n}}{\varepsilon}), \delta)$. Compared with Thm. 5 for response poisoning attacks, *Hybrid* offers similar correctness since the data aggregator uses the degree estimate collected via R_{Lap} as its final estimate as before. However, the soundness guarantee is improved by an additive factor of $O(\frac{m}{\varepsilon})$, which comes from the smaller threshold τ.

In conclusion, we observe that response poisoning attacks are more damaging than input poisoning attacks. In other words, our proposed degree estimation protocols offer better robustness against input poisoning attacks.

4.8 Discussion

Adversary Collusion. It is important to note that all our results (Thms. 1 to 9) are completely *attack-agnostic* in their respective classes (response or input poisoning). In other words, our robustness guarantees hold against *any* attack with the malicious users free to follow arbitrary collusion strategies. A direct consequence of the above statement is that our results must hold against the worst-case scenario where *all* m malicious users are colluding with each other. Note that our consistency checks work for the case where an edge is shared with at least one honest user. Detecting malicious behavior for subgraphs controlled completely by the malicious users is beyond the scope of our robustness results since the malicious users can consistently lie about their common edges. For instance, in the degree inflation attack, the target malicious user can always expect its degree to be inflated by at least $m-1$ when all the malicious users are colluding. Formally, this is reflected by the $\Omega(m)$ term in all of our results including the non-private base case (Thm. 4).

One strategy to deal with this worst-case collusion is as follows. In addition to using our proposed protocols for collecting the data, the aggregator could perform some analysis on the graph structure to detect possible collusion patterns. Some collusion patterns to detect could be star-graphs where the non-center nodes have very low degree and disconnected cliques. For this we can borrow techniques from the rich body of work in social network collusion analysis [222, 181, 11, 69].

Auxiliary Information About Attacks. As mentioned above, our results do not make any assumptions on the data distribution or adversary. Hence, our results hold for the worst-case attacks. However, in case the data aggregator has some auxiliary information about the problem setup, one can expect to get even better robustness guarantees. For instance, our discussion in Sec. 4.7 shows that restricting the attacks to just input manipulation leads to improvement in the robustness guarantees.

Difference From Tabular Data. Recall, that the key observation behind our robustness protocols is that graph data is naturally redundant. Collecting this distributed information and verifying its consistency forms the crux of our technical idea. However, one of the key differences between graph data and tabular data is that the later has no natural redundancy. As a result, we cannot propose robust protocols for analyzing tabular data without making assumptions on the problem setting. This is corroborated by the ad-hoc defense strategies proposed in prior work (see Sec. 4.10) – they are tailored to specific attacks, make strong assumptions about the data distribution and/or require access to prior knowledge.

4.9 Evaluation

In this section, we present our evaluation results. Our evaluation seeks to empirically answer the following questions:

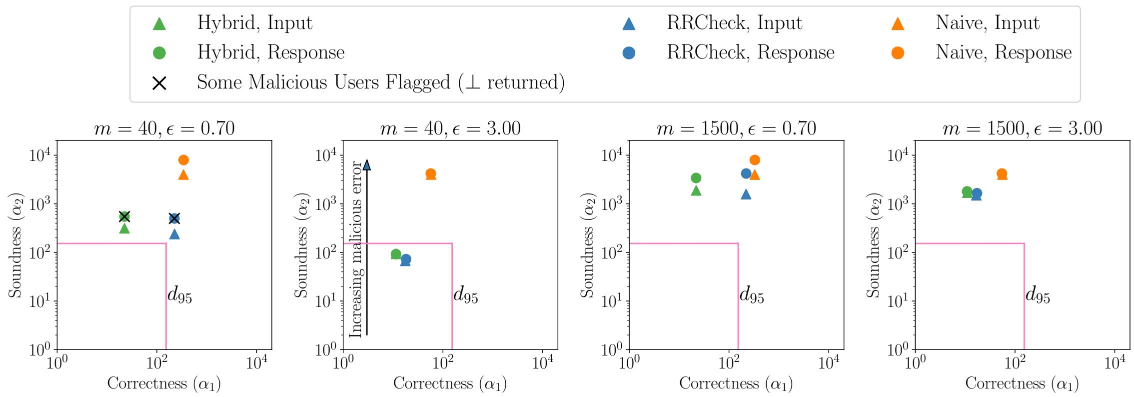
- **Q1.** How do the different protocols perform in terms of correctness and soundness?
- **Q2.** How do the efficacies of input and response poisoning attacks compare?
- **Q3.** What is the impact of the privacy parameter ε on the poisoning attacks?

4.9.1 Experimental Setup

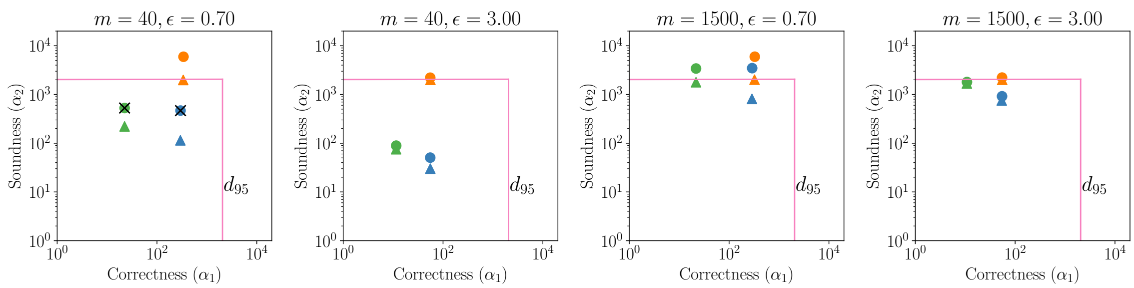
Datasets. We evaluate our protocols on two graphs – a real-world sparse graph and a synthetically generated dense graph.

- *FB.* This graph corresponds to data from Facebook [142] representing the friendships of 4082 Facebook users. The graph has 88K edges.
- *Syn.* To test a more dense regime, we evaluate our protocols on a synthetic graph generated

using the Erdos-Renyi model [81] with parameters $G(n = 4000, p = 0.5)$ (n is the number of edges; p is the probability of including any edge in the graph). The graph has ≈ 8 million edges.



(a) *FB*: Degree Inflation Attack

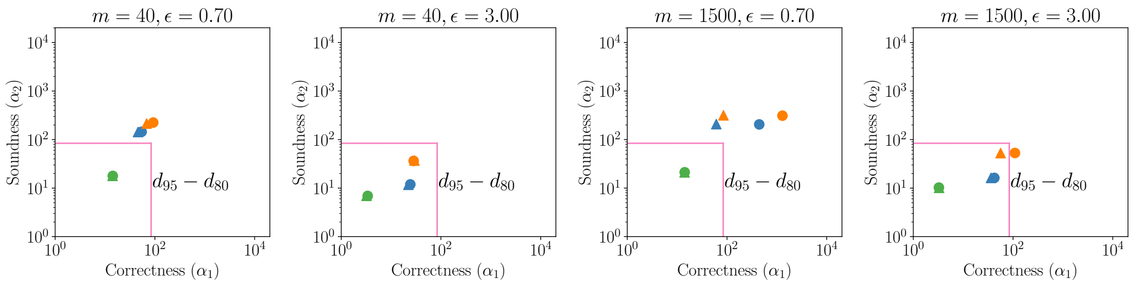


(b) *Syn*: Degree Inflation Attack

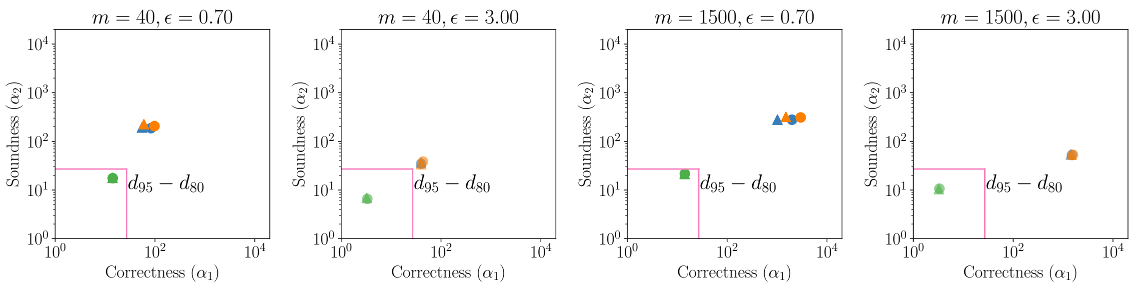
Figure 4.4. Robustness Analysis for Degree Inflation Attack: We plot the empirical correctness (error of honest user) and soundness (error of malicious user). d_{95} denotes the 95-th percentile of the degree distribution.

Protocols. We evaluate our proposed *RRCheck* and *Hybrid* protocols and use *SimpleRR* as a baseline protocol with poor robustness.

Attacks. For each protocol, we evaluate two types of attacks – degree inflation and degree deflation where the goal of the malicious users is to increase (resp. decrease) the degree estimate of a target malicious (resp. honest) user by as much as possible. We choose these attacks because firstly, they can meet the asymptotic theoretical error bounds. Secondly, these attacks are grounded in real-world motivations and represent practical threats (see Sec. 4.2.3). For each



(a) *FB*: Degree Deflation Attack



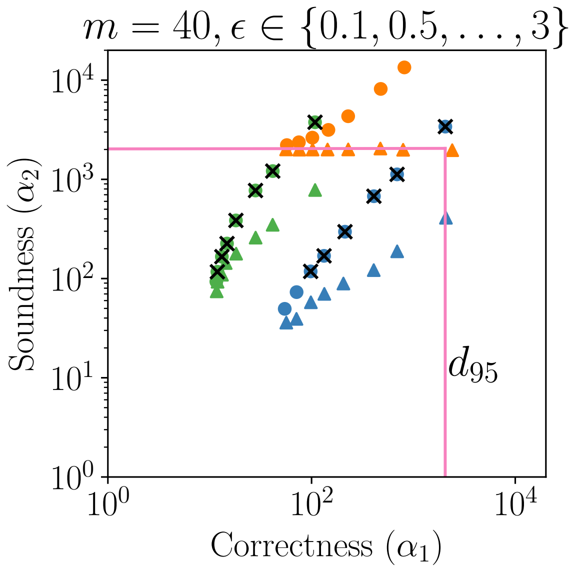
(b) *Syn*: Degree Deflation Attack

Figure 4.5. Robustness Analysis for Degree Deflation Attack: We plot the empirical correctness (error of honest user) and soundness (error of malicious user). d_k denotes the degree of the k percentile node.

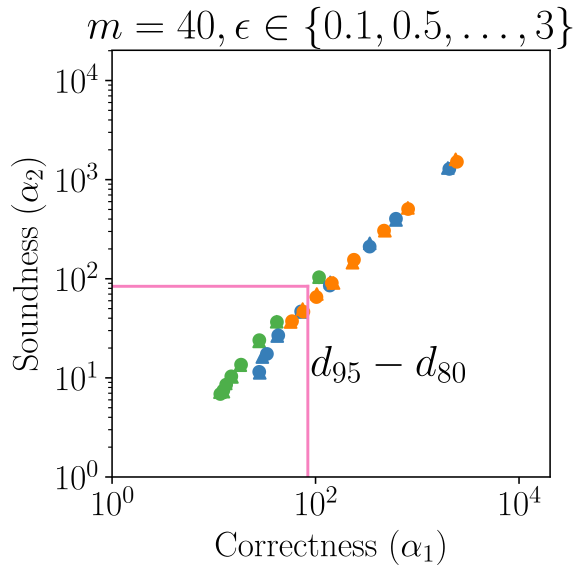
attack type, we consider both input and response poisoning versions. In the following, let U_t represent the target user.

RRCheck. For the degree inflation attack, the non-target malicious users always report a 1 for the malicious user U_t (i.e. that they are connected to U_t) in the hopes of increasing U_t's degree estimate. Likewise, U_t reports 1s for all other malicious users. For the honest users, U_t reports extra 1s (for non-neighbors) in the hopes of further increasing their degree estimate. The exact mechanism depends on whether it is response poisoning or input poisoning and is detailed in App. D.3.

For degree deflation, we consider the worst-case scenario where m of the neighbors of the honest user U_t act maliciously. The malicious neighbors always report 0 for their edges to U_t (see App. D.3).



(a) *Syn*: Degree Inflation Attack



(b) *FB*: Degree Deflation Attack

Figure 4.6. Robustness analysis with varying ε.

Hybrid. For degree inflation, the non-target malicious users report their edges using the same strategy as in *RRCheck*. For $\tilde{d}_i^{lap}$, they send their true degree estimates since their degrees are not the targets. Similarly, U_t uses the same strategy as in *RRCheck* for reporting their edges. For $\tilde{d}_t^{lap}$, U_t reports an inflated value based on the reported edges and the threshold τ (see App. D.3).

For degree deflation, we consider the worst-case scenario where m of the neighbors of U_t act maliciously. The malicious users behave as they did in *RRCheck* and report their true degrees for $\tilde{d}_i^{lap}$, as these are not the targeted degrees.

SimpleRR. For degree inflation, we consider the worst-case scenario where the target malicious user U_t is responsible reporting all their edges, and chooses to reports all 1s.

For degree deflation, we again consider the worst case scenario where the malicious users are responsible for reporting the edges to U_t, and they report 0s.

Configuration. For degree inflation, we report the maximum error over all honest users (correctness, α_1) and the error of the malicious target (soundness, α_2). For degree deflation, we report the error of the honest target (correctness) and the maximum error over all the malicious users (soundness). We run each experiment 50 times and report the mean. We use $\delta = 10^{-6}$ and $c = 0.9$ for *Hybrid*. Additional details are in App. D.3.

4.9.2 Results

We consider two values of m (number of malicious users): $m = 40$ and $m = 1500$. The former corresponds to $= 1\%$ malicious users which represents a realistic threat model. We also consider $m = 1500$ to showcase the efficacy of our protocols even with a large number of malicious users ($= 37.5\%$).

Degree Inflation. We report our results for degree inflation in Figs. 4.4a and 4.4b. In terms of correctness, we observe that *Hybrid* performs the best — this is expected since the error introduced by the Laplace mechanism of $\tilde{O}(\frac{1}{\varepsilon})$ is the lowest. Both *RRCheck* and *SimpleRR* have higher honest error of $\tilde{O}(\frac{\sqrt{n}}{\varepsilon})$ due to the underlying randomized response mechanisms. *RRCheck* has better correctness than *SimpleRR*, particularly for *FB* with low privacy ($\varepsilon = 3$). This is because the degree estimator for *SimpleRR* is based on the random variable $count^1$ (Sec. 4.4.2) which has a variance $n\rho(1-\rho)$. However, the degree estimator for *RRCheck* is based on the random variable $count^{11}$ (Sec. 4.5.1). The variance of $count^{11}$ is a function of the graph degree distribution and is lower than that of $count^1$ for sparse graphs and high ε. In terms of soundness, both our proposed protocols perform better than the baseline – *SimpleRR* has $43\times$ and $60\times$ higher malicious error than *Hybrid* and *RRCheck*, respectively, for *FB* for $\varepsilon = 3, m = 40$ for input poisoning. Note that *RRCheck* performs slightly better than *Hybrid*. This is because, although both the protocols have the same asymptotic soundness guarantee, the constants are higher for *Hybrid*(see Sec. 4.6). Finally, we observe that response poisoning is worse than input poisoning for all the protocols for soundness, which is to be expected and is consistent with the theory (response poisoning has an extra $O(\frac{m}{\varepsilon})$ term). For instance, for *RRCheck*, response poisoning is $4.2\times$ worse than input poisoning for *Syn* with $m = 1500, \varepsilon = 0.7$. As expected, the separation becomes less prominent with higher privacy and lower m, as it is harder to pull off strong attacks for these regimes.

Figs. 4.4a and 4.4b also mark the degree of 95^{th} percentile node (d_{95}) for the respective graphs. The way to interpret this is as follows. If an error of a protocol falls outside the box, then any malicious user can inflate their degree estimate to be in the $> d_{95}$ percentile by staging a poisoning attack. Our protocols are better performant for the dense graph *Syn* (attacks are prevented for both values of ε). This is because of the $\tilde{O}(\frac{\sqrt{n}}{\varepsilon})$ term in the soundness guarantee – this term dominates the malicious error for sparse graphs.

Finally, for both datasets for $m = 40$ and $\varepsilon = 0.7$, our protocols flag the malicious target user (returning $\perp$) for $30\% - 70\%$ of the trials. We indicate this with an "X" in the figures. This indicates that our proposed protocols are able to detect malicious users, thereby disincentivizing malicious activity.

Degree Deflation. Figs. 4.5a and 4.5b show the results for the degree deflation attacks on *FB* and *Syn*, respectively. We observe that *Hybrid* performs the best in terms of both correctness and soundness. For instance, it has $58\times$ and $20\times$ better correctness than *SimpleRR* and *RRCheck*, respectively, for *FB* and response poisoning with $\varepsilon = 0.7, m = 1500$. *RRCheck* and *SimpleRR* have similar correctness. This is because the correctness guarantees for both protocols are asymptotically the same. The soundness error for these two protocols are from just the underlying randomized response mechanisms. We observe that *RRCheck* has better soundness than *SimpleRR* for *FB* for $\varepsilon = 3$ because of a better degree estimator as described previously. Finally, we observe that the response poisoning is more damaging than input poisoning for *SimpleRR* and *RRCheck*. For instance, response poisoning has $2\times$ worse correctness than input poisoning for *RRCheck* for *Syn* with $\varepsilon = 0.7, m = 1500$. However, the two types of poisoning have a similar impact on the correctness of *Hybrid*, as the final degree estimate sent by the target user via the estimate from the Laplace mechanism is not poisoned.

We note that none of the degree deflation attacks resulted in the honest user being falsely flagged, demonstrating empirically that our protocols do not have false positives.

We plot the measure $d_{95} - d_{80}$ in Figs. 4.5a and 4.5b where d_k denotes the degree of the k percentile node. The way to interpret this is as follows. If an error falls outside the box, then malicious users can successfully deflate the degree of an honest target from > 95 percentile to < 80 percentile. Based on our results, we observe that *Hybrid* is mostly effective in protecting against this attack even with a large number of malicious users of $m = 1500$.

The effect of ε. Figs. 4.6b and 4.6a show the impact of the attacks with varying privacy parameter ε. We observe that, increasing privacy (lower ε) leads to more skew for all attacks on all three protocols. For instance, the response poisoning version of the degree deflation attack for *FB* is 74$\times$ worse for $\varepsilon = 0.1$ than that for $\varepsilon = 3$ for *RRCheck*. Additionally, we observe that malicious users get flagged only for input poisoning for larger ε. This is because for small values of ε, the thresholds are large due to more noise introduced by privacy which makes it harder to detect malicious users.

Another interesting observation is that even a relatively small number of malicious parties ($m = 1\%$) can stage significantly damaging poisoning attacks. This demonstrates the practical threat of such poisoning attacks. As expected, the impact of the poisoning attacks worsens with increasing m. Nevertheless, our results show that our proposed protocols are able to significantly reduce the impact of poisoning attacks even with a large number of malicious users ($m = 37.5\%$).

4.9.3 Discussion

Here, we discuss our empirical results in the context of our three evaluation questions. First, we observe that the baseline protocol *SimpleRR* performs worse than both our proposed protocols. *Hybrid* offers the best correctness guarantee and its soundness is at least comparable to that of *RRCheck* for all the attacks. Concretely, *Hybrid* can improve the correctness and soundness by up to 25$\times$ and 58$\times$, respectively, as compared to that for *SimpleRR*. Second, we observe that the response poisoning is stronger than input poisoning for all four attacks and all three protocols. Concretely, the response poisoning can be up to $4.2\times$ worse than input poisoning. Finally, we see that the privacy exacerbates the impact of the poisoning attacks – both correctness and soundness gets worse with lower values of ε for all attacks on all three protocols. Concretely, poisoning can worsen by up to $74\times$ for a drop in privacy from $\varepsilon = 3$ to $\varepsilon = 0.7$.

4.10 Related Work

Data Poisoning for LDP. A recent line of work [46, 34, 212, 132] has explored the impact of poisoning attacks in the LDP setting. However, these works focused either on tabular data or key-value data. Additionally, prior work mostly focuses on the task of frequency estimation which is different from our problem of degree estimation. For the former, each user has some item from an input domain and the data aggregator wants to compute the histogram over all the users' items. Whereas, we compute the degree vector $\langle \hat{d}_1, \ldots, \hat{d}_n \rangle$ – each user directly reports their degree d_i (a count or via an adjacency list). More specifically, Cao et al. [34] proposed attacks where an adversary could increase the estimated frequencies for adversary-chosen target items or promote them to be identified as heavy hitters. Wu et al. [212] extended the attacks for key-value data where an adversary aims to simultaneously promote the estimated frequencies and mean values for some adversary-chosen target keys. Both the works also presented some ad-hoc defenses against the proposed attacks. However, their efficacy is heavily dependent on instantiation specific factors such as the data distribution or the percentage of fake users. Cheu et al. [46] formally analyzed the poisoning attacks on categorical data and showed that local algorithms are highly vulnerable to adversarial manipulation – when the privacy level is high or the input domain is large, an adversary who controls a small fraction of the users in the protocol can completely obscure the distribution of the users' inputs. This is essentially an impossibility result for robust estimation of categorical data via non-interactive LDP protocols. Additionally, they showed that poisoning the noisy messages can be far more damaging than poisoning the data itself. A recent work [132] studies the impact of data poisoning for mean and variance estimation for tabular data. In the shuffle DP model, Cheu et al.[48] have studied the impact of poisoning on histogram estimation. In terms of general purpose defenses, prior work has explored strategies strategy based on cryptographically verifying implementations of LDP randomizers [123, 8, 151] – this would restrict the class of poisoning attacks to input poisoning only.

A slew of poisoning attacks [28, 147, 85, 26, 45, 13, 216] on machine learning (ML) models have been proposed in the federated learning setting [115]. A recent line of work [172, 33] explores defense strategies based on zero-knowledge proofs that aim to detect the poisoned inputs. Note that the aforementioned poisoning attacks on ML models are fundamentally different from the ones in our setting. For ML, the users send parameter gradient updates (multi-dimensional real-valued vector) and the attack objective is to misclassify data. On the other hand, our input here is a binary vector/single integral value with an underlying graphical structure and the attack objective is to perturb the degree estimates. Hence, none of the techniques from this literature are directly applicable to our setting.

LDP on Graphs. DP on graphs has been widely studied, with most prior work being in the centralized model [31, 44, 98, 58, 122, 160, 120, 184, 206]. In the local model, there exists work exploring the release of data about graphs in various settings [204, 165, 55, 221, 213, 224]. Recent work exploring private subgraph counting [103, 105] is the most relevant to our setting, as the degree information is the count of an edge, the simplest subgraph. However, no previous work explores poisoning LDP protocols in the graph setting.

4.11 Conclusion

In this paper, we have investigated the impact of poisoning attacks on degree estimation for graphs under LDP. We have presented a formal framework for analyzing the robustness of a protocol against poisoning. Our framework can quantify the impact of a poisoning attack on both honest and malicious users. Additionally, we have proposed novel robust degree estimation protocols under LDP by leveraging the natural data redundancy in graphs, that can significantly reduce the impact of poisoning attacks. Chapter 4, in full, is currently being prepared for submission for publication of the material. Jacob Imola, Amrita Roy Chowdhury, and Kamalika Chaudhuri. "Robustness of Locally Differentially Private Graph Analysis Under Data Poisoning Attacks." The dissertation author was the first author of this paper.

Chapter 5

Differentially Private Hierarchical Clustering with Provable Approximation Guarantees

5.1 Introduction

Hierarchical Clustering is a staple of unsupervised machine learning with more than 60 years of history [207]. Contrary to *flat* clustering methods (such as k-means, [106]), which provide a single partitioning of the data, *hierarchical* clustering algorithms produce a recursive refining of the partitions into increasingly fine-grained clusters. The clustering process can be described by a tree (or dendrogram), and the objective of the tree is to cluster the most similar items in the lowest possible clusters, while separating dissimilar items as high as possible.

The versatility of such methods is apparent from the widespread use of hierarchical clustering in disparate areas of science, such as social networks analysis [130, 139], bioinformatics [61], phylogenetics [183, 107], gene expression analysis [78], text classification [185] and finance [195]. Popular hierarchical clustering methods (such as linkage [106]) are commonly available in standard scientific computing packages [197] as well as large-scale production systems [20, 59].

Despite the fact that many of these applications involve private and sensitive user data, all research on hierarchical clustering (with few exceptions [125, 214] discussed later) has ignored the problem of defining *privacy-preserving* algorithms. In particular, to the best of our knowledge, no work has provided *differentially-private (DP)* [73] algorithms for hierarchical clustering with provable approximation guarantees.

In this work, we seek to address this limitation by advancing the study of differentially-private approximation algorithms for hierarchical clustering under the rigorous optimization framework introduced by [57]. This celebrated framework introduces an objective function for hierarchical clustering (see Section 5.3 for a formal definition) formalizing the goal of clustering similar items lower in the tree.

Our algorithms are edge-level *Differentially Private (DP)* on an input similarity graph, which is relevant when edges of the input graph represents sensitive user information. Designing an edge-level DP algorithm requires proving that the algorithm is insensitive to changes to a single edge of the similarity graph. As we shall see, this is especially challenging for hierarchical clustering. In fact, commonly-used hierarchical clustering algorithms (such as linkage-based ones [106]) are *deterministically* sensitive to a single edge, thus leaking directly the input edges. Moreover, as we show, strong inapproximability bounds exist for Dasgupta's objective under differential privacy, highlighting the technical difficulty of the problem.

Main contributions

First, we show in Section 5.4 that no edge-level ε-DP algorithm (even with exponential time) exists for Dasgupta's objective with less than $O(|V|^2/\varepsilon)$ additive error. This prevents defining private algorithms with meaningful approximation guarantees for *arbitrary* sparse graphs.

Second, on the positive side, we provide the first polynomial time, edge-level approximation algorithm for Dasguta's objective with $O(|V|^{2.5}/\varepsilon)$ additive error and multiplicative error matching that of the best non-private algorithm [4]. This algorithm is based on recent advances in private cut sparsifiers [79]. Moreover, we show an (exponential time) algorithm with $O(|V|^2 \log n/\varepsilon)$ additive error, almost matching the lower bound.

Third, given the strong lower bounds, in Section 5.6 we focus on a popular model of graphs with a planted hierarchical clustering based on the *Stochastic Block Model (SBM)* [53]. For such graphs, we present a private $1+o(1)$ approximation algorithm recovering almost exactly the hierarchy on the blocks. Our algorithm uses, as a black-box, any reconstruction algorithm for the stochastic block model.

Fourth, we introduce a practical and efficient DP SBM community reconstruction algorithm (Section 5.6). This algorithm is based on perturbation theory of graph spectra combined with dimensionality reduction to avoid adding high noise in the Gaussian mechanism. Combined with our clustering algorithm, this results in the first private approximation algorithm for hierarchical clustering in the hierarchical SBM.

Finally, we show in Section 5.7 that this algorithm can be efficiently implemented and works well in practice.

5.2 Related Work

Our work spans the areas of differential privacy, hierarchical clustering and community detection in stochastic block model. For a complete discussion, see Appendix E.1.

Graph algorithms under DP

Differential privacy [72] has recently the gold standard of privacy. We refer to [73] for a survey. Relevant to this work is the area of differential privacy in graphs. Definitions based on edge-level [80, 79] and node-level [122] privacy have been proposed. The most related work is that on graph cut approximation [79, 10], as well as that of private correlation clustering [32, 52].

Hierarchical Clustering

Until recently, most work on hierarchical clustering were heuristic in nature, with the most well-known being the linkage-based ones [106, 20]. [57] introduced a combinatorial objective for hierarchical clustering which we study in this paper. Since this work, many authors have designed algorithms for variants of the problem with no privacy [53, 54, 37, 154, 4, 38].

Limited work has been devoted to DP hierarchical clustering algorithms. One paper [214] initiates private clustering via MCMC methods, which are not guaranteed to be polynomial time. Follow-up work [125] shows that sampling from the Boltzmann distribution (essentially the exponential mechanism [145] in DP) produces an approximation to the maximization version of Dasgupta's function, which is a different problem formulation. Again, this algorithm is not provably polynomial time.

Private flat clustering

Contrary to hierarchical clustering, the area of private *flat* clustering on metric spaces has received large attention. Most work in this area has focused on improving the privacy-approximation trade-off [91, 14] and on efficiency [99, 51, 50].

Stochastic block models

The Stochastic Block Model (SBM) is a classic model for random graphs with planted partitions which has received a significant attention in the literature [95, 150, 149, 87, 63, 135]. For our work, we focus on a variant which has nested ground-truth communities arranged in hierarchical fashion. This model has received attention for hierarchical clustering [53].

The study of private algorithms for SBMs is instead very recent. One of the only results known for private (non-hierarchical) SBMs is the work of [179] which provides quasi-polynomial time community detection algorithms for some regimes of the model. Finally, concurrently to our work, the manuscript of [40] provides strong approximation guarantees using semi-definite programming for recovering SBM communities. Community detection is a distinct problem from hierarchical clustering, and this work is independent of ours.

The connection between existing work in the SBM and ours is that, in Section 5.6, we design a hierarchical clustering algorithm (Algorithm 15) which uses community detection as a black-box. Moreover, we show a novel algorithm for hierarchical SBM community detection (Algorithm 16), independent of [40], which is of practical interest because it uses SVDs, instead of semidefinite programming, and thus does not have a large polynomial run-time.

5.3 Preliminaries

Our results involve the key concepts of hierarchical clustering and differential privacy. We define these two concepts in the next sections.

5.3.1 Hierarchical Clustering

Hierarchical clustering seeks to produce a tree clustering a set V of n items by their similarity. It takes as input an undirected graph $G = (V, E, w)$, where $E \subseteq V \times V$ is the set of edges and $w : V \times V \to \mathbb{R}^+$ is a weight function indicating similarity; i.e. a higher $w(u, v)$ indicates u, v are more similar. We extend the weight function w and say that $w(u, v) = 0$ if $w(u, v) \notin E$.

A hierarchical clustering (HC) of G is a tree T whose leaves are V. The tree can be viewed as a sequence of merges of subtrees of T, with the final merge being the root node. A good hierarchical clustering merges more similar items closer to the bottom of the tree. The cost function $\omega_G(T)$ of Dasgupta [57], captures this intuition. We have

$$\omega_G(T) = \sum_{(u,v)\in V^2} w(u, v)|\text{leaves}(T[u \wedge v])|, \tag{5.1}$$

where $T[u \wedge v]$ indicates the smallest subtree containing u, v in T and $|\text{leaves}(T[u \wedge v])|$ indicates the number of leaves in this subtree. This cost function charges a tree T for each edge based on the similarity $w(u, v)$ and how many leaves are in the subtree in which it is merged.

Additional Notation

We let $\omega_G^* = \min_T \omega_G(T)$ denote the best possible cost attained by any tree T. We write $w(A, B) = \sum_{a\in A, b\in B} w(a, b)$ and we say $w(G) = w(G, G)$. Let $\mathscr{A}(G)$ be a hierarchical clustering algorithm. We say $\mathscr{A}$ is an (a_n, b_n)-approximation if

$$\mathbb{E}[\omega_G(\mathscr{A}(G))] \leq a_n \omega_G^* + b_n, \tag{5.2}$$

where the expectation is over the random coins of $\mathscr{A}$. If an algorithm is a $(a_n, 0)$-approximation algorithm, we often refer to it as simply an a_n-approximation.

5.3.2 Differential Privacy

For hierarchical clustering we use the notion of graph privacy known as edge differential privacy. Intuitively, our private algorithm behaves similarly whether or not the adjacency matrix of G is altered in L_1 distance by up to 1. Specifically, we say $G = (V, E, w)$ and $G' = (V, E', w')$ are *adjacent graphs* if $\sum_{u,v \in V} |w(u,v) - w'(u,v)| \leq 1$, meaning that the adjacency matrices have L_1 distance at most one [1]. This notion has been used before by [79, 31] and it has many real-world applications, such as when the graph is a social network and the edges between users encode relationships between them [80]. The definition of edge-DP is as follows:

Definition 4. *An algorithm $\mathscr{A} : \mathscr{G} \to \mathscr{Y}$ satisfies (ε, δ)-edge DP if, for any $G = (V, E, w), G' = (V, E', w')$ that are adjacent, and any set of trees $\mathscr{T}$,*

$$\Pr[\mathscr{A}(G') \in \mathscr{T}] \leq e^{\varepsilon} \Pr[\mathscr{A}(G) \in \mathscr{T}] + \delta.$$

Edge DP states that given any output $\mathscr{T}$ of $\mathscr{A}$, it is provably hard to tell whether an adjacent G or G' was used. For 0/1 weighted graphs, Definition 4 is equivalent to standard edge DP for unweighted graphs (c.f. Definition 2.2.1 in [163]).

5.4 Lower Bounds

We show that for the both objective functions considered, there are unavoidable lower bounds on the objective function for any differentially private algorithm. Our theorem applies a packing-style argument [97], in which we construct a large family $\mathscr{F}$ of graphs such that no tree can cluster more than one graph in $\mathscr{F}$ well. However, a DP algorithm $\mathscr{A}$ is forced to place

[1] the constant one may be changed to any constant to match the application, and our results carry over easily.

mass on all trees. This limits its utility as significant mass must be placed on trees which do not cluster the input graphs well. Formally, we prove the following theorem:

Theorem 10. *For any $\varepsilon \leq \frac{1}{20}$ and n sufficiently large, let $\mathscr{A}(G)$ be a hierarchical clustering algorithm which satisfies ε-edge differential privacy. Then, there is a weighted graph G with $\omega_G^* \leq O(\frac{n}{\varepsilon})$ such that*

$$\mathbb{E}[\omega_G(\mathscr{A}(G))] \geq \Omega(\frac{n^2}{\varepsilon}).$$

We prove this theorem in Section 5.4.1; we discuss the implications of the theorem here. Since there exists a graph such that $\omega_G^* \leq O(\frac{n}{\varepsilon})$, yet $\omega_G(\mathscr{A}(G)) \geq \Omega(\frac{n^2}{\varepsilon})$, this means that no differentially private algorithm $\mathscr{A}$ can be a $(O(n^\alpha), O(\frac{n^{2\alpha}}{\varepsilon}))$ approximation to hierarchical clustering for any $\alpha < 1$. It is possible for $\mathscr{A}$ to be a $(1, O(\frac{n^2}{\varepsilon}))$-approximation— in this case, for graphs with W total weight, it easy to see that $\omega_G^* \leq O(nW)$ and can be as small as $O(W)$. Thus, it is necessary for W to be much bigger than $\frac{n}{\varepsilon}$, meaning that G cannot be too sparse.

5.4.1 Proof of Theorem 10

To construct our lower bound, we consider the family of graphs $\mathscr{P}(n,5)$ consisting of $\frac{n}{5}$ cycles of size 5. We observe the following facts:

- Each $G \in \mathscr{P}(n,5)$ has n edges. Thus, any $G_1, G_2 \in \mathscr{P}(n,5)$ differ in at most $2n$ edges.
- For any $G \in \mathscr{P}(n,5)$, any binary tree which splits the graph into its cycles before splitting any edges in the cycles incurs a cost of at most $\frac{n}{5}W_5$, where $W_5 = \omega_{C_5}^* \leq 18$.

It will be convenient to use the following definition:

Definition 5. *For a graph G, a* balanced cut *is partition (A,B) of V such that $\frac{n}{3} \leq |A|, |B| \leq \frac{2n}{3}$.*

Any hierarchical clustering T can be mapped to a balanced cut on G in the following way:

Definition 6. *For a binary tree T whose leaves are V, let the sequence $N_0, N_1, \ldots, N_r$ denote a recursive sequence of internal nodes such that N_0 is the root node, and N_i is child of N_{i-1} with more leaves in its subtree. Finally, N_r is the first node in the sequence with fewer than $\frac{2n}{3}$ leaves in its subtree. Then, the balanced cut (A,B) induced by T is the partition $(leaves(N_r), V \setminus leaves(N_r))$.*

It is easy to see that (A,B) in the above definition is indeed a balanced cut of G, and for any edge (u,v) crossing (A,B), we have $|\text{leaves}(T[u \wedge v])| \geq \frac{2n}{3}$.

Our class $\mathscr{C}$ of graphs is a subset of $\mathscr{P}(n,5)$ for which no tree clusters more than one element of $\mathscr{C}$ well. We characterize a condition for which a tree T definitely does not cluster $G \in \mathscr{P}(n,5)$ well:

Definition 7. *For a binary tree T, let (A,B) be its balanced cut. We say (A,B)* misses *a cycle $C \subseteq G$ if at least one vertex of C lies in A and at least one vertex lies in B.*

Now, we show that if T misses many cycles in its balanced cut, it must incur high cost.

Lemma 1. *For a graph $G \in \mathscr{P}(n,5)$, let T be a HC with balanced cut (A,B), and suppose that B misses at least $\alpha\frac{n}{5}$ of the cycles in G, for $0 < \alpha \leq 1$. Then,*

$$\omega_G(T) \geq \frac{4\alpha}{15} n^2.$$

Proof: From the given information, we have that $w(A,B) \geq 2\alpha\frac{n}{5}$, as a missed cycle implies at least two edges are cut. Thus,

$$\begin{aligned}\omega_G(T) &\geq \sum_{u \in A, v \in B} w(u,v) |\text{leaves}(T[u \wedge v])| \\ &\geq \tfrac{2n}{3} w(A,B) \geq \tfrac{4\alpha}{15} n^2. \quad \square\end{aligned}$$

We generate graphs from $\mathscr{P}(n,5)$ at random, showing that the probability that there exists a balanced cut (A,B) which misses few cycles in both G_1, G_2 is exponentially small.

This will allow us to generate a large family of graphs such that no balanced cut misses few cycles in more than one graph. This results in the following lemma—in the following, let $\mathscr{B}(G,r) = \{T \in \mathscr{T}_n : \omega_G(T) < r\}$.

Lemma 2. *For n sufficiently large, there exists a family $\mathscr{F} \subseteq \mathscr{P}(n,5)$ of size $2^{0.2n}$ such that $\mathscr{B}(G,r) \cap \mathscr{B}(G',r) = \emptyset$ for any $G, G' \in \mathscr{F}$ with $r = \frac{n^2}{400}$.*

The proof of this lemma appears in Appendix E.2. Thus, no tree can cluster more than one of our random graphs well, and we can apply the packing argument to obtain Theorem 10. We prove it as follows.

Proof of Theorem 10: Let $\mathscr{F}$ be the set of graphs guaranteed by Lemma 2. We have $|\mathscr{F}| = 2^{0.2n}$. Let $\mathscr{F}_W$ contain the same graphs of $\mathscr{F}$, but with each edge weighted by a positive integer W satisfying $0.02 \leq \varepsilon W < 0.07$. Each $G, G' \in \mathscr{F}$ differs by up to $2n$ edges, and applying group privacy W times, we have that an algorithm A which satisfies ε-DP satisfies $2nW\varepsilon$-DP on the graphs in $\mathscr{F}_W$.

Now, suppose A satisfies $\mathbb{E}[\mathrm{cost}_G(A(G))] < \frac{W}{800}n^2$ for any $G \in \mathscr{F}_W$. This implies $\Pr[\mathrm{cost}_G(A(G)) \in \mathscr{B}(G, \frac{W}{400}n^2)] \geq \frac{1}{2}$ for all $G \in \mathscr{F}_W$. However, we know these balls are disjoint because of the disjointness property on $\mathscr{F}$. Furthermore, we have that $\Pr[A(G) \in \mathscr{B}(G', \frac{W}{400}n^2)] \geq e^{-2nW\varepsilon}\frac{1}{2} > 2^{-0.2n}$ for all $G' \in \mathscr{F}_W$.

$$\begin{aligned}
1 &\geq \sum_{G' \in \mathscr{F}_W} \Pr[A(G) \in \mathscr{B}(G', \tfrac{W}{400}n^2)] \\
&> 2^{0.2n} 2^{-0.2n} = 1.
\end{aligned}$$

This is a contradiction, and thus the algorithm A must have error higher than $\frac{W}{800}n^2 \geq \Omega(\frac{n^2}{\varepsilon})$ on some graph. $\square$

5.5 Algorithms for Private Hierarchical Clustering

In this section, we design private algorithms for hierarchical clustering which work on any input graph. In Section 5.5.1, we propose a polynomial time $(\alpha, O(\frac{n^{2.5}}{\varepsilon}))$ approximation algorithm, where α is the best approximation ratio of a black-box, *non-private* hierarchical clustering algorithm. Then, in Section 5.5.2, we show that the exponential mechanism is a $(1, O(\frac{n^2 \log n}{\varepsilon}))$-approximation algorithm, implying our lower bound is tight. The proofs of the results in this section appear in Appendix E.3.2

5.5.1 Polynomial-Time Algorithm

Our algorithm makes use of a recent algorithm which releases a sanitized, synthetic graph G' that approximates the cuts in the private graph G [79, 10]. Via post-processing, it is then possible to run a non-private, black-box clustering algorithm. We are able to relate the cost in G' to that of G by reducing the cost $\omega_G(T)$ to a sum of cuts. We start by defining the notion of G' approximating the cuts in G.

Definition 8. *For a given graph $G = (V, E, w)$, we say $G' = (V, E', w')$ is an (α_n, β_n)-approximation to cut queries in G if for all $S \subseteq V$, we have*

$$(1-\alpha_n)w(S,\overline{S}) - \beta_n \min\{|S|, n-|S|\} \leq w'(S,\overline{S}) \leq (1+\alpha_n)w(S,\overline{S}) + \beta_n \min\{|S|, n-|S|\}.$$

As we alluded, earlier work shows that it is possible to release an $(\tilde{O}(\frac{1}{\varepsilon\sqrt{n}}), \tilde{O}(\frac{\sqrt{n}}{\varepsilon}))$-approximation to cut queries while satisfying differential privacy. Using this result, we are able to run any blackbox hierarchical clustering algorithm, and by post-processing, the final clustering T' will still satisfy privacy. Even though T' is computed only viewing G', we are able to relate $\omega_G(T')$ to ω_G^* using the fact that G' approximates the cuts in G, and a decomposition of $\omega_{G'}(T')$

into a sum of cuts. This idea recently appeared in [4], and is a critical component of our theorem. In the end, we obtain the following:

Theorem 11. *Given an* $(a_n, 0)$*-approximation to the cost objective of hierarchical clustering, there exists an* (ε, δ)*-DP algorithm which, with probability at least* 0.8*, is a*

$$((1+o(1))a_n, O(n^{2.5}\frac{\log^2 n \log^2 \frac{1}{\delta}}{\varepsilon}))$$

approximation algorithm to the cost objective.

Plugging in a state-of-the-art, $\sqrt{\log n}$ hierarchical clustering algorithm of [37], we obtain a $((1+o(1))\sqrt{\log n}, \tilde{O}(\frac{n^{2.5}}{\varepsilon}))$-approximation. In a graph with total edge weight W, we have $W \leq \omega_G(T) \leq nW$, and thus an approximation is possible if $W > \frac{n^{1.5}}{\varepsilon}$. This means the graph can have an average degree of $\frac{\sqrt{n}}{\varepsilon}$.

5.5.2 Exponential Mechanism

We consider an algorithm based on the well-known exponential mechanism [145]. This algorithm takes exponential time, but achieves greater performance that is nearly tight with our lower bound (showing that the lower bound can't be improved significantly from an information-theoretic point of view).

The exponential mechanism $M : \mathscr{X} \to \mathscr{Y}$ releases an element from $\mathscr{Y}$ with probability proportional to

$$\Pr[M(X) = Y] \propto e^{\varepsilon u_X(Y)/(2S)},$$

where $u_X(Y)$ is a utility function, and $S = \max_{X,X',Y} |u_X(Y) - u_{X'}(Y)|$ is the sensitivity of the utility function in X. This ubiquitous mechanism satisfies $(\varepsilon, 0)$-DP.

In our setting, we use the utility function $u_G(T) = -\omega_G(T)$. The sensitivity is bounded in the following fact.

Fact 1. *For two adjacent input graphs $G = (V, E, w)$ and $G' = (V, E, w')$, we have for all trees T that $|\omega_G(T) - \omega_{G'}(T)| \leq n$.*

Proof: We can write the difference as as

$$
\begin{aligned}
& |\omega_G(T) - \omega_{G'}(T)| \\
& = \left|\sum_{u,v \in V^2} (w(u,v) - w'(u,v)) |\texttt{leaves}(T[u \wedge v])|\right| \\
& \leq \sum_{u,v \in V^2} |w(u,v) - w'(u,v))| \cdot |\texttt{leaves}(T[u \wedge v])| \\
& \leq n \sum_{u,v \in V^2} |w(u,v) - w'(u,v)| \leq n. \quad \square
\end{aligned}
$$

Having controlled the sensitivity, we can apply utility results for the exponential mechanism.

Lemma 3. *There exists an $(\varepsilon, 0)$-DP, $(1, O(\frac{n^2 \log n}{\varepsilon}))$-approximation algorithm for hierarchical clustering.*

Thus, the exponential mechanism improves on the cost, and shows that private hierarchical clustering can be done on graphs with average degree $O(\frac{n}{\varepsilon})$.

5.6 Private Hierarchical Clustering in the Stochastic Block Model

In this section, we propose a hierarchical clustering algorithm designed for input graph generated from the hierarchical stochastic block model (HSBM), a graph model with planted communities arranged in a hierarchical structure. We define this model in Section 5.6.1. Next, in Section 5.6.2, we outline DPHCBlocks, a lightweight private hierarchical clustering algorithm in the HSBM, which uses community detection as a black box. This approach enables any DP community detection algorithm to be used as a sub-routine. Finally, in Section 5.6.3, we propose a practical, private community detection algorithm which is the first to work in the general HSBM. Combining the results in Sections 5.6.2 and 5.6.3, we obtain a private, $1 + o(1)$-approximation algorithm to the Dasgupta cost function.

5.6.1 Hierarchical Stochastic Block Model of Graphs

In this section, we consider unweighted graphs (V,E) where each edge has weight 1. Observe that differential privacy (Definition 4) corresponds to adding or removing an edge from G. In the HSBM [53], there is a partition of V into blocks (communities) $B_1, B_2, \ldots, B_k$ of V with the properties that two items in the same block have the same set of edge probabilities, and that items in different blocks are less likely to be connected with these probabilities following a hierarchical structure.

The probabilities of the edges in B are specified by a tree P with leaves $B = B_1, \ldots, B_k$, internal nodes N, and a function $f : N \cup B \to [0,1]$. To capture the decreasing probability of edges, f must satisfy $f(n_1) < f(n_2)$ whenever n_1 is an ancestor of n_2 in P. Formally, we have [53]:

Definition 9. *Let $B = B_1, \ldots, B_k$; P be a tree with leaves in B and internal nodes N; and $f : N \cup B \to [0,1]$ be a function satisfying that $f(n_1) < f(n_2)$ whenever n_1 is an ancestor of n_2 in P. We refer to the triplet (B,P,f) as a ground-truth tree. Then,* $\mathrm{HSBM}(B,P,f)$ *is a distribution over graphs G whose edges are drawn independently, such that for $u,v \in P$, we have*

$$\Pr[(u,v) \in G] = f(LCA_P(B_u, B_v)),$$

where LCA_P denotes the least common ancestor of the blocks B_u, B_v containing u,v in P.

Due to the randomness of the graph G, it would be unreasonable to expect to be able to recover the exact (B,P,f) from G. Our algorithms will recover an approximate ground-truth tree, according to the following definition:

Definition 10. *(From [53]): Let (B,P,f) be a ground-truth tree, and let (B,T,f') be another ground-truth tree with the same set of blocks. We say (B,T,f') is a γ approximate ground-truth tree if for all $u,v \in B$, $\gamma^{-1} f(LCA_P(u,v)) \leq f'(LCA_{P'}(u,v)) \leq \gamma f(LCA_P(u,v))$.*

For $\gamma \approx 1$, an approximate ground-truth tree means that essentially, $\mathrm{HSBM}(B,P,f)$ and $\mathrm{HSBM}(B,P',f')$ are the same distribution.

5.6.2 Producing a DP Hierarchical Clustering Given Communities

Given the blocks (communities) of an HSBM, we now propose DPHCBlocks, a lightweight, private algorithm for returning a $1+o(1)$-approximation to the Dasgupta cost. Our algorithm uses some ideas from the non-private algorithm proposed in [53, 54].

DPHCBlocks takes in G generated from $\mathrm{HSBM}(B,P,f)$, as well as the blocks B. To produce an approximate ground-truth tree, it considers similarities $sim(B_i,B_j) = \frac{w_G(B_i,B_j)}{|B_i||B_j|}$ for every pair of blocks. It then performs a process similar to single linkage: until all blocks are merged, it greedily merges the groups with the highest similarity, and considers the similarity between this new group and any other groups to be the maximum similarity of any pair of blocks between the groups. Privacy comes from addition of Laplace noise in the similarity calculation, which is the only place in which the private graph G is used. DPHCBlocks appears as Algorithm 15.

DPHCBlocks accesses the graph via the initial similarities $sim(B_i,B_j)$. By observing the sensitivity $\max_{B_i,B_j} |w_{G'}(B_i,B_j) - w_G(B_i,B_j)|$ is at most 1, we are able to prove its privacy. We also use the fact that adding an edge can only affect $sim(B_i,B_j)$ for just one choice of B_i,B_j.

Theorem 12. *DPHCBlocks satisfies ε-edge DP in the parameter G.*

Proof. Observe the algorithm can be viewed as a post-processing of the set $\mathscr{B} = \{sim(B_i,B_j) + \mathscr{L}_{ij} : i,j \in k\}$ where $\mathscr{L}_{ij} \sim Lap(\frac{1}{\varepsilon})$ i.i.d. Suppose an edge is added between B_i,B_j. Then, $sim(B_i,B_j)+\mathscr{L}_{ij}$ is protected by ε-edge DP by the Laplace mechanism, observing the sensitivity of $w_G(B_i,B_j)$ is 1. The other quantities in $\mathscr{B}$ follow the same distribution, so $\mathscr{B}$ itself satisfies ε-edge DP. □

We stress that, crucially, Algorithm 15 and all our algorithms are DP for any input graph G, even if the graphs do not come from the HSBM model. We will use the input distribution assumptions only in the utility proofs.

We are also able to show a utility guarantee that DPHCBlocks is a $(1+o(1),0)$-approximation to the cost objective. In order to prove this, we need to assume that the blocks in the HSBM are sufficiently large (at least $n^{2/3}$) and that the edge probabilities are at least $\frac{\log n}{\sqrt{n}}$. These assumptions are necessary to ensure concentration of the graph cuts between blocks, so that an accurate approximate tree may be formed. Also, it requires that $\varepsilon \geq \frac{1}{\sqrt{n}}$—this is an extremely light assumption, and it still permits us to use a small, constant value of ε to guarantee strong privacy. Formally,

Theorem 13. *For $\varepsilon \geq \frac{1}{\sqrt{n}}$ and a graph G drawn from $\mathrm{HSBM}(B,P,f)$ such that $|B_i| \geq n^{2/3}$ and $f \geq \frac{\log n}{\sqrt{n}}$, with probability $1-\frac{2}{n}$, the tree T outputted by DPHCBlocks satisfies $\omega_G(T) \leq (1+o(1))\omega_G(T')$.*

In fact, we show a stronger result that the tuple (B,T,f') returned by DPHCBlocks is a $1+o(1)$-approximate ground-truth tree for $\mathrm{HSBM}(B,P,f)$. By a result from [54], this implies it achieves the approximation guarantee. We defer the proof to Appendix E.4.1.

Data: $G=(V,E)$ drawn from the HSBM; blocks $B_1,\ldots B_k$ partitioning V, privacy parameter ε
Result: Tree T.
for $i=1$ *to* k **do**
    T_i is a random HC with leaves B_i;
end
$sim(B_i,B_j) \leftarrow \frac{w_G(B_i,B_j)+\mathscr{L}_{ij}}{|B_i||B_j|}$, where $\mathscr{L}_{ij} \sim Lap(\frac{1}{\varepsilon})$;
$\mathscr{C}=\{B_1,\ldots,B_k\}$;
$T=forest(T_1,\ldots,T_k)$;
while $|\mathscr{C}| \geq 1$ **do**
    $A_1,A_2=\arg\max_{A_1,A_2\in\mathscr{C}} sim(A_1,A_2)$;
    Merge A_1,A_2 in T; $C=A_1\cup A_2$;
    $f'(C)=sim(A_1,A_2)$;
    $\mathscr{C}=(\mathscr{C}\setminus\{A_1,A_2\})\cup\{C\}$;
    for $S\in\mathscr{C}\setminus\{C\}$ **do**
        $sim(S,C) \leftarrow \max_{B_i\in S,B_j\in C} sim(B_i,B_j)$;
    end
end
return (B,T,f')

Algorithm 15: DPHCBlocks, a hierarchical clustering algorithm in the HSBM given the blocks.

5.6.3 DP Community Detection in the HSBM

We now develop a DP method of identifying the blocks B of graph drawn from the HSBM. Combined with our clustering algorithm DPHCBlocks, this forms an end-to-end algorithm for hierarchical clustering in the HSBM in which the communities are not known.

In order to describe our algorithm, DPCommunity, we introduce some notation. For a model HSBM(B,P,f), we associate an $n\times n$ expectation matrix A given by the probabilities that edge (i,j) appears in G. We then let $\hat{A}$ be a randomized rounding of A to $\{0,1\}$ which is simply the adjacency matrix of G. DPCommunity recovers communities when they are separated in the sense defined by

$$\Delta = \min_{u\in B_i, v\in B_j : i\neq j} \|A_u - A_v\|_2,$$

where A_u is the uth column of A. Next, we let $\sigma_1(A),\dots,\sigma_n(A)$ denote the singular values of A in order of decreasing magnitude. Finally, we let $\Pi_A^{(k)}$ denote the projection onto the top k left singular values of A—formally, if U_k consists of the top k singular values of A, then $\Pi_A^{(k)} = U_k U_k^T$.

DPCommunity is given the adjacency matrix $\hat{A}$ of a graph drawn from HSBM(B,P,f), as well as k, the number of blocks. In practice, k may be treated as a hyperparameter to be optimized. DPCommunity uses the spectral method [144, 199] to cluster the columns of $\hat{A}$. These results show that the columns in $F = \Pi_{\hat{A}}^{(k)}(\hat{A})$ forms a clustering of the points into their original blocks. To make this private, we use stability results of the SVD to compute (an upper bound of) the sensitivity Γ of F, and add noise N via the Gaussian mechanism. Since N,F are both $n\times n$ matrices, the l_2 error introduced by N grows with $\sqrt{n}$, which is large. Our final observation is that, since the distances in F are all that matter, we may project F to $\log(n)$-dimensional space using Johnson-Lindenstrauss [108], and then add Gaussian noise whose error grows with $\sqrt{\log n}$. DPCommunity is shown in Algorithm 16.

There are two important remarks about DPCommunity. First, to ensure an accurate, private upper bound on Γ, we need the mild assumption that the spectral gap $\sigma_k(\hat{A}) - \sigma_{k+1}(\hat{A})$ is not too small, and if it is, the algorithm returns $\perp$. For most choices of parameters in the SBM, the spectral gap is always much larger than needed—the check is only to ensure privacy even for input graphs not from the SBM. Second, due to ease of theoretical analysis, $\hat{A}$ is split into two parts, and one part is projected onto the top k singular vectors of the other. This removes probabilistic dependence between variables, but the high level ideas are the same.

Data: $\hat{A}$, adjacency matrix generated from $\mathrm{HSBM}(B,P,f)$, privacy parameter ε.
Result: f_z, an estimate of blocks on a set $Z_2 \subseteq V$.
Compute a random partition $Y \sqcup Z_1 \sqcup Z_2$ of V such that $|Y| = \frac{n}{2}$, $|Z_1| = |Z_2| = \frac{n}{4}$;
$\tilde{A}_1 \leftarrow \tilde{A}_{YZ_1}$ (submatrix of $\hat{A}$ with rows Y, cols. Z_1);
$\tilde{A}_2 \leftarrow \tilde{A}_{YZ_2}$;
$\tilde{d}_k \leftarrow \sigma_k(\hat{A}_1) - \sigma_{k+1}(\hat{A}_1) - \frac{8}{\varepsilon}\ln\frac{4}{\delta} + Lap(\frac{8}{\varepsilon})$;
$\tilde{\sigma}_1 \leftarrow \sigma_1(\hat{A}_2) + \frac{4}{\varepsilon}\ln\frac{4}{\delta} + Lap(\frac{4}{\varepsilon})$;
if $\hat{d}_k \leq 10(\frac{8}{\varepsilon}\ln\frac{4}{\delta})$ **then**
  return $\perp$
else
  $\tilde{\Gamma} \leftarrow \frac{\tilde{\sigma}_1}{\hat{d}_k}, m \leftarrow 64\ln\frac{2n}{\delta}$.
$F \leftarrow P\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2)$, where $P \sim \mathcal{N}(0, \frac{1}{\sqrt{m}})^{m\times n/2}$;
$\tilde{F} \leftarrow F + N$, where $N \sim \frac{3k\tilde{\Gamma}}{\varepsilon}\sqrt{2\ln\frac{5}{\delta}}\mathcal{N}(0,1)^{m\times n/4}$;
return $\hat{F}$

Algorithm 16: DPCommunity, a community recovery algorithm.

We now analyze privacy and utility. Full proofs of the results in this section appear in Appendix 5.6. Our privacy analysis involves analyzing the release of the singular values $\sigma_1, \sigma_k, \sigma_{k+1}$, and $\tilde{F}$. The bulk of this analysis comes from analyzing the sensitivity of $\tilde{F}$, which uses the accuracy of the Johnson-Lindenstrauss transform and spectral perturbation bounds.

Theorem 14. *(Privacy): For $\varepsilon < 1$, Algorithm 16 satisfies (ε, δ)-DP with respect to a change of one edge in $\hat{A}$.*

To prove the utility of DPCommunity, we prove that recovery is possible provided that Δ is larger than some threshold depending on ε, the singular values of A, the minimum edge probability, and the minimum block size, along with other mild assumptions on k and the block sizes. These assumptions are necessary, as there will be too little data for concentration otherwise. Formally,

Theorem 15. *(Utility): Let $\hat{A}$ be drawn from* $\mathrm{HSBM}(B,P,f)$*,* $\tau = \max f(x)$*, and* $s = \min_{i=1}^{k} |B_i|$. *There is a universal constant C such that if $\tau \geq C\frac{\log n}{n}$, $s \geq C\sqrt{n\log n}$, $k < n^{1/4}$, $\delta < \frac{1}{n}$, $\sigma_k(A) \geq C\max\{\sqrt{n\tau}, \frac{1}{\varepsilon}\ln\frac{4}{\delta}\}$, and*

$$\Delta > C\max\left\{\frac{k(\ln\frac{1}{\delta})^{3/2}}{\varepsilon}\frac{\sigma_1(A)}{\sigma_k(A)}, \sqrt{\frac{n\tau}{s}} + \sqrt{k\tau\log n} + \frac{\sqrt{nk\tau}}{\sigma_k}\right\},$$

then with probability at least $1-3n^{-1}$, DPCommunity returns a set of points $\tilde{F} = \{f_i : i \in Z_2\}$ such that

$$\|f_i - f_j\|_2 \leq \tfrac{2\Delta}{5} \quad \textit{if } \exists u.\ i,j \in B_u$$
$$\|f_i - f_j\|_2 \geq \tfrac{4\Delta}{5} \quad \textit{otherwise.}$$

Thus, if the assumptions are met, then $\tilde{F}$ consists of k well-separated clusters which indicate the communities of each point in the sampled set $Z_2 \subset V$. These communities can be found using a simple routine such as k-centers. In order to cluster all of V, we can simply divide the privacy budget into $\log n$ parts, run DPCommunity $\log n$ times, and merge the clusters.

To illustrate our theorem in a simple example, consider the HSBM with k equal-sized blocks, and let $f_P(n) = p$ when n is a parent of a leaf in P, and $f_P(n) = q$ otherwise, with $p \geq q$. This corresponds to probability p of an edge within a block and probability q of an edge between any two blocks. In this case, we obtain the following.

Corollary 1. *In the above HSBM, DPCommunity recovers the exact communities when $\delta \leq \frac{1}{n}$, $k < n^{1/4}$, and $\sqrt{p} - \sqrt{q} \geq \Omega(\frac{k\ln\frac{1}{\delta}}{\sqrt{\varepsilon}n^{1/4}})$.*

Compared to previous work in the SBM with privacy, our algorithm requires a larger assumption on $\sqrt{p} - \sqrt{q}$ ([179, 40] require $\sqrt{p} - \sqrt{q} \geq \sqrt{\frac{k}{\varepsilon n}}$). However, previous work either uses semi-definite programming or does not run in polynomial time, whereas DPCommunity is a practical use of the significantly more efficient Singular Vector Decomposition. Furthermore, our algorithm works in the fully-general HSBM, whereas previous work has no analogue of

Theorem 15.

Data: $\hat{A}$, adjacency matrix generated from $\mathrm{HSBM}(B,P,f)$, number of blocks k, privacy parameter ε.
Result: An hierarchical clustering T of $\hat{A}$.

for $i \in \{1,\ldots,\log n\}$ **do**
  $\hat{F} \leftarrow \mathsf{DPCommunity}(\hat{A}, \frac{\varepsilon}{2\log n})$;
  $B_1^i,\ldots,B_k^i \leftarrow \mathsf{k\text{-}centers}(\hat{F},k)$;
end
$B_1,\ldots,B_k \leftarrow \mathsf{Union\text{-}Find}(B_1^1,\ldots,B_k^1,\ldots,B_k^{\log n})$;
$T \leftarrow \mathsf{DPHCBlocks}(\hat{A},\{B_1,\ldots,B_k\},\frac{\varepsilon}{2})$;
return T

Algorithm 17: DPClusterHSBM a hierarchical clustering algorithm in the HSBM.

Combining Theorems 13 and 15, we are able to obtain DPClusterHSBM, an end-to-end hierarchical clustering algorithm in the HSBM (Algorithm 17). This algorithm runs DPCommunity $\log n$ times, using k-centers each run to find the well-separated communities in the subset $Z_2 \subseteq V$ returned by DPCommunity. Running $\log n$ times ensures that with high probability, each point in V will participate in at least one Z_2; these clusters may then be merged using a union-find data structure.

Corollary 2. *Let $\hat{A}$ be drawn from $\mathrm{HSBM}(B,P,f)$, and let $\tau = \max f(x)$ and $s = \min_{i=1}^k |B_i|$. Then, if $\varepsilon > \frac{1}{\sqrt{n}}$, $\delta < \frac{1}{n}$, $s \geq n^{3/4}$, $f \geq \frac{\log n}{\sqrt{n}}$, and the parameters s,τ,A,Δ satisfy the conditions of Theorem 15, then DPClusterHSBM satisfies (ε,δ)-edge DP and is a $1+o(1)$ approximation to the Dasgupta cost.*

Corollary 2 gives a $1+o(1)$ multiplicative approximation the the Dasgupta cost for the given parameter regimes of the HSBM. This is a nearly-optimal cost that avoids the additive error of the algorithms in Section 5.5.

5.7 Experiments

The purpose of this section is evaluate Algorithm 15 designed for the HSBM model. First, we outline our methods and then we discuss our results.

Experimental Setup

We tested our clustering algorithms on a real-world graph and generated synthetic graphs from the HSBM model. We compared the performance of DPClusterHSBM to several baseline algorithms. We ran algorithms at $\varepsilon \in \{0.5, 1.0, 2.0\}$, as well as with no privacy.

To enable the replication of our work, we make the code available **open-source** [2].

Datasets

Our real-world graph was generated from the MNIST digits dataset [128] (with 1797 digits) by, for each digit, adding an undirected edge corresponding to one of its 120 nearest neighbors in pixel space. We generated graphs from $\text{HSBM}(B, P, f)$ with $n = 2048$ nodes, $k = \{4, 8\}$ blocks, with block sizes chosen proportional to $\{1, \gamma, \dots, \gamma^{k-1}\}$, where $\gamma^{k-1} = 3$. This has the effect of creating differently-sized blocks. We selected P to be a balanced tree over the blocks, and f that increases uniformly in the interval $[0.1, 0.9]$ as the tree is descended.

Algorithms

We ran DPClusterHSBM and several baseline algorithms. In the implementation of DPClusterHSBM, we used a modified version of DPCommunity for practical considerations. This does not affect the privacy guarantees but it simplifies the algorithm. In particular, we privately release $\tilde{A}_1$ using the Laplace mechanism, and compute $\Pi_{\tilde{A}_1}(\hat{A}_2)$ without projection. We are then able to add Gaussian noise tailored to the sensitivity of $\Pi_{\tilde{A}_1}$, rather than to Γ which proved to be a rough upper bound in practice.

For our baselines, we considered a naive private approach in which we release A using the Laplace mechanism and truncate these values to be non-negative to form a sanitized, weighted graph. Then, we ran single, complete, and average linkage, and recorded the best of these methods. We refer collectively to these baselines as Linkage. Second, we formed a tree by recursively partitioning the graph into its (approximately) sparsest cut. As shown in [37], this is an $O(\sqrt{\log n}, 0)$-approximation in the *sanitized* graph. We refer to this baseline as SparseCut.

[2] https://bitbucket.org/jjimola/dphc/src/master/

Metrics

For each graph and clustering algorithm, and the value of ε, we computed $\omega_G(T)$, averaged over 5 runs.

5.7.1 Results

Our results appear in Figure 5.1. In addition to the cost for each algorithm, we included the cost of a random tree. The data had low variance: for each of the 5 runs used to compute each bar, the values were within 0.5% of each other.

For all trials, the cost of Linkage was much higher than the other two algorithms; even with $\varepsilon = 2$, Linkage did not offer improvement of more than 10% reduction in cost over the random tree. Thus, the rest of our discussion focuses on DPClusterHSBM and SparseCut.

For the synthetic graphs, the cost of DPClusterHSBM is lower than SparseCut, particularly when $\varepsilon = 0.5$. In this case, when $k = 4$ (resp. 8), DPClusterHSBM offered a 14.4% (resp. 14.2%) reduction in cost over the random tree, whereas SparseCut offered an 11.5% (resp. 10.3%) reduction. Thus, DPClusterHSBM offers up to 38% more reduction in cost than SparseCut, over the cost of a random tree. Even when $\varepsilon = 0.5$, the cost of DPClusterHSBM is just 5.8% (resp. 9.6%) higher than the cost of the best tree with no privacy.

For $\varepsilon = 1, 2$ on HSBM graphs, the costs of SparseCut and DPClusterHSBM fall to within 1% of each other, though DPClusterHSBM consistently outperforms the former for all values of ε. Moreover, notice that for $\varepsilon = 2$, the costs of both algorithms are within 1% of the non-private tree, indicating that for higher ε the cost of privacy becomes negligible.

For the graph generated from MNIST, all algorithms perform as poorly as a random tree for $\varepsilon = 0.5$. This indicates that the noise introduced by the high privacy constraint destroys the clusters, which are less-well structured than those of the HSBM graphs. At $\varepsilon = 1$, the error of SparseCut is 10% higher than DPClusterHSBM. For $\varepsilon = 2$, the cost of SparseCut is 5% higher than that of DPClusterHSBM, and DPClusterHSBM attains error within 3% of the best tree with no privacy. This is consistent with our previous observation that DPClusterHSBM offers

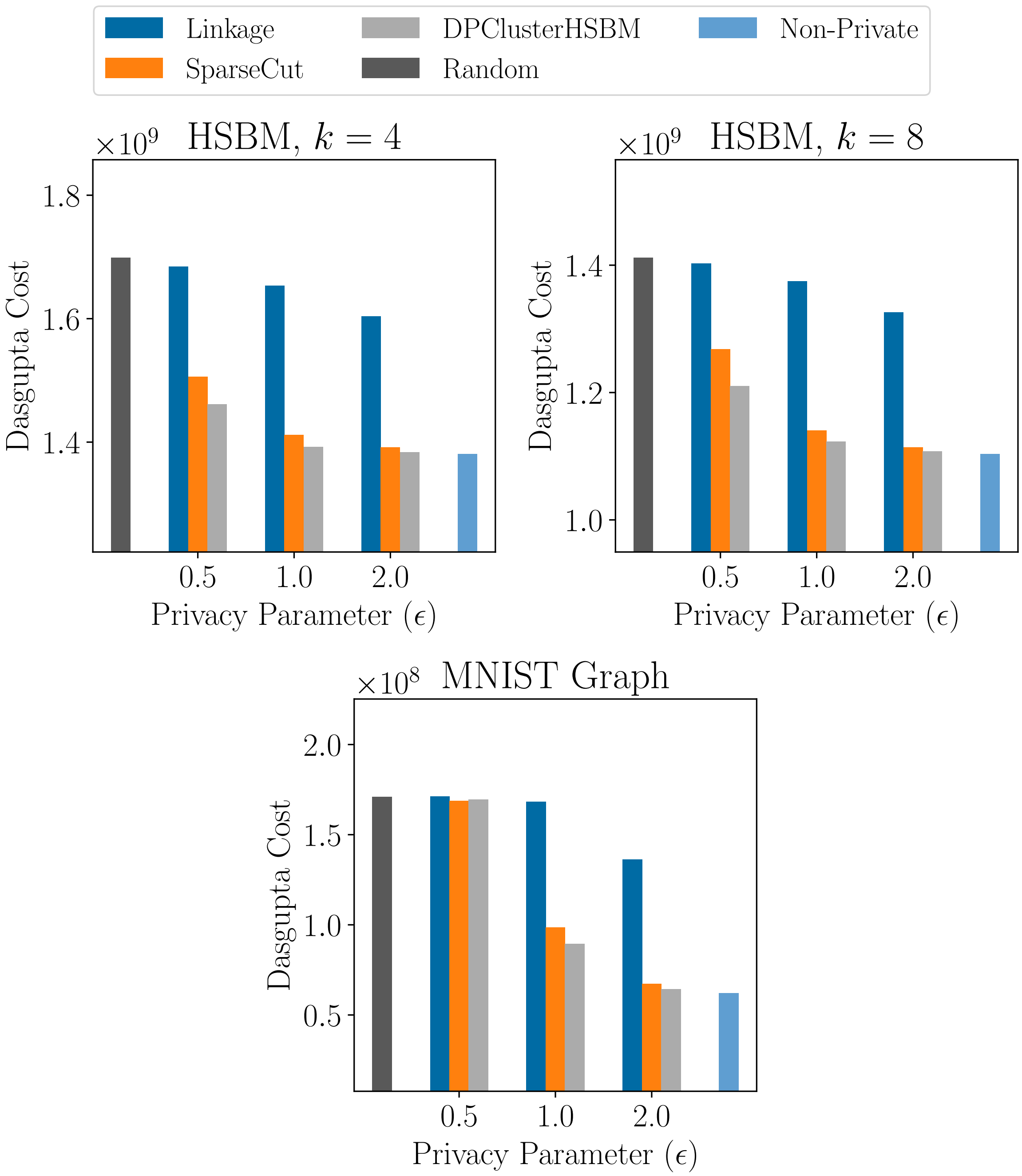


Figure 5.1. Cost for HSBM graphs with 2048 nodes and k clusters and MNIST graph with 1797 nodes.

improvement over the baselines, particularly when ε is not too high.

5.8 Conclusion

We have considered hierarchical clustering under differential privacy in Dasgupta's cost framework. While strong lower bounds exist for the problem, we have proposed algorithms with nearly matching approximation guarantees. Furthermore, we showed the lower bounds can be overcome in the HSBM, and nearly optimal trees can be found in this setting using efficient methods. For future work, one could consider private hierarchical clustering in a less structured model than the HSBM in hopes of overcoming the lower bound here as well.

Chapter 5, in full, is currently being prepared for submission for publication of the material. Jacob Imola, Alessandro Epasto, Mohammad Mahdian, Vincent Cohen-Addad, and Vahab Mirrokni. "Differentially Private Hierarchical Clustering with Provable Approximation Guarantees." The dissertation author was the first author of this paper.

Appendix A

A.1 Effectiveness of empirical estimation in $\mathsf{LocalRR}_\triangle$

In Section 1.4.2, 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 [165, 218].

As the RR without empirical estimation, we applied the RR to the lower triangular part of the adjacency matrix **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 A.1 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 from 0.1 to 2. The experimental set-up is the same as Section 1.5.1. Figure A.1 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 1.5, the l_2 loss of $\mathsf{LocalRR}_\triangle$ is also significantly reduced by an additional round of interaction.

A.2 Experiments on Barabási-Albert Graphs

Experimental set-up. In Section 1.5, 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 [16] 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 [96], 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 $\tilde{d}_{max} = d_{max}$.

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 1.5, 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 A.2 shows the results. Overall, Figure A.2 has a similar tendency to Figures 1.5, 1.6, and 1.7. For example, $\mathsf{Local2Rounds}_\triangle$ significantly outperforms $\mathsf{LocalRR}_\triangle$, especially when the graph G is sparse; i.e., $\lambda = 10$. In $\mathsf{Local2Rounds}_\triangle$, $\mathsf{CentralLap}\triangle$, $\mathsf{LocalLap}_2\star$, and $\mathsf{CentralLap}_2\star$, 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 A.2 also shows that the l_2 loss is roughly consistent with our upper-bounds in Section 1.4. For example, recall that $\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, $\mathsf{CentralLap}_\triangle$, $\mathsf{LocalLap}_{2\star}$, and $\mathsf{CentralLap}_{2\star}$ 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 A.2 are roughly consistent with these upper-bounds. In addition, the right panels of Figure A.2 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 $\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, and $\mathsf{CentralLap}_\triangle$, respectively. They are roughly consistent with our upper-bounds – for small ε, the expected l_2 loss of $\mathsf{LocalRR}_\triangle$, $\mathsf{Local2Rounds}_\triangle$, and $\mathsf{CentralLap}_\triangle$ is $O(\varepsilon^{-6})$, $O(\varepsilon^{-4})$, and $O(\varepsilon^{-2})$, respectively.

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

A.3 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 A.3 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 $\mathscr{A} = \{(V, E' \cup N : N \subseteq M\}$. 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 $\mathscr{A}$ will remove at least $\frac{d_{max}}{2} - 2$ triangles. This implies that $\mathscr{A}$ is an $(n, \frac{d_{max}}{2} - 2)$ independent cube for $f_\triangle$.

A.4 Proof of Statements in Section 1.4

Here we prove the statements in Section 1.4. Our proofs will repeatedly use the well-known bias-variance decomposition [156], which we briefly explain below. We denote the variance of the random variable X by $\mathbb{V}[X]$. If we are producing a private, randomized estimate $\hat{f}(G)$ of the graph function $f(G)$, then the expected l_2 loss (over the randomness in the algorithm) can be written as:

$$\mathbb{E}[l_2^2(\hat{f}(G), f(G))] = \left(\mathbb{E}[\hat{f}(G)] - f(G)\right)^2 + \mathbb{V}[\hat{f}(G)]. \tag{A.1}$$

The first term is the bias, and the second term is the variance. If the estimate is unbiased (i.e., $\mathbb{E}[\hat{f}(G)] = f(G)$), then the expected l_2 loss is equal to the variance.

A.4.1 Proof of Theorem 1

Let $\mathscr{R}_i$ be $\mathsf{LocalLap}_{k\star}$. Let $d_i, d_i' \in \mathbb{Z}_{\geq 0}$ be the number of "1"s in two neighbor lists $\mathbf{a}_i, \mathbf{a}_i' \in \{0,1\}^n$ that differ in one bit. Let $r_i = \binom{d_i}{k}$ and $r_i' = \binom{d_i'}{k}$. Below we consider two cases about d_i: when $d_i < \tilde{d}_{max}$ and when $d_i \geq \tilde{d}_{max}$.

Case 1: $d_i < \tilde{d}_{max}$. In this case, both $\mathbf{a}_i$ and $\mathbf{a}_i'$ do not change after graph projection, as $d_i' \leq d_i + 1 \leq \tilde{d}_{max}$. Then we obtain:

$$\begin{aligned}
\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{r}_i] &= \exp\left(-\frac{\varepsilon|\hat{r}_i - r_i|}{\Delta}\right) \\
\Pr[\mathscr{R}_i(\mathbf{a}_i') = \hat{r}_i] &= \exp\left(-\frac{\varepsilon|\hat{r}_i - r_i'|}{\Delta}\right),
\end{aligned}$$

where $\Delta = \binom{\tilde{d}_{max}}{k-1}$. Therefore,

$$\begin{aligned}
\frac{\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{r}_i]}{\Pr[\mathscr{R}_i(\mathbf{a}_i') = \hat{r}_i]} &= \exp\left(\frac{\varepsilon|\hat{r}_i - r_i'|}{\Delta} - \frac{\varepsilon|\hat{r}_i - r_i|}{\Delta}\right) \\
&\leq \exp\left(\frac{\varepsilon|r_i' - r_i|}{\Delta}\right) \qquad \text{(A.2)} \\
&\quad \text{(by the triangle inequality).}
\end{aligned}$$

If $d_i' = d_i + 1$, then $|r_i' - r_i|$ in (A.2) can be written as follows:

$$|r_i' - r_i| = \binom{d_i+1}{k} - \binom{d_i}{k} = \binom{d_i}{k-1} < \binom{\tilde{d}_{max}}{k-1} = \Delta,$$

Since we add $\mathrm{Lap}(\frac{\Delta}{\varepsilon})$ to r_i, we obtain:

$$\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{r}_i] \leq e^{\varepsilon} \Pr[\mathscr{R}_i(\mathbf{a}_i') = \hat{r}_i]. \tag{A.3}$$

If $d_i' = d_i - 1$, then $|r_i' - r_i| = \binom{d_i}{k} - \binom{d_i-1}{k} = \binom{d_i-1}{k-1} < \Delta$ and (A.3) holds. Therefore, $\mathsf{LocalLap}_{k\star}$ provides ε-edge LDP.

Case 2: $d_i \geq \tilde{d}_{max}$**.** Assume that $d_i' = d_i + 1$. In this case, $d_i' > \tilde{d}_{max}$. Therefore, d_i' becomes $\tilde{d}_{max}$ after graph projection. In addition, d_i also becomes $\tilde{d}_{max}$ after graph projection. Therefore, we obtain $d_i = d_i' = \tilde{d}_{max}$ after graph projection. Thus $\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{r}_i] = \Pr[\mathscr{R}_i(\mathbf{a}_i') = \hat{r}_i]$.

Assume that $d_i' = d_i - 1$. If $d_i > \tilde{d}_{max}$, then $d_i = d_i' = \tilde{d}_{max}$ after graph projection. Thus $\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{r}_i] = \Pr[\mathscr{R}_i(\mathbf{a}_i') = \hat{r}_i]$. If $d_i = \tilde{d}_{max}$, then (A.3) holds. Therefore, $\mathsf{LocalLap}_{k\star}$ provides ε-edge LDP. □

A.4.2 Proof of Theorem 2

Assuming the maximum degree d_{max} of G is at most $\tilde{d}_{max}$, the only randomness in the algorithm will be the Laplace noise since graph projection will not occur. Since the Laplacian noise $\mathrm{Lap}(\frac{\Delta}{\varepsilon})$ has mean 0, the estimate $\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max})$ is unbiased. Then by the bias-variance decomposition [156], the expected l_2 loss $\mathbb{E}[l_2^2(\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max}), f_{k\star}(G))]$ is equal to the variance of $\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max})$. The variance of $\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max})$ can be written as follows:

$$\begin{aligned}\mathbb{V}[\hat{f}_{k\star}(G, \varepsilon, \tilde{d}_{max})] &= \mathbb{V}\left[\sum_{i=1}^{n} \mathrm{Lap}\left(\frac{\Delta}{\varepsilon}\right)\right] \\ &= \frac{n\Delta^2}{\varepsilon^2}.\end{aligned}$$

Since $\Delta = \binom{\tilde{d}_{max}}{k-1} = O(\tilde{d}_{max}^{k-1})$, we obtain:

$$\begin{aligned}\mathbb{E}[l_2^2(\hat{f}_{k\star}(G,\varepsilon,\tilde{d}_{max}), f_{k\star}(G))] &= \mathbb{V}[\hat{f}_{k\star}(G,\varepsilon,\tilde{d}_{max})] \\ &= O\left(\frac{n\tilde{d}_{max}^{2k-2}}{\varepsilon^2}\right).\end{aligned}$$

□

A.4.3 Proof of Proposition 2

Let $\mu = e^{\varepsilon}$ and $\mathbf{Q} \in [0,1]^{4\times 4}$ be a 4×4 matrix such that:

$$\mathbf{Q} = \frac{1}{(\mu+1)^3}\begin{pmatrix} \mu^3 & 3\mu^2 & 3\mu & 1 \\ \mu^2 & \mu^3+2\mu & 2\mu^2+1 & \mu \\ \mu & 2\mu^2+1 & \mu^3+2\mu & \mu^2 \\ 1 & 3\mu & 3\mu^2 & \mu^3 \end{pmatrix}. \tag{A.4}$$

Let $c_3, c_2, c_1, c_0 \in \mathbb{Z}_{\geq 0}$ be respectively the number of triangles, 2-edges, 1-edge, and no-edges in G. Then we obtain:

$$(\mathbb{E}[m_3], \mathbb{E}[m_2], \mathbb{E}[m_1], \mathbb{E}[m_0]) = (c_3, c_2, c_1, c_0)\mathbf{Q}. \tag{A.5}$$

In other words, $\mathbf{Q}$ is a transition matrix from a type of subgraph (i.e., triangle, 2-edges, 1-edge, or no-edge) in G to a type of subgraph in G'.

Let $\hat{c}_3, \hat{c}_2, \hat{c}_1, \hat{c}_0 \in \mathbb{R}$ be the empirical estimate of (c_3, c_2, c_1, c_0). By (A.5), they can be written as follows:

$$(\hat{c}_3, \hat{c}_2, \hat{c}_1, \hat{c}_0) = (m_3, m_2, m_1, m_0)\mathbf{Q}^{-1}. \tag{A.6}$$

Let $\mathbf{Q}_{i,j}^{-1}$ be the (i,j)-th element of $\mathbf{Q}^{-1}$. By using Cramer's rule, we obtain:

$$\mathbf{Q}_{1,1}^{-1} = \tfrac{\mu^3}{(\mu-1)^3},\ \mathbf{Q}_{2,1}^{-1} = -\tfrac{\mu^2}{(\mu-1)^3}, \tag{A.7}$$

$$\mathbf{Q}_{3,1}^{-1} = \tfrac{\mu}{(\mu-1)^3},\ \mathbf{Q}_{4,1}^{-1} = -\tfrac{1}{(\mu-1)^3}. \tag{A.8}$$

By (A.6), (A.7), and (A.8), we obtain:

$$\hat{c}_3 = \tfrac{\mu^3}{(\mu-1)^3}m_3 - \tfrac{\mu^2}{(\mu-1)^3}m_2 + \tfrac{\mu}{(\mu-1)^3}m_1 - \tfrac{1}{(\mu-1)^3}m_0.$$

Since $\mu = e^\varepsilon$ and the empirical estimate is unbiased [114, 201], we obtain (1.4) in Proposition 2.

□

A.4.4 Proof of Theorem 3

Since $\mathsf{LocalRR}_\triangle$ applies the RR to the lower triangular part of the adjacency matrix $\mathbf{A}$, it provides ε-edge LDP for $(R_1,\dots,R_n)$. Lines 5 to 8 in Algorithm 2 are post-processing of $(R_1,\dots,R_n)$. Thus, by the immunity to post-processing [73], $\mathsf{LocalRR}_\triangle$ provides ε-edge LDP for the output $\frac{1}{(\mu-1)^3}(\mu^3 m_3 - \mu^2 m_2 + \mu m_1 - m_0)$.

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 $\mathbf{A}$. Therefore, $\mathsf{LocalRR}_\triangle$ provides ε-relationship DP.

A.4.5 Proof of Theorem 4

By Proposition 2, the estimate $\hat{f}_\triangle(G,\varepsilon)$ by $\mathsf{LocalRR}_\triangle$ is unbiased. Then by the bias-variance decomposition [156], the expected l_2 loss $\mathbb{E}[l_2^2(\hat{f}_\triangle(G,\varepsilon), f_\triangle(G))]$ is equal to the variance of $\hat{f}_\triangle(G,\varepsilon)$. Let $a_3 = \frac{\mu^3}{(\mu-1)^3}$, $a_2 = -\frac{\mu^2}{(\mu-1)^3}$, $a_1 = \frac{\mu}{(\mu-1)^3}$, and $a_0 = -\frac{1}{(\mu-1)^3}$. Then the variance of $\hat{f}_{\triangle}(G,\varepsilon)$ can be written as follows:

$$\begin{aligned}\mathbb{V}[\hat{f}_{\triangle}(G,\varepsilon)] &= \mathbb{V}[a_3m_3+a_2m_2+a_1m_1+a_0m_0] \\ &= a_3^2\mathbb{V}[m_3]+a_2^2\mathbb{V}_{RR}[m_2]+a_1^2\mathbb{V}[m_1]+a_0^2\mathbb{V}[m_0] \\ &\quad + \sum_{i=0}^{3}\sum_{j=0,j\neq i}^{3} 2a_ia_j\mathrm{cov}(m_i,m_j), \qquad \text{(A.9)}\end{aligned}$$

where $\mathrm{cov}(m_i,m_j)$ represents the covariance of m_i and m_j. The covariance $\mathrm{cov}(m_i,m_j)$ can be written as follows:

$$\begin{aligned}\mathrm{cov}(m_i,m_j) &\leq \sqrt{\mathbb{V}[m_i]\mathbb{V}[m_j]} \\ &\quad \text{(by Cauchy-Schwarz inequality)} \\ &\leq \max\{\mathbb{V}[m_i],\mathbb{V}[m_j]\} \\ &\leq \mathbb{V}[m_i]+\mathbb{V}[m_j]. \qquad \text{(A.10)}\end{aligned}$$

By (A.9) and (A.10), we obtain:

$$\begin{aligned}&\mathbb{V}[\hat{f}_{\triangle}(G,\varepsilon)] \\ &\leq (a_3^2+4a_3(a_2+a_1+a_0))\mathbb{V}[m_3] \\ &\quad + (a_2^2+4a_2(a_3+a_1+a_0))\mathbb{V}[m_2] \\ &\quad + (a_1^2+4a_1(a_3+a_2+a_0))\mathbb{V}[m_1] \\ &\quad + (a_0^2+4a_0(a_3+a_2+a_1))\mathbb{V}[m_0] \\ &= O\left(\frac{e^{6\varepsilon}}{(e^{\varepsilon}-1)^6}(\mathbb{V}[m_3]+\mathbb{V}[m_2]+\mathbb{V}[m_1]+\mathbb{V}[m_0])\right). \qquad \text{(A.11)}\end{aligned}$$

Below we calculate $\mathbb{V}[m_3]$, $\mathbb{V}[m_2]$, $\mathbb{V}[m_1]$, and $\mathbb{V}[m_0]$ by assuming the Erdös-Rényi model $\mathbf{G}(n,\alpha)$ for G:

Lemma 1. *Let $G \sim \mathbf{G}(n,\alpha)$. Let $p = \frac{1}{e^{\varepsilon}+1}$ and $\beta = \alpha(1-p)+(1-\alpha)p$. Then $\mathbb{V}[m_3] =$*

$O(\beta^5 n^4 + \beta^3 n^3)$, $\mathbb{V}[m_2] = O(\beta^3 n^4 + \beta^2 n^3)$, and $\mathbb{V}[m_1] = \mathbb{V}[m_0] = O(\beta n^4)$.

Before going into the proof of Lemma 1, we prove Theorem 4 using Lemma 1. By (A.11) and Lemma 1, we obtain:

$$\mathbb{V}[\hat{f}_{\triangle}(G,\varepsilon)] = O\left(\frac{e^{6\varepsilon}}{(e^{\varepsilon}-1)^6}\beta n^4\right),$$

which proves Theorem 4. □

We now prove Lemma 1:

Proof of Lemma 1. Fist we show the variance of m_3 and m_0. Then we show the variance of m_2 and m_1.

Variance of m_3 and m_0. Since each edge in the original graph G is independently generated with probability $\alpha \in [0,1]$, each edge in the noisy graph G' is independently generated with probability $\beta = \alpha(1-p) + (1-\alpha)p \in [0,1]$, where $p = \frac{1}{e^{\varepsilon}+1}$. Thus m_3 is the number of triangles in graph $G' \sim \mathbf{G}(n,\beta)$.

For $i,j,k \in [n]$, let $y_{i,j,k} \in \{0,1\}$ be a variable that takes 1 if and only if (v_i, v_j, v_k) forms a triangle. Then $\mathbb{E}[m_3^2]$ can be written as follows:

$$\mathbb{E}[m_3^2] = \sum_{i<j<k} \sum_{i'<j'<k'} \mathbb{E}[y_{i,j,k} y_{i',j',k'}] \tag{A.12}$$

$\mathbb{E}[y_{i,j,k} y_{i',j',k'}]$ in (A.12) is the probability that both (v_i, v_j, v_k) and $(v_{i'}, v_{j'}, v_{k'})$ form a triangle. This event can be divided into the following four types:

1. $(i,j,k) = (i',j',k')$. There are $\binom{n}{3}$ such terms in (A.12). For each term, $\mathbb{E}[y_{i,j,k} y_{i',j',k'}] = \beta^3$.
2. (i,j,k) and (i',j',k') have two elements in common. There are $\binom{n}{2}(n-2)(n-3) = 12\binom{n}{4}$ such terms in (A.12). For each term, $\mathbb{E}[y_{i,j,k} y_{i',j',k'}] = \beta^5$.
3. (i,j,k) and (i',j',k') have one element in common. There are $n\binom{n-1}{2}\binom{n-3}{2} = 30\binom{n}{5}$ such terms in (A.12). For each term, $\mathbb{E}[y_{i,j,k} y_{i',j',k'}] = \beta^6$.

4. (i,j,k) and (i',j',k') have no common elements. There are $\binom{n}{3}\binom{n-3}{3} = 20\binom{n}{6}$ such terms in in (A.12). For each term, $\mathbb{E}[y_{i,j,k}y_{i',j',k'}] = \beta^6$.

Moreover, $\mathbb{E}[m_3]^2 = \binom{n}{3}^2\beta^6$. Therefore, the variance of m_3 can be written as follows:

$$
\begin{aligned}
\mathbb{V}[m_3] &= \binom{n}{3}\beta^3 + 12\binom{n}{4}\beta^5 + 30\binom{n}{5}\beta^6 + 20\binom{n}{6}\beta^6 - \binom{n}{3}^2\beta^6 \\
&= \binom{n}{3}\beta^3(1-\beta^3) + 12\binom{n}{4}\beta^5(1-\beta) \\
&= O(\beta^5 n^4 + \beta^3 n^3).
\end{aligned}
$$

By changing β to $1-\beta$ and counting triangles, we get a random variable with the same distribution as m_0. Thus,

$$
\begin{aligned}
\mathbb{V}[m_0] &= \binom{n}{3}(1-\beta)^3(1-(1-\beta)^3) + 12\binom{n}{4}(1-\beta)^5\beta \\
&= O(\beta n^4).
\end{aligned}
$$

Variance of m_2 and m_1. For $i,j,k \in [n]$, let $z_{i,j,k} \in \{0,1\}$ be a variable that takes 1 if and only if (v_i, v_j, v_k) forms 2-edges (i.e., exactly one edge is missing in the three nodes). Then $\mathbb{E}[m_2^2]$ can be written as follows:

$$
\mathbb{E}[m_2^2] = \sum_{i<j<k}\sum_{i'<j'<k'} \mathbb{E}[z_{i,j,k}z_{i',j',k'}] \tag{A.13}
$$

$\mathbb{E}[z_{i,j,k}z_{i',j',k'}]$ in (A.13) is the probability that both (v_i, v_j, v_k) and $(v_{i'}, v_{j'}, v_{k'})$ form 2-edges. This event can be divided into the following four types:

1. $(i,j,k) = (i',j',k')$. There are $\binom{n}{3}$ such terms in (A.13). For each term, $\mathbb{E}[z_{i,j,k}z_{i',j',k'}] = 3\beta^2(1-\beta)$.

2. (i,j,k) and (i',j',k') have two elements in common. There are $\binom{n}{2}(n-2)(n-3) = 12\binom{n}{4}$ such terms in (A.13). For example, consider a term in which $i = i' = 1$, $j = j' = 2$, $k = 3$,

and $k' = 4$. Both (v_1, v_2, v_3) and (v_1, v_2, v_4) form 2-edges if:

(a) $(v_1, v_2), (v_1, v_3), (v_1, v_4) \in E'$, $(v_2, v_3), (v_2, v_4) \notin E'$,

(b) $(v_1, v_2), (v_1, v_3), (v_2, v_4) \in E'$, $(v_2, v_3), (v_1, v_4) \notin E'$,

(c) $(v_1, v_2), (v_2, v_3), (v_1, v_4) \in E'$, $(v_1, v_3), (v_2, v_4) \notin E'$,

(d) $(v_1, v_2), (v_2, v_3), (v_2, v_4) \in E'$, $(v_1, v_3), (v_1, v_4) \notin E'$, or

(e) $(v_1, v_3), (v_1, v_4), (v_2, v_3), (v_2, v_4) \in E'$, $(v_1, v_2) \notin E'$.

Thus, $\mathbb{E}[z_{i,j,k} z_{i',j',k'}] = 4\beta^3(1-\beta)^2 + \beta^4(1-\beta)$ for this term. Similarly, $\mathbb{E}[z_{i,j,k} z_{i',j',k'}] = 4\beta^3(1-\beta)^2 + \beta^4(1-\beta)$ for the other terms.

3. (i, j, k) and (i', j', k') have one element in common. There are $n\binom{n-1}{2}\binom{n-3}{2} = 30\binom{n}{5}$ such terms in (A.13). For each term, $\mathbb{E}[z_{i,j,k} z_{i',j',k'}] = (3\beta^2(1-\beta))^2 = 9\beta^4(1-\beta)^2$.

4. (i, j, k) and (i', j', k') have no common elements. There are $\binom{n}{3}\binom{n-3}{3} = 20\binom{n}{6}$ such terms in (A.13). For each term, $\mathbb{E}[z_{i,j,k} z_{i',j',k'}] = (3\beta^2(1-\beta))^2 = 9\beta^4(1-\beta)^2$.

Moreover, $\mathbb{E}[m_2]^2 = (3\binom{n}{3}\beta^2(1-\beta))^2 = 9\binom{n}{3}^2\beta^4(1-\beta)^2$. Therefore, the variance of m_2 can be written as follows:

$$
\begin{aligned}
\mathbb{V}[m_2] &= \mathbb{E}[m_2^2] - \mathbb{E}[m_2]^2 \\
&= 3\binom{n}{3}\beta^2(1-\beta) + 12\binom{n}{4}\left(4\beta^3(1-\beta)^2 + \beta^4(1-\beta)\right) \\
&\quad +270\binom{n}{5}\beta^4(1-\beta)^2 + 180\binom{n}{6}\beta^4(1-\beta)^2 \\
&\quad -9\binom{n}{3}^2\beta^4(1-\beta)^2.
\end{aligned}
$$

By simple calculations,

$$
270\binom{n}{5} + 180\binom{n}{6} - 9\binom{n}{3}^2 = -108\binom{n}{4} - 9\binom{n}{3}.
$$

Thus we obtain:

$$\begin{aligned}\mathbb{V}[m_2] &= 3\binom{n}{3}\beta^2(1-\beta)\left(1-3\beta^2(1-\beta)\right) \\ &\quad +12\binom{n}{4}\beta^3(1-\beta)\left(4(1-\beta)+\beta-9\beta(1-\beta)\right) \\ &= O(\beta^3 n^4+\beta^2 n^3).\end{aligned}$$

Similarly, the variance of m_1 can be written as follows:

$$\begin{aligned}\mathbb{V}[m_1] &= 3\binom{n}{3}\beta(1-\beta)^2\left(1-3\beta(1-\beta)^2\right) \\ &\quad +12\binom{n}{4}\beta(1-\beta)^3\left(4\beta+(1-\beta)-9\beta(1-\beta)\right) \\ &= O(\beta n^4).\end{aligned}$$

□

A.4.6 Proof of Proposition 3

Let $t_* = \sum_{i=1}^n t_i$ and $s_* = \sum_{i=1}^n s_i$. Let $s_*^\wedge$ be the number of triplets (v_i, v_j, v_k) such that $j < k < i$, $a_{i,j} = a_{i,k} = 1$, and $a_{j,k} = 0$. Let $s_*^\triangle$ be the number of triplets (v_i, v_j, v_k) such that $j < k < i$, $a_{i,j} = a_{i,k} = a_{j,k} = 1$. Note that $s_* = s_*^\wedge + s_*^\triangle$ and $s_*^\triangle = f_\triangle(G)$.

Consider a triangle $(v_i, v_j, v_k) \in G$. This triangle is counted $1 - p_1$ $(= \frac{e^{\varepsilon_1}}{e^{\varepsilon_1}+1})$ times in expectation in t_*. Consider 2-edges $(v_i, v_j, v_k) \in G$ (i.e., exactly one edge is missing in the three nodes). This is counted p_1 $(= \frac{1}{e^{\varepsilon_1}+1})$ times in expectation in t_*. No other events can change t_*. Therefore, we obtain:

$$\mathbb{E}[t_*] = (1-p_1)s_*^\triangle + p_1 s_*^\wedge.$$

By $s_* = s_*^{\wedge} + s_*^{\triangle}$ and $s_*^{\triangle} = f_\triangle(G)$, we obtain:

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{n} w_i\right] &= \mathbb{E}\left[\sum_{i=1}^{n}(t_i - p_1 s_i)\right] \\
&= \mathbb{E}[t_* - p_1 s_*] \\
&= \mathbb{E}[t_*] - p_1\mathbb{E}[s_*^{\wedge} + s_*^{\triangle}] \\
&= (1-p_1)s_*^{\triangle} + p_1 s_*^{\wedge} - p_1(s_*^{\wedge} + s_*^{\triangle}) \\
&= (1-2p_1)f_\triangle(G),
\end{aligned}$$

hence

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

$\square$

A.4.7 Proof of Theorem 5

Let $\mathscr{R}_i$ be $\mathsf{Local2Rounds}_\triangle$. Consider two neighbor lists $\mathbf{a}_i, \mathbf{a}'_i \in \{0,1\}^n$ that differ in one bit. Let d_i (resp. d'_i) $\in \mathbb{Z}_{\geq 0}$ be the number of "1"s in $\mathbf{a}_i$ (resp. $\mathbf{a}'_i$). Let $\bar{\mathbf{a}}_i$ (resp. $\bar{\mathbf{a}}'_i$) $\in \{0,1\}^n$ be neighbor lists obtained by setting all of the i-th to the n-th elements in $\mathbf{a}_i$ (resp. $\mathbf{a}'_i$) to 0. Let $\bar{d}_i$ (resp. $\bar{d}'_i$) $\in \mathbb{Z}_{\geq 0}$ be the number of "1"s in $\bar{\mathbf{a}}_i$ (resp. $\bar{\mathbf{a}}'_i$). For example, if $n=6$, $\mathbf{a}_4 = (1,0,1,0,1,1)$, and $\mathbf{a}'_4 = (1,1,1,0,1,1)$, then $d_4 = 4$, $d'_4 = 5$, $\bar{\mathbf{a}}_4 = (1,0,1,0,0,0)$, $\bar{\mathbf{a}}'_4 = (1,1,1,0,0,0)$, $\bar{d}_4 = 2$, and $\bar{d}'_4 = 3$.

Furthermore, let t_i (resp. 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 $(v_j, v_k) \in E'$ in $\mathbf{a}_i$ (resp. $\mathbf{a}'_i$). Let s_i (resp. 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 $(v_i, v_k) \in E$ in $\mathbf{a}_i$ (resp. $\mathbf{a}'_i$). Let $w_i = t_i - p_1 s_i$ and $w'_i = t'_i - p_1 s'_i$. Below we consider two cases about d_i: when $d_i < \tilde{d}_{max}$ and when $d_i \geq \tilde{d}_{max}$.

Case 1: $d_i < \tilde{d}_{max}$**.** Assume that $d'_i = d_i + 1$. In this case, we have either $\bar{\mathbf{a}}'_i = \bar{\mathbf{a}}_i$ or $\bar{d}'_i = \bar{d}_i + 1$. If $\bar{\mathbf{a}}'_i = \bar{\mathbf{a}}_i$, then $s_i = s'_i$, $t_i = t'_i$, and $w_i = w'_i$, hence $\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{w}_i] = \Pr[\mathscr{R}_i(\mathbf{a}'_i) = \hat{w}_i]$. If $\bar{d}'_i = \bar{d}_i + 1$, then s_i and s'_i can be expressed as $s_i = \binom{\bar{d}_i}{2}$ and $s'_i = \binom{\bar{d}'_i}{2} = \binom{\bar{d}_i+1}{2}$, respectively. Then we obtain:

$$s'_i - s_i = \binom{\bar{d}_i + 1}{2} - \binom{\bar{d}_i}{2} = \bar{d}_i.$$

In addition, since we consider an additional constraint "$(v_j, v_k) \in E'$" in counting t_i and t'_i, we have $t'_i - t_i \leq s'_i - s_i$. Therefore,

$$\begin{aligned}
|w'_i - w_i| &= |t'_i - t_i - p_1(s'_i - s_i)| \\
&\leq (1 - p_1)\bar{d}_i \\
&\leq (1 - p_1)d_i \\
&< \tilde{d}_{max} \quad \text{(by } p_1 > 0 \text{ and } d_i < \tilde{d}_{max}\text{)}.
\end{aligned}$$

Since we add $\mathrm{Lap}(\frac{\tilde{d}_{max}}{\varepsilon_2})$ to w_i, we obtain:

$$\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{w}_i] \leq e^{\varepsilon_2} \Pr[\mathscr{R}_i(\mathbf{a}'_i) = \hat{w}_i]. \tag{A.14}$$

Assume that $d'_i = d_i - 1$. In this case, we have either $\bar{\mathbf{a}}'_i = \bar{\mathbf{a}}_i$ or $\bar{d}'_i = \bar{d}_i - 1$. If $\bar{\mathbf{a}}'_i = \bar{\mathbf{a}}_i$, then $\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{w}_i] = \Pr[\mathscr{R}_i(\mathbf{a}'_i) = \hat{w}_i]$. If $\bar{d}'_i = \bar{d}_i - 1$, then we obtain $s_i - s'_i = \bar{d}_i - 1$ and $t_i - t'_i \leq s_i - s'_i$. Thus $|w'_i - w_i| \leq (1 - p_1)(\tilde{d}_i - 1) < \tilde{d}_{max}$ and (A.14) holds. Therefore, $\mathsf{Local2Rounds}_\triangle$ provides ε_2-edge LDP at the second round. Since $\mathsf{Local2Rounds}_\triangle$ provides ε_1-edge LDP at the first round (by Theorem 3), it provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP in total by the composition theorem [73].

Case 2: $d_i \geq \tilde{d}_{max}$**.** Assume that $d'_i = d_i + 1$. In this case, we obtain $d_i = d'_i = \tilde{d}_{max}$ after graph projection.

Note that $\mathbf{a}_i$ and $\mathbf{a}'_i$ can differ in *zero or two bits* after graph projection. For example, consider the case where $n = 8$, $\mathbf{a}_5 = (1,1,0,1,0,1,1,1)$, $\mathbf{a}'_5 = (1,1,1,1,0,1,1,1)$, and $\tilde{d}_{max} = 4$.

If the permutation is 1,4,6,8,2,7,5,3, then $\mathbf{a}_5 = \mathbf{a}'_5 = (1,0,0,1,0,1,0,1)$ after graph projection. However, if the permutation is 3,1,4,6,8,2,7,5, then $\mathbf{a}_5$ and $\mathbf{a}'_5$ become $\mathbf{a}_5 = (1,0,0,1,0,1,0,1)$ and $\mathbf{a}'_5 = (1,0,1,1,0,1,0,0)$, respectively; i.e., they differ in the third and eighth elements.

If $\mathbf{a}_i = \mathbf{a}'_i$, then $\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{w}_i] = \Pr[\mathscr{R}_i(\mathbf{a}'_i) = \hat{w}_i]$. If $\mathbf{a}_i$ and $\mathbf{a}'_i$ differ in two bits, $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in *at most two bits* (because we set all of the i-th to the n-th elements in $\mathbf{a}_i$ and $\mathbf{a}'_i$ to 0). For example, we can consider the following three cases:

- If $\mathbf{a}_5 = (1,0,0,1,0,1,0,1)$ and $\mathbf{a}'_5 = (1,0,0,1,0,1,1,0)$,
  then $\bar{\mathbf{a}}_5 = \bar{\mathbf{a}}'_5 = (1,0,0,1,0,0,0,0)$.

- If $\mathbf{a}_5 = (1,0,0,1,0,1,0,1)$ and $\mathbf{a}'_5 = (1,0,1,1,0,1,0,0)$, then $\bar{\mathbf{a}}_5 = (1,0,0,1,0,0,0,0)$ and $\bar{\mathbf{a}}'_5 = (1,0,1,1,0,0,0,0)$; i.e., they differ in one bit.

- If $\mathbf{a}_5 = (1,1,0,1,0,1,0,0)$ and $\mathbf{a}'_5 = (1,0,1,1,0,1,0,0)$, then $\bar{\mathbf{a}}_5 = (1,1,0,1,0,0,0,0)$ and $\bar{\mathbf{a}}'_5 = (1,0,1,1,0,0,0,0)$; i.e., they differ in two bits.

If $\bar{\mathbf{a}}_i = \bar{\mathbf{a}}'_i$, then $\Pr[\mathscr{R}_i(\mathbf{a}_i) = \hat{w}_i] = \Pr[\mathscr{R}_i(\mathbf{a}'_i) = \hat{w}_i]$. If $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in one bit, then $\bar{d}'_i = \bar{d}_i + 1$. In this case, we obtain (A.14) in the same way as **Case 1**.

We need to be careful when $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in two bits. In this case, $\bar{d}'_i = \bar{d}_i$ (because $d_i = d'_i = \tilde{d}_{max}$ after graph projection). Then we obtain $s_i = s'_i = \binom{\tilde{d}_{max}}{2}$. Since the number of 2-stars that involve a particular user in $\bar{\mathbf{a}}_i$ is $\bar{d}_i - 1$, we obtain $t'_i - t_i \leq \bar{d}_i - 1$. Therefore,

$$|w'_i - w_i| = |t'_i - t_i| \leq \bar{d}_i - 1 < \tilde{d}_{max},$$

and (A.14) holds. Therefore, if $d'_i = d_i + 1$, then $\mathsf{Local2Rounds}_\triangle$ provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP in total.

Assume that $d'_i = d_i - 1$. If $d_i > \tilde{d}_{max}$, then $d_i = d'_i = \tilde{d}_{max}$ after graph projection. Thus $\mathsf{Local2Rounds}_\triangle$ provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP in total in the same as above. If $d_i = \tilde{d}_{max}$, then we obtain (A.14) in the same way as **Case 1**, and therefore $\mathsf{Local2Rounds}_\triangle$ provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP in total.

In summary, $\mathsf{Local2Rounds}_\triangle$ provides $(\varepsilon_1+\varepsilon_2)$-edge LDP in both **Case 1** and **Case 2**. $\mathsf{Local2Rounds}_\triangle$ also provides $(\varepsilon_1+\varepsilon_2)$-relationship DP because it uses only the lower triangular part of the adjacency matrix $\mathbf{A}$. □

A.4.8 Proof of Theorem 6

When the maximum degree d_{max} of G is at most $\tilde{d}_{max}$, no graph projection will occur. By Proposition 3, the estimate $f_\triangle(G,\varepsilon)$ by $\mathsf{Local2Rounds}_\triangle$ is unbiased.

By bias-variance decomposition (A.1), the expected l_2 loss $\mathbb{E}[l_2^2(\hat{f}_\triangle(G,\varepsilon), f_\triangle(G))]$ is equal to $\mathbb{V}[\hat{f}_\triangle(G,\varepsilon)]$. Recall that $p_1=\frac{1}{1+e^{\varepsilon_1}}$. $\mathbb{V}[\hat{f}_\triangle(G,\varepsilon)]$ can be written as follows:

$$
\begin{aligned}
&\mathbb{V}[\hat{f}_\triangle(G,\varepsilon)] &\text{(A.15)}\\
&=\tfrac{1}{(1-2p_1)^2}\mathbb{V}\left[\textstyle\sum_{i=1}^n \hat{w}_i\right]\\
&=\tfrac{1}{(1-2p_1)^2}\mathbb{V}\left[\textstyle\sum_{i=1}^n t_i - p_1 s_i + \mathrm{Lap}\left(\frac{\tilde{d}_{max}(1-p_1)}{\varepsilon_2}\right)\right]\\
&=\tfrac{1}{(1-2p_1)^2}\left(\mathbb{V}\left[\textstyle\sum_{i=1}^n t_i - p_1 s_i\right] + \mathbb{V}\left[\textstyle\sum_{i=1}^n \mathrm{Lap}\left(\frac{\tilde{d}_{max}(1-p_1)}{\varepsilon_2}\right)\right]\right)\\
&=\tfrac{1}{(1-2p_1)^2}\mathbb{V}\left[\textstyle\sum_{i=1}^n t_i\right] + \tfrac{n}{(1-2p_1)^2}2\frac{\tilde{d}_{max}^2(1-p_1)^2}{\varepsilon_2^2}. &\text{(A.16)}
\end{aligned}
$$

In the last line, we are able to get rid of the s_i's because they are deterministic. We are also able to sum the variances of the Lap random variables since they are independent; we are not able to do the same with the sum of the t_is.

Recall the definition of E' computed by the first round of $\mathsf{Local2Rounds}_\triangle$—the noisy edges released by randomized response. Now,

$$t_i = \sum_{a_{i,j}=a_{i,k}=1, j<k<i} \mathbf{1}((v_j,v_k)\in E').$$

This gives

$$\begin{aligned}
\sum_{i=1}^{n} t_i &= \sum_{i=1}^{n} \sum_{\substack{a_{i,j}=a_{i,k}=1 \\ j<k<i}} \mathbf{1}((v_j, v_k) \in E') \\
&= \sum_{1\le j<k\le n} \sum_{\substack{i>k \\ a_{i,j}=a_{i,k}=1}} \mathbf{1}((v_j, v_k) \in E') \\
&= \sum_{1\le j<k\le n} |\{i : i > k, a_{i,j} = a_{i,k} = 1\}| \mathbf{1}((v_j, v_k) \in E'|.
\end{aligned}$$

Let $c_{jk} = |\{i : i > k, a_{i,j} = a_{i,k} = 1\}|$. Notice that $\mathbf{1}((v_j, v_k) \in E')$ are independent events. Thus, the variance of the above expression is

$$\begin{aligned}
\mathbb{V}\left[\sum_{i=1}^{n} t_i\right] &= \mathbb{V}\left[\sum_{1\le j<k\le n} c_{jk} \mathbf{1}((v_j, v_k) \in E')\right] \\
&= \sum_{1\le j<k\le n} c_{jk}^2 \mathbb{V}[\mathbf{1}((v_j, v_k \in E'))] \\
&= p_1(1-p_1) \sum_{1\le j<k\le n} c_{jk}^2. \qquad \text{(A.17)}
\end{aligned}$$

c_{jk} is the number of ordered 2-paths from j to k in G. Because the degree of user v_j is at most $\tilde{d}_{max}$, $0 \le c_{jk} \le \tilde{d}_{max}$. There are at most $n\tilde{d}_{max}^2$ ordered 2-paths in G, since there are only $\tilde{d}_{max}$ nodes to go to once a first is picked. Thus, $\sum_{1\le j<k\le n} c_{jk} \le n\tilde{d}_{max}^2$. Using a Jensen's inequality style argument, the best way to maximize (A.17) is to have all c_{jk} be 0 or $\tilde{d}_{max}$. At most $n\tilde{d}_{max}$ of the c_{jk} can be $\tilde{d}_{max}$, and the rest are zero. Thus,

$$\begin{aligned}
\mathbb{V}\left[\sum_{i=1}^{n} t_i\right] &= p_1(1-p_1) \sum_{1\le j<k\le n} c_{ij}^2 \\
&\le p_1(1-p_1) n\tilde{d}_{max} \times \tilde{d}_{max}^2.
\end{aligned}$$

Plugging this into (A.16)

$$\begin{aligned}
\mathbb{V}[\hat{f}_{\triangle}(G,\varepsilon)] &\leq \frac{p_1(1-p_1)n\tilde{d}_{max}^3}{(1-2p_1)^2} + \frac{2n\tilde{d}_{max}^2(1-p_1)^2}{(1-2p_1)^2\varepsilon_2^2} \\
&\leq O\left(\frac{p_1 n\tilde{d}_{max}^3 + n\tilde{d}_{max}^2/\varepsilon_2^2}{(1-2p_1)^2}\right) \\
&\leq O\left(\frac{e^{\varepsilon_1}}{(1-e^{\varepsilon_1})^2}\left(n\tilde{d}_{max}^3 + \frac{e^{\varepsilon_1}}{\varepsilon_2^2}n\tilde{d}_{max}^2\right)\right).
\end{aligned}$$

□

A.4.9 Proof of Theorem 7

Preliminaries.

We begin by defining a Boolean version of the independent cube in Definition 5, which we call the *Boolean independent cube*. The Boolean independent cube works for functions $g : \{0,1\}^{\kappa} \to \mathbb{R}$ in the local DP model, where each of $\kappa \in \mathbb{N}$ users has a *single bit* and obfuscates the bit to provide ε-DP. As shown later, there is a one-to-one correspondence between the independent cube in Definition 5 and the Boolean independent cube. Based on this, we show a lower-bound for the Boolean independent cube, and use the lower-bound to prove Theorem 7.

Below we define the Boolean independent cube. For $i \in [\kappa]$, let $x_i \in \{0,1\}$ be a bit of user v_i. Let $X = (x_1, \ldots, x_\kappa)$. We assume user v_i obfuscates x_i using a randomizer $\mathscr{S}_i : \{0,1\} \to \mathscr{Z}_i$, where $\mathscr{S}_i$ satisfies ε-DP and $\mathscr{Z}_i$ is a range of $\mathscr{S}_i$. Examples of $\mathscr{S}_i$ include Warner's RR. Furthermore, we assume the one-round setting, where each $\mathscr{S}_i$ is independent, and where the estimator $\hat{g}$ for g has the form

$$\hat{g}(X) = \tilde{g}(\mathscr{S}_1(x_1), \ldots, \mathscr{S}_\kappa(x_\kappa)). \tag{A.18}$$

$\tilde{g}$ is an aggregate function that takes $\mathscr{S}_1(x_1), \ldots, \mathscr{S}_\kappa(x_\kappa)$ as input and outputs $\hat{g}(X)$.

We will prove a lower bound which uses the following stripped-down form of an independent cube (Definition 5).

Definition 14. *[Boolean (κ, D)-independent cube] Let $g : \{0,1\}^\kappa \to \mathbb{R}$, and $D \in \mathbb{R}$. We say g has a* Boolean (κ, D)-independent cube *if for all $(x_1, \dots, x_\kappa) \in \{0,1\}^\kappa$ we have*

$$g(x_1, \dots, x_\kappa) = g(0, 0, \dots, 0) + \sum_{i=1}^{\kappa} x_i C_i,$$

where $C_i \in \mathbb{R}$ satisfies $|C_i| \geq D$ for any $i \in [\kappa]$.

The following theorem applies to the Boolean independent cube and will help us establish Theorem 7. We prove this theorem in Section A.4.10.

Theorem 24. *Let $g : \mathscr{X}^\kappa \to \mathbb{R}$ be a function that has a Boolean (κ, D)-independent cube. Let $\hat{g}(X)$ be an estimator having the form of* (A.18)*, where each $\mathscr{S}_i$ provides ε-DP and is mutually independent. Let X be drawn uniformly from $\{0,1\}^\kappa$. Over the randomness both in selecting X and in $\mathscr{S}_1, \dots, \mathscr{S}_\kappa$, $\mathbb{E}_{X, \mathscr{S}_1, \dots, \mathscr{S}_\kappa}[l_2^2(g(X), \hat{g}(X))] = \Omega\left(\frac{e^\varepsilon}{(e^\varepsilon + 1)^2} \kappa D^2\right)$.*

Proof of Theorem 7 using Theorem 24.

To prove Theorem 7, let $\mathscr{A}$ be the (n, D)-independent cube (Definition 5) for f given in the statement of Theorem 7. Let G be the graph, and $\mathbf{A}$ be the corresponding symmetric adjacency matrix. Below we sometimes write f as a function on neighbor lists $\mathbf{a}_1, \dots, \mathbf{a}_n$ (rather than G) because there is a one-to-one correspondence between G and $\mathbf{a}_1, \dots, \mathbf{a}_n$. Let M be the perfect matching that defines $\mathscr{A}$. Let $n = 2\kappa$.

The idea is to pair up users that M matches to make a new function g that has a Boolean (κ, D)-independent cube and new randomizers $\mathscr{S}_1, \dots, \mathscr{S}_\kappa$ that satisfy ε-DP. In other words, we regard a pair of users in M as a *virtual* user (since $n = 2\kappa$, there are κ virtual users in total). Then we apply Theorem 24.

Assume that $M = \{(v_1, v_2), (v_3, v_4), \dots, (v_{2\kappa-1}, v_{2\kappa})\}$ without loss of generality (we can construct g and $\mathscr{S}_1,\ldots,\mathscr{S}_\kappa$ for arbitrary M in the same way). For $x_1,\ldots,x_\kappa \in \{0,1\}$, define

$$\begin{aligned} g(x_1,\ldots,x_\kappa) =& f(\mathbf{a}_1 + x_1\mathbf{e}_2,\ \mathbf{a}_2 + x_1\mathbf{e}_1,\ \ldots, \\ & \mathbf{a}_{2\kappa-1} + x_\kappa\mathbf{e}_{2\kappa},\ \mathbf{a}_{2\kappa} + x_\kappa\mathbf{e}_{2\kappa-1}), \end{aligned}$$

where $\mathbf{e}_i \in \{0,1\}^n$ is the i-th standard basis vector that has 1 in the i-th coordinate and 0 elsewhere. In other words, $x_i \in \{0,1\}$ indicates whether the i-th edge in M should be added to G. Thus, g has a Boolean (κ,D)-independent cube, and there is a one-to-one correspondence between an (n,D)-independent cube $\mathscr{A}$ in Definition 5 and (κ,D)-Boolean independent cube $\{0,1\}^\kappa$ in Definition 14. Figure A.4 shows a $(2,2)$-Boolean independent cube for g corresponding to the $(4,2)$-independent cube for f in Figure 1.3.

Now, for $i \in [\kappa]$, define $\mathscr{S}_i(x_i)$ for $x_i \in \{0,1\}$ by

$$\mathscr{S}_i(x_i) = (\mathscr{R}_{2i-1}(\mathbf{a}_{2i-1} + x_i\mathbf{e}_{2i}), \mathscr{R}_{2i}(\mathbf{a}_{2i} + x_i\mathbf{e}_{2i-1})). \tag{A.19}$$

In other words, $\mathscr{S}_i(x_i)$ is simply the product of the outputs of users (v_{2i-1}, v_{2i}), with x_i indicating whether to add the edge in M between them.

Assume that each $\mathscr{R}_i$ is mutually independent and that $(\mathscr{R}_1,\ldots,\mathscr{R}_n)$ provides ε-relationship DP in Definition 3. Then by (1.3) and (A.19), each $\mathscr{S}_i$ provides ε-DP and is mutually independent.

Define the estimator $\hat{g}$ by

$$\hat{g}(x_1,\ldots,x_\kappa) = \tilde{f}(\mathscr{S}_1(x_1),\ldots,\mathscr{S}_\kappa(x_\kappa)).$$

Then by Theorem 24, for $X = (x_1,\ldots,x_\kappa)$,

$$\mathbb{E}_{X,\mathscr{S}_1,\ldots,\mathscr{S}_\kappa}[l_2^2(g(X),\hat{g}(X))] \geq \Omega\left(\frac{e^\varepsilon}{(e^\varepsilon+1)^2}\kappa D^2\right).$$

Since there is a one-to-one correspondence between the (n,D)-independent cube $\mathscr{A}$ and the (κ,D)-Boolean independent cube $\{0,1\}^{\kappa}$, we also have

$$\mathbb{E}_{G,\mathscr{R}_1,\ldots,\mathscr{R}_n}[l_2^2(f(G),\hat{f}(G))] \geq \Omega\left(\frac{e^{\varepsilon}}{(e^{\varepsilon}+1)^2}nD^2\right),$$

where G is drawn uniformly from $\mathscr{A}$, which proves Theorem 7. $\square$

A.4.10 Proof of Theorem 24

Assume that $\mathscr{S}_i : \{0,1\} \to \mathscr{Z}_i$. For $X = (x_1,\ldots,x_{\kappa}) \in \{0,1\}^{\kappa}$, let $S(X) = (\mathscr{S}_1(x_1),\cdots\mathscr{S}_{\kappa}(x_{\kappa}))$ and $Z = (z_1,\ldots,z_{\kappa})$ with $z_i \in \mathscr{Z}_i$. We rewrite the quantity of interest as

$$\mathbb{E}_{X,S(X)}[l_2^2(g(X),\hat{g}(X))] = \mathbb{E}_{X,S(X)}[(g(X)-\tilde{g}(S(X)))^2].$$

By the law of total expectation, this quantity is the same as the expected value of the conditional expected value of $(g(X)-\tilde{g}(S(X)))^2$ given $S(X)=Z$:

$$\begin{aligned}&\mathbb{E}_{X,S(X)}[(g(X)-\tilde{g}(S(X)))^2]\\ &= \mathbb{E}_{S(X)}\mathbb{E}_X[(g(X)-\tilde{g}(Z))^2|S(X)=Z]. \qquad \text{(A.20)}\end{aligned}$$

Let $\mu_Z = \mathbb{E}_X[g(X)|S(X) = Z]$. Then the inner expectation in (A.20) can be written as follows:

$$\begin{aligned}
&\mathbb{E}_X[(g(X) - \tilde{g}(Z))^2|S(X) = Z] \\
&= \mathbb{E}_X[((g(X) - \mu_Z) + (\mu_Z - \tilde{g}(Z)))^2|S(X) = Z] \\
&= \mathbb{E}_X[(g(X) - \mu_Z)^2|S(X) = Z] \\
&\quad + 2(\mu_Z - \tilde{g}(Z))\mathbb{E}_X[(g(X) - \mu_Z)|S(X) = Z] \\
&\quad + (\mu_Z - \tilde{g}(Z))^2 \\
&= \mathbb{E}_X[(g(X) - \mu_Z)^2|S(X) = Z] + (\mu_Z - \tilde{g}(Z))^2 \\
&= \mathbb{V}_X[g(X)|S(X) = Z] + (\mu_Z - \tilde{g}(Z))^2.
\end{aligned}$$

Thus, it suffices to show that $\mathbb{V}_X[g(X)|S(X) = Z] \geq \Omega\left(\frac{e^\varepsilon}{(1+e^\varepsilon)^2}\kappa D^2\right)$. For $B = (b_1, \ldots, b_\kappa) \in \{0,1\}^\kappa$, we have

$$\Pr[X = B|S(X) = Z] = \frac{\Pr[X = B]\Pr[S(X) = Z|X = B]}{\Pr[S(X) = Z]}.$$

Since $\Pr[S(X) = Z]$ does not depend on B and $\Pr[X = B] = \frac{1}{2^\kappa}$, $\Pr[X = B|S(X) = Z]$ can also be expressed as

$$\Pr[X = B|S(X) = Z] \propto \Pr[S(X) = Z|X = B]. \tag{A.21}$$

Since $S_1, \ldots, S_\kappa$ are independently run, we have

$$\begin{aligned}
\Pr[S(X) = Z|X = B] &= \Pr[\mathscr{S}_1(b_1) = z_1, \ldots, \mathscr{S}_\kappa(b_\kappa) = z_\kappa] \\
&= \prod_{i=1}^{\kappa}\Pr[\mathscr{S}_i(b_i) = z_i].
\end{aligned}$$

Define

$$p_i = \frac{\Pr[\mathscr{S}_i(1) = z_i]}{\Pr[\mathscr{S}_i(0) = z_i] + \Pr[\mathscr{S}_i(1) = z_i]}.$$

Because each $\mathscr{S}_i$ satisfies ε-DP, we have $\frac{1}{1+e^\varepsilon} \leq p_i \leq \frac{e^\varepsilon}{1+e^\varepsilon}$. By (A.21) and $\sum_{B \in \{0,1\}^\kappa} \Pr[X = B|S(X) = Z] = 1$, we have

$$\Pr[X = B|S(X) = Z] = \prod_{i=1}^{\kappa} (p_i)^{b_i} (1-p_i)^{1-b_i}. \tag{A.22}$$

This means that $\Pr[X = B|S(X) = Z]$ is distributed according to the independent product of $Bernoulli(p_i)$ for $i \in [\kappa]$.

Now, because g has a Boolean (κ, D)-independent cube, there are $C_1, \ldots, C_\kappa \in \mathscr{S}$ with $|C_i| \geq D$ such that

$$g(X) = g(0, \ldots, 0) + \sum_{i=1}^{\kappa} x_i C_i.$$

By (A.22), x_i is an independent draw from $Bernoulli(p_i)$ given $S(X) = Z$. Thus, the variance of $g(X)$ given $S(X) = Z$ is

$$\begin{aligned}
\mathbb{V}_X[g(X)|S(X) = Z] &= \sum_{i=1}^{\kappa} \mathbb{V}[x_i|S(X) = Z] C_i^2 \\
&\geq \sum_{i=1}^{\kappa} p_i (1-p_i) D^2 \\
&\geq \sum_{i=1}^{\kappa} \frac{e^\varepsilon}{(1+e^\varepsilon)^2} D^2 \\
&\geq \kappa \frac{e^\varepsilon}{(1+e^\varepsilon)^2} D^2.
\end{aligned}$$

$\square$

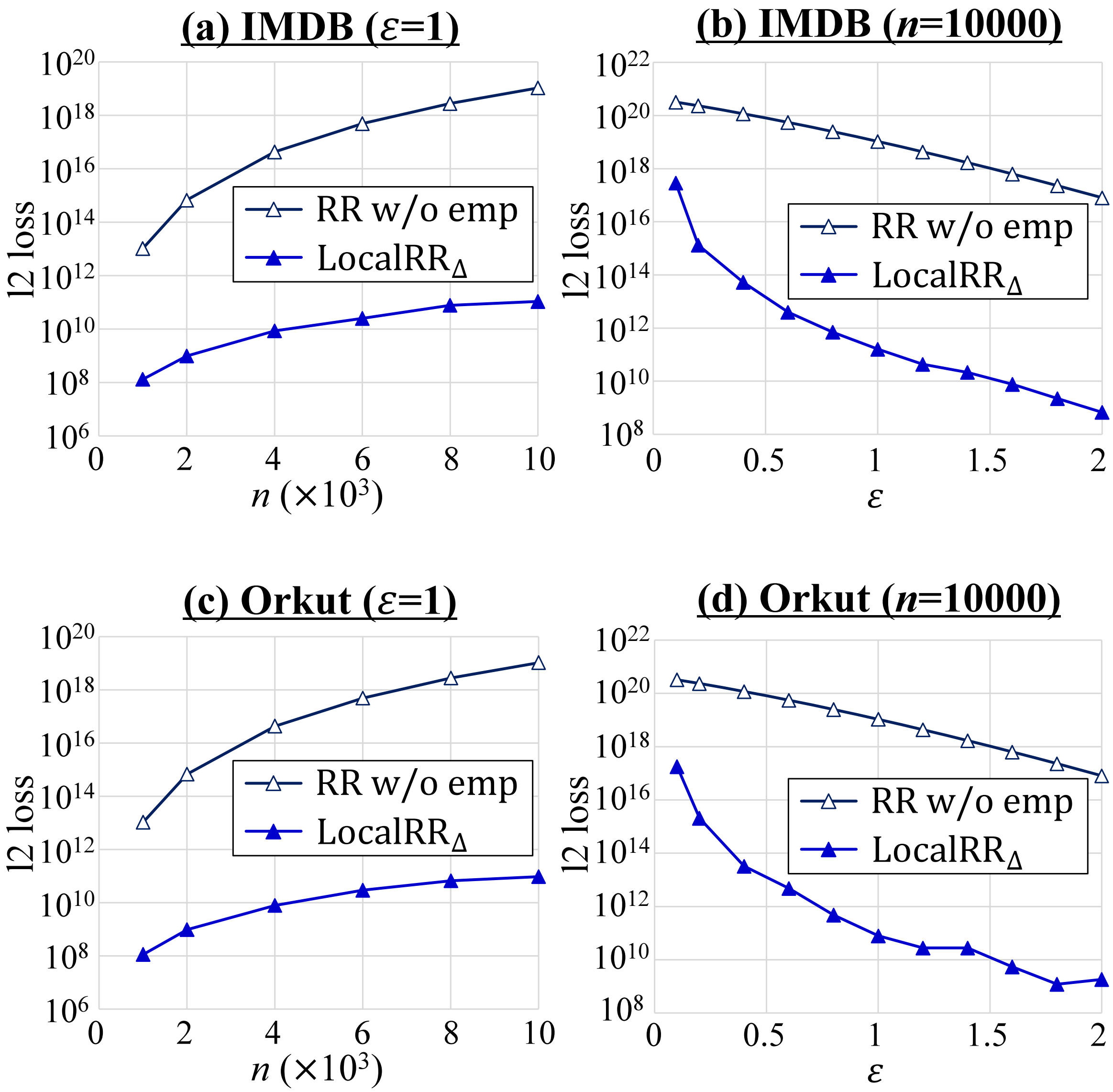


Figure A.1. l_2 loss of LocalRR$_\triangle$ and RR without empirical estimation (RR w/o emp).

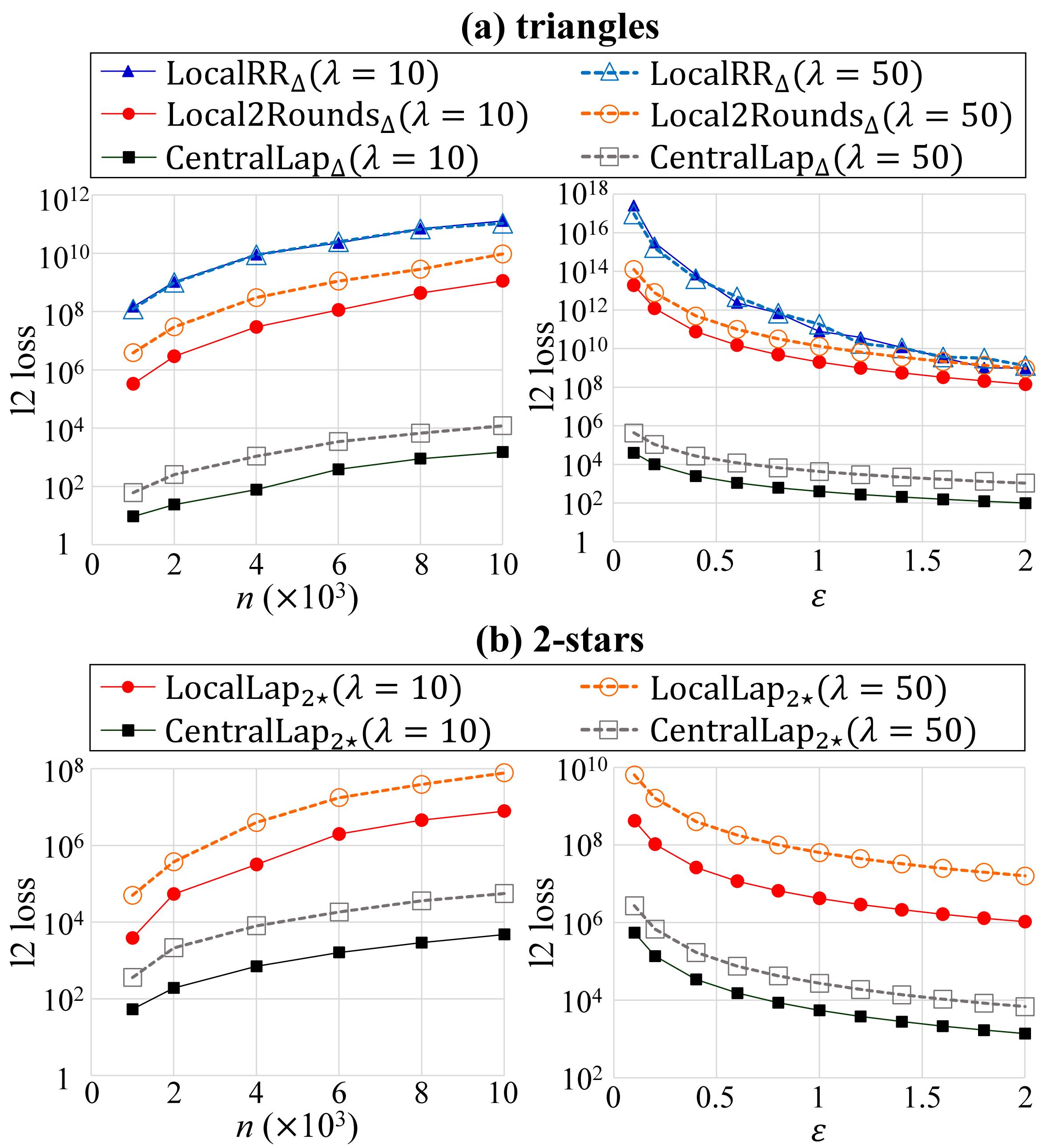


Figure A.2. l_2 loss in the Barabási-Albert graph datasets (left: $\varepsilon = 1$, right: $n = 10000$). We set the attachment parameter λ in the BA model to $\lambda = 10$ or 50, and $\tilde{d}_{max}$ to $\tilde{d}_{max} = d_{max}$.

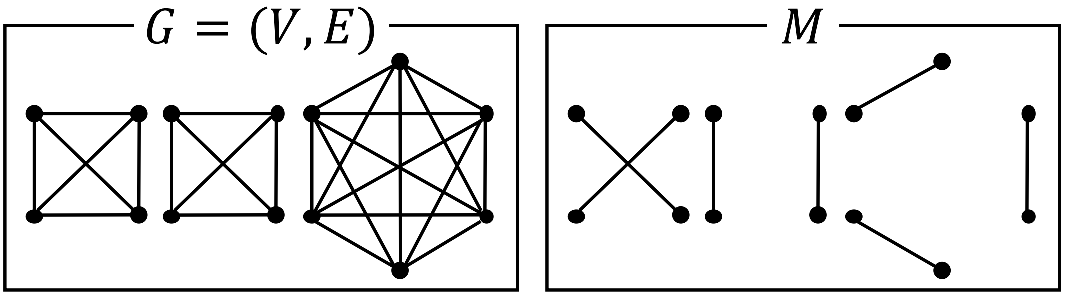


Figure A.3. Examples of G and M for constructing an independent cube for $f_{\triangle}$ ($n = 14$, $d_{max} = 8$, $\eta_1 = 3$, $\eta_2 = 2$).

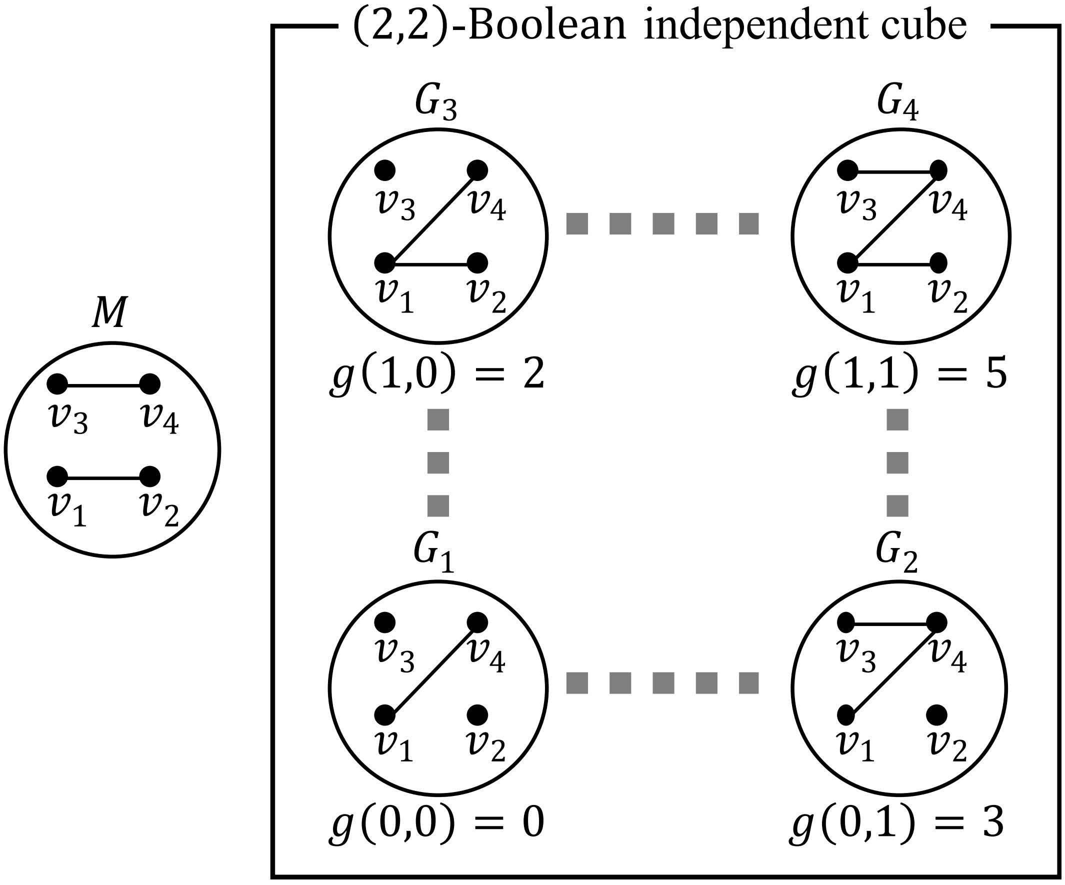


Figure A.4. $(2,2)$-Boolean independent cube for g corresponding to the $(4,2)$-independent cube for f in Figure 1.3.

Appendix B

B.1 Basic Notations

Table B.1 shows the basic notations used in this paper.

B.2 Comparison with One-Round Algorithms

Below we show that one-round triangle counting algorithms suffer from a prohibitively large estimation error.

First, we note that all of the existing one-round triangle algorithms in [102, 218, 219] are inefficient and *cannot be directly applied to a large-scale graph* such as Gplus and IMDB in Section 2.6. Specifically, in their algorithms, each user v_i applies Warner's RR to each bit of her neighbor list $\mathbf{a}_i$ and sends the noisy neighbor list to the server. Then the server counts the number of noisy triangles, each of which has three noisy edges, and estimates $f_\triangle(G)$ based on the noisy triangle count. The noisy graph G' in the server is dense, and there are $O(n^3)$ noisy triangles in G'. Thus, the time complexity of the existing one-round algorithms [102, 218, 219] is $O(n^3)$. It is also reported in [102] that when $n = 10^6$, the one-round algorithms would require about 35 years even using a supercomputer, due to the enormous number of noisy triangles.

Therefore, we evaluated the existing one-round algorithms by taking the following two steps. First, we evaluate all the existing algorithms in [102, 218, 219] using small graph datasets ($n = 10000$) and show that the algorithm in [102] achieves the lowest estimation error. Second,

Table B.1. Basic notations.

Symbol	Description
$G = (V, E)$	Graph with n users V and edges E.
v_i	i-th user in V (i.e., $V = \{v_1, \dots, v_n\}$).
d_{max}	Maximum degree of G.
$\mathscr{G}$	Set of possible graphs with n users.
$f_\triangle(G)$	Triangle count in G.
$\mathbf{A} = (a_{i,j})$	Adjacency matrix.
$\mathbf{a}_i$	Neighbor list of v_i (i.e., i-th row of $\mathbf{A}$).
$\mathscr{R}_i$	Local randomizer of v_i.
M_i	Message sent from the server to user v_i.
μ	Parameter in the ARR.
μ^*	$= \mu, \mu^2, \mu^3$ in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, and $\mathsf{ARRTwoNS}_\triangle$, respectively.
$\tilde{d}_i$	Noisy degree of user v_i.
κ_i	Clipping threshold of user v_i.
ε_0	Privacy budget for edge clipping.
ε_1	Privacy budget for the ARR.
ε_2	Privacy budget for the Laplacian noise.
ε	Total privacy budget.

we improve the time complexity of the algorithm in [102] using the ARR (i.e., edge sampling after Warner's RR) and compare it with our two-rounds algorithms using large graph datasets in Section 2.6.

Small Datasets. We first evaluated the existing algorithms in [102, 218, 219] using small datasets. For both Gplus and IMDB in Section 2.6, we first randomly selected $n = 10000$ users from all users and extracted a graph with n users. Then we evaluated the relative error of the following three algorithms: (i) RR (biased) [102, 218], (ii) RR (bias-reduced) [219], and (iii) RR (unbiased) [102]. All of them provide ε-edge LDP.

RR (biased) simply uses the number of noisy triangles in the noisy graph G' obtained by Warner's RR as an estimate of $f_\triangle(G)$. Clearly, it suffers from a very large bias, as G' is dense. RR (bias-reduced) reduces this bias by using a noisy degree sent by each user. However, it introduces some approximation to estimate $f_\triangle(G)$, and consequently, it is not clear whether the estimate is unbiased. We used the mean of the noisy degrees as a representative degree to obtain the optimal privacy budget allocation (see [219] for details). RR (unbiased) calculates an

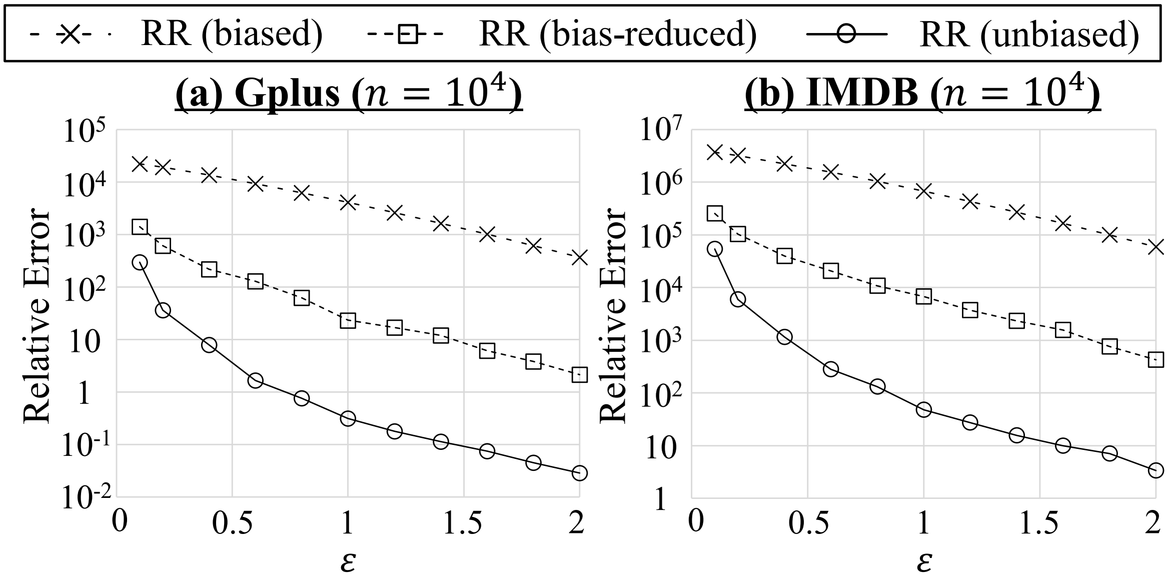


Figure B.1. Relative error of one-round algorithms for small datasets ($n = 10000$).

unbiased estimate of $f_\triangle(G)$ via empirical estimation. It is proved that the estimate is unbiased [102].

In all of the three algorithms, each user v_i obfuscates bits for smaller user IDs in her neighbor list $\mathbf{a}_i$. We averaged the relative error over 10 runs.

Figure B.1 shows the results. RR (bias-reduced) significantly outperforms RR (biased) and is significantly outperformed by RR (unbiased). We believe this is caused by the fact that RR (bias-reduced) introduces some approximation and does not calculate an unbiased estimate of $f_\triangle(G)$.

Large Datasets. Based on Figure B.1, we improve the time complexity of RR (unbiased) using the ARR and compare it with our two-rounds algorithms in large datasets.

Specifically, RR (unbiased) counts *triangles*, 2-*edges* (three nodes with two edges), 1-*edges* (three nodes with one edge), and *no-edges* (three nodes with no edges) in G' obtained by Warner's RR. Let $m_3, m_2, m_1, m_0 \in \mathbb{Z}_{\geq 0}$ be the numbers of triangles, 2-edges, 1-edges, and no-edges, respectively, after applying Warner's RR. RR (unbiased) calculates an unbiased estimate of $f_\triangle(G)$ from these four values. Thus, we improve RR (unbiased) by using the ARR, which samples each edge with probability p_2 after Warner's RR, and then calculating unbiased estimates of m_3, m_2, m_1, and m_0.

Let $\hat{m}_3, \hat{m}_2, \hat{m}_1, \hat{m}_0 \in \mathbb{R}$ be the unbiased estimates of m_3, m_2, m_1, and m_0, respectively. Let $m_3^*, m_2^*, m_1^*, m_0^* \in \mathbb{Z}_{\geq 0}$ be the number of triangles, 2-edges, 1-edges, no-edges, respectively, after applying the ARR. Since the ARR samples each edge with probability p_2, we obtain:

$$
\begin{aligned}
m_3^* &= p_2^3 \hat{m}_3 \\
m_2^* &= 3p_2^2(1-p_2)\hat{m}_3 + p_2^2\hat{m}_2 \\
m_1^* &= 3p_2(1-p_2)^2\hat{m}_3 + 2p_2(1-p_2)\hat{m}_2 + p_2\hat{m}_1.
\end{aligned}
$$

By these equations, we obtain:

$$\hat{m}_3 = \frac{m_3^*}{p_2^3} \tag{B.1}$$

$$\hat{m}_2 = \frac{m_2^*}{p_2^2} - 3(1-p_2)\hat{m}_3 \tag{B.2}$$

$$\hat{m}_1 = \frac{m_1^*}{p_2} - 3(1-p_2)^2\hat{m}_3 - 2(1-p_2)\hat{m}_2 \tag{B.3}$$

$$\hat{m}_0 = \frac{n(n-1)(n-2)}{6} - \hat{m}_3 - \hat{m}_2 - \hat{m}_1. \tag{B.4}$$

Therefore, after applying the ARR to the lower triangular part of $\mathbf{A}$, the server counts m_3^*, m_2^*, m_1^*, and m_0^* in G', and then calculates the unbiased estimates $\hat{m}_3$, $\hat{m}_2$, $\hat{m}_1$, and $\hat{m}_0$ by (B.1), (B.2), (B.3), and (B.4), respectively. Finally, the server estimates $f_\triangle(G)$ from $\hat{m}_3$, $\hat{m}_2$, $\hat{m}_1$, and $\hat{m}_0$ in the same way as RR (unbiased). We denote this algorithm by ARR (unbiased). The time complexity of ARR (unbiased) is $O(\mu^3 n^3)$, where μ is the ARR parameter.

We compared ARR (unbiased) with our three algorithms with double clipping using Gplus ($n = 107614$) and IMDB ($n = 896308$). For the sampling probability p_2, we set $p_2 = 10^{-3}$ or 10^{-6}. We averaged the relative error over 10 runs.

Figure B.2 shows the results, where we set $\mu^* = 10^{-6}$ or 10^{-3}. In ARR (unbiased), we used μ^* as the ARR parameter μ. Thus, we can see *how much the relative error is reduced by*

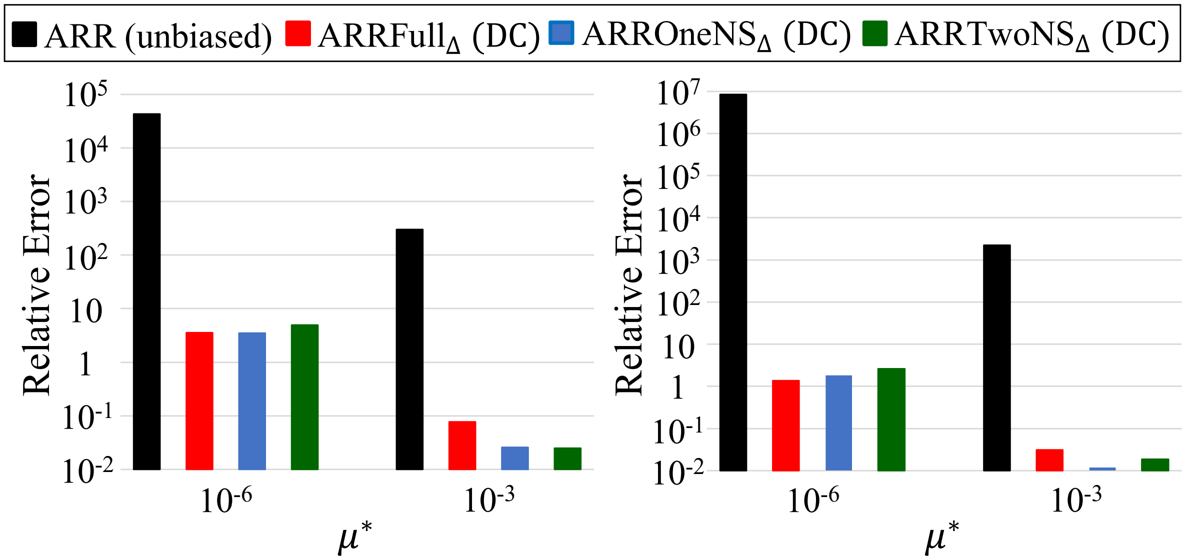


Figure B.2. Relative error of the one-round algorithm ARR (unbiased) and our three two-rounds algorithms with double clipping for large datasets ($n = 107614$ in Gplus, $n = 896308$ in IMDB).

introducing an additional round with ARRFull$_\triangle$. Figure B.2 shows that the relative error of ARR (unbiased) is prohibitively large; i.e., relative error $\gg 1$. This is because three edges are noisy in any noisy triangle. The relative error is significantly reduced by introducing an additional round because only one edge is noisy in each noisy triangle at the second round.

In summary, one-round algorithms are far from acceptable in terms of the estimation error for large graphs, and two-round algorithms such as ours are necessary.

B.3 Clustering Coefficient

Here we evaluate the estimation error of the clustering coefficient using our algorithms.

We first estimated a triangle count by using our ARROneNS$_\triangle$ with double clipping ($\varepsilon_0 = \frac{\varepsilon}{10}$ and $\varepsilon_1 = \varepsilon_2 = \frac{9\varepsilon}{20}$) because it provides the best performance in Figures 2.7, 2.8, and 2.10. Then we estimated a 2-star count by using the one-round 2-star algorithm in [102] with the edge clipping in Section 2.5.

Specifically, we calculated a noisy degree $\tilde{d}_i$ of each user v_i by using the edge clipping with the privacy budget ε_0. Then we calculated the number $r_i \in \mathbb{Z}_{\geq 0}$ of 2-stars of which user v_i is a center, and added Lap$(\frac{\Delta}{\varepsilon_1})$ to r_i, where $\Delta = \binom{\tilde{d}_i}{2}$. Let $\hat{r}_i = r_i + \text{Lap}(\frac{\Delta}{\varepsilon_1})$ be the noisy 2-star of v_i.

Finally, we calculated the sum $\sum_{i=1}^{n} \hat{r}_i$ as an estimate of the 2-star count. This 2-star algorithm provides $(\varepsilon_0 + \varepsilon_1)$-edge privacy (see [102] for details). For the privacy budgets ε_0 and ε_1, we set $\varepsilon_0 = \frac{\varepsilon}{10}$ and $\varepsilon_1 = \frac{9\varepsilon}{10}$.

Based on the triangle and 2-star counts, we estimated the clustering coefficient as $\frac{3 \times \hat{f}_{\triangle}(G)}{\hat{f}_{2\star}(G)}$, where $\hat{f}_{\triangle}(G)$ (resp. $\hat{f}_{2\star}(G)$) is the estimate of the triangle (resp. 2-star) count.

Figure B.3 shows the relative errors of the triangle count, 2-star count, and clustering coefficient. Note that the relative error of the 2-star count is not changed by changing μ^* because the 2-star algorithm does not use the ARR. Figure B.3 shows that the relative error of the 2-star count is much smaller than that of the triangle count. This is because each user can count her 2-stars locally (whereas she cannot count her triangles), as described in Section 2.1. Consequently, the relative error of the clustering coefficient is almost the same as that of the triangle count, as the denominator $\hat{f}_{2\star}(G)$ in the clustering coefficient is very accurate.

Note that the clustering coefficient requires the privacy budgets for calculating both $\hat{f}_{\triangle}(G)$ and $\hat{f}_{2\star}(G)$ (in Figure B.3, 2ε in total). However, we can accurately calculate $\hat{f}_{2\star}(G)$ with a very small privacy budget, as shown in Figure B.3. Thus, we can accurately estimate the clustering coefficient with almost the same privacy budget as the triangle count by assigning a very small privacy budget (e.g., $\varepsilon = 0.1$ or 0.2) for $\hat{f}_{2\star}(G)$.

In summary, we can accurately estimate the clustering coefficient as well as the triangle count under edge LDP by using our $\mathsf{ARROneNS}_{\triangle}$ with double clipping.

B.4 Experiments Using the Barabási-Albert Graph Datasets

In Section 2.6, we evaluated our algorithms using two real datasets. Below we also evaluate our algorithms using a synthetic graph based on the BA (Barabási-Albert) graph model [16], which has a power-law degree distribution.

In the BA graph model, a graph of n nodes is generated by attaching new nodes one by one. Each new node is connected to $m \in \mathbb{Z}_{\geq 0}$ existing nodes, and each edge is connected to an existing node with probability proportional to its degree. We used NetworkX [96], a Python package for complex networks, to generate synthetic graphs based on the BA graph model.

We generated a graph $G=(V,E)$ with the same number of nodes as Gplus; i.e., $n=107614$ nodes. For the number m of edges per node, we set $m=50$, 114, or 500. Using these graphs, we evaluated our three algorithms with double clipping. We set parameters in the same as Section 2.6; i.e., $\alpha=150$, $\beta=10^{-6}$, $\varepsilon_0=\frac{\varepsilon}{10}$, and $\varepsilon_1=\varepsilon_2=\frac{9\varepsilon}{20}$. For each algorithm, we averaged the relative error over 10 runs.

Figure B.4 shows the results, where $\varepsilon=1$ and $\mu^*=10^{-3}$. We observe that $\mathsf{ARROneNS}_\triangle$ significantly outperforms $\mathsf{ARRFull}_\triangle$ and $\mathsf{ARRTwoNS}_\triangle$ when $m=500$, and that $\mathsf{ARROneNS}_\triangle$ performs almost the same as $\mathsf{ARRFull}_\triangle$ when $m=50$ or 114.

To examine the reason for this, we also decomposed the estimation error into two components (the first error by empirical estimation and the second error by the Laplacian noise) in the same way as Figure 2.11. Figure B.5 shows the results. We also show in Table B.2 the number C_4 of 4-cycles in each BA graph ($m=50$, 114, or 500) and Gplus.

From Figure B.5 and Table B.2, we can explain Figure B.4 as follows. The BA graphs with $m=50$ and 114 have a much smaller number C_4 of 4-cycles than Gplus, as shown in Table B.2. Consequently, the Laplacian noise is relatively large and dominant for these two graphs, as shown in Figure B.5. In particular, the Laplacian noise is the largest in $\mathsf{ARRTwoNS}_\triangle$ because it cannot effectively reduce the global sensitivity by double clipping, as explained in Section 2.5. In contrast, the BA graph with $m=500$ has a larger number C_4 of 4-cycles than Gplus, and therefore the Laplacian noise is not dominant (except for $\mathsf{ARRTwoNS}_\triangle$). This explains the results in Figure B.4.

These results show that $\mathsf{ARROneNS}_\triangle$ outperforms $\mathsf{ARRFull}_\triangle$ especially when the number C_4 of 4-cycles is large. As we have shown in Section 2.6 and Appendix B.4, C_4 is large in a large graph (e.g., $n\approx 10^6$) or dense graph (e.g., Gplus, BA graph with $m=500$).

Table B.2. #4-cycles C_4 in each graph dataset.

	$m = 50$	$m = 114$	$m = 500$	Gplus
C_4	8.8×10^8	1.7×10^{10}	3.1×10^{12}	2.8×10^{12}

B.5 Edge Clipping and Noisy Triangle Clipping

In Section 2.6, we showed that our double clipping significantly reduces the estimation error. To investigate the effect of edge clipping and noisy triangle clipping independently, we also performed the following ablation study.

We evaluated our three algorithms with only edge clipping; i.e., each user calculates a noisy degree $\tilde{d}_i$ (possibly with edge clipping) and then adds Lap($\frac{\tilde{d}_i}{\varepsilon_2}$) to her noisy triangle count. Then we compared them with our algorithms with double clipping and without clipping.

Figure B.6 shows the results, where $\varepsilon = 1$, $\mu^* = 10^{-6}$ or 10^{-3}, and "EC" represents our algorithms with only edge clipping. We observe that "EC" outperforms "d_{max}" (w/o clipping) and is outperformed by "DC" (double clipping). The difference between "EC" and "DC" is significant especially when $\mu^* = 10^{-6}$ ("DC" is smaller than $\frac{1}{100}$ of "EC"). This is because our noisy triangle clipping reduces the global sensitivity by using a small value of μ^*. From Figure B.6, we conclude that each component (i.e., edge clipping, noisy triangle clipping) is essential in our double clipping.

B.6 Proof of Proposition 5

Let $\mathscr{R}_i(\mathbf{a}_i) = (\mathscr{R}_i^1(\mathbf{a}_i), \mathscr{R}_i^2(M_i)(\mathbf{a}_i))$ be the randomizer used by user v_i in the composition. To establish that $\mathscr{R}_i(\mathbf{a}_i)$ satisfies ε-edge LDP for every $v_i \in V$, we will prove that (2.1) holds for $\mathscr{R}_i(\mathbf{a}_i)$. To do this, first write

$$\begin{aligned}
&\Pr[(\mathscr{R}_i^1(\mathbf{a}_i), \mathscr{R}_i^2(M_i)(\mathbf{a}_i)) = (r_i^1, r_i^2)] = \\
&\quad \Pr[\mathscr{R}_i^1(\mathbf{a}_i) = r_i^1] \Pr[\mathscr{R}_i^2(M_i)(\mathbf{a}_i) = r_i^2 | \mathscr{R}_i^1(\mathbf{a}_i) = r_i^1] \\
&\quad \Pr[\mathscr{R}_i^1(\mathbf{a}_i) = r_i^1] \Pr[\mathscr{R}_i^2(M_i)(\mathbf{a}_i) = r_i^2 | M_i = \lambda_i(r_i^1)],
\end{aligned}$$

where the last equality follows because $M_i = \lambda_i(\mathscr{R}_i^1(\mathbf{a}_i))$ for a post-processing algorithm λ_i. Notice that the same equalities are true when we replace $\mathbf{a}_i$ with $\mathbf{a}'_i$. Because $\mathscr{R}_i^1$ and $\mathscr{R}_i^2(M_i)$ (for any M_i) satisfy $\varepsilon_1, \varepsilon_2$-edge LDP, respectively, we have

$$
\begin{aligned}
&\Pr[\mathscr{R}_i^1(\mathbf{a}_i) = r_i^1] \Pr[\mathscr{R}_i^2(M_i)(\mathbf{a}_i) = r_i^2 | M_i = \lambda_i(r_i^1)] \\
&\quad \leq e^{\varepsilon_1} \Pr[\mathscr{R}_i^1(\mathbf{a}'_i) = r_i^1] e^{\varepsilon_2} \Pr[\mathscr{R}_i^2(\mathbf{a}'_i) = r_i^2 | M_i = \lambda_i(r_i^1)] \\
&\quad = e^{\varepsilon_1 + \varepsilon_2} \Pr[(\mathscr{R}_i^1(\mathbf{a}'_i), \mathscr{R}_i^2(M_i)(\mathbf{a}'_i)) = (r_i^1, r_i^2)].
\end{aligned}
$$

This establishes the result. □

B.7 Proof of Statements in Section 2.4

B.7.1 Proof of Theorem 8

Let $\mathbf{a}_i, \mathbf{a}'_i \in \{0,1\}^n$ be two neighbor lists that differ in one bit. Let t'_i, s'_i, and w'_i be respectively the values of t_i (line 11 of Algorithm 4), s_i (line 12), and w_i (line 13) when the neighbor list of user v_i is $\mathbf{a}'_i$. Let $\Delta w_i = |w'_i - w_i|$. Then we have $t'_i - t_i \in [0, d_{max}]$ and $s'_i - s_i \in [0, d_{max}]$, and therefore $\Delta w_i = |(t'_i - t_i) - \mu^* \rho (s'_i - s_i)| \leq d_{max}$.

Since we add $\mathrm{Lap}\left(\frac{d_{max}}{\varepsilon_2}\right)$ to w_i, the second round provides ε_2-edge LDP. The first round uses $ARR_{\varepsilon_1,\mu}$ and provides ε_1-edge LDP. Thus, by sequential composition (Proposition 5), Algorithm 4 provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP in total. It also provides $(\varepsilon_1 + \varepsilon_2)$-relationship DP because it uses only the lower-triangular part of $\mathbf{A}$ (Proposition 4). □

B.7.2 Proof of Theorem 9

Unbiased Estimators. First, we will show that $\hat{f}_\triangle(G)$ satisfies $\mathbb{E}[\hat{f}_\triangle(G)] = f_\triangle(G)$ for all $G \in \mathscr{G}$, in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, $\mathsf{ARRTwoNS}_\triangle$. Regardless of algorithm, we have

$$\begin{aligned}
&\mathbb{E}[\hat{f}_\triangle(G)] \\
&= \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^{n} \mathbb{E}[w_i] \\
&= \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^{n} \mathbb{E}[t_i - \mu^* \rho s_i] \\
&= \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^{n} \sum_{\substack{1 \le j < k < i \le n \\ a_{i,j}=a_{i,k}=1}} \mathbb{E}[\mathbf{1}_{(v_j,v_k) \in M_i} - \mu^* \rho],
\end{aligned} \tag{B.5}$$

where $\rho = e^{-\varepsilon_1}$, and the quantites μ^*, t_i, s_i are defined in Algorithm 4. Given that $a_{i,j} = a_{i,k} = 1$, we have that $\Pr[(v_i, v_j) \in E'] = \Pr[(v_i, v_k) \in E'] = \mu$ by definition of ARR. Furthermore, $\Pr[(v_j, v_k) \in E'] = \mu$ if $a_{j,k} = 1$, and $\Pr[(v_j, v_k) \in E'] = \mu\rho$ otherwise. Examining (2.7), (2.8), and (2.9), we have

$$\Pr[(v_j, v_k) \in M_i] = \begin{cases} \mu^* & a_{j,k} = 1 \\ \mu^* \rho & a_{j,k} = 0 \end{cases}$$

for all the three algorithms (note that $\mu^* = \mu$, μ^2, and μ^3 in $\mathsf{ARRFull}_\triangle$, $\mathsf{ARROneNS}_\triangle$, $\mathsf{ARRTwoNS}_\triangle$, respectively). Thus, $\mathbb{E}[\mathbf{1}_{(v_j,v_k) \in M_i}] = \mu^*(\rho + (1-\rho)a_{j,k})$ ($= \mu^*$ if $a_{i,j} = 1$ and $\mu^*\rho$ if

$a_{i,j} = 0$). Plugging into (B.5), we have

$$\begin{aligned}
&\mathbb{E}[\hat{f}_\triangle(G)] \\
&= \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^{n} \sum_{\substack{1 \le j < k < i \le n \\ a_{i,j} = a_{i,k} = 1}} \mu^*(\rho + (1-\rho)a_{j,k}) - \mu^*\rho \\
&= \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^{n} \sum_{\substack{1 \le j < k < i \le n \\ a_{i,j} = a_{i,k} = 1}} \mu^*(1-\rho)a_{j,k} \\
&= \sum_{i=1}^{n} \sum_{\substack{1 \le j < k < i \le n \\ a_{i,j} = a_{i,k} = 1}} a_{j,k} \\
&= f_\triangle(G).
\end{aligned}$$

Thus, $\hat{f}_\triangle(G)$ is unbiased. □

l_2 **Loss of Estimators.** Using bias-variance decomposition, we have for any graph G,

$$\begin{aligned}
l_2^2(f_\triangle(G), \hat{f}_\triangle(G)) &= \mathbb{E}[(\hat{f}_\triangle(G) - f_\triangle(G))^2] \\
&= \mathbb{E}[(f_\triangle(G) - \mathbb{E}[\hat{f}_\triangle(G)])^2] + \mathbb{V}[\hat{f}_\triangle(G)] \\
&= \mathbb{V}[\hat{f}_\triangle(G)],
\end{aligned}$$

where the last step follows because $\hat{f}$ is unbiased. Since $\hat{f}_{\triangle}(G) = \frac{1}{\mu^*(1-\rho)} \sum_{i=1}^n \hat{w}_i$, we have

$$\begin{aligned}
&\mathbb{V}[\hat{f}_{\triangle}(G)] \\
&= \frac{1}{(\mu^*)^2(1-\rho)^2} \mathbb{V}\left[\sum_{i=1}^n \hat{w}_i\right] \\
&= \frac{1}{(\mu^*)^2(1-\rho)^2} \mathbb{V}\left[\sum_{i=1}^n w_i + \mathrm{Lap}\left(\frac{d_{max}}{\varepsilon_2}\right)\right] \\
&= \frac{1}{(\mu^*)^2(1-\rho)^2} \left(\mathbb{V}\left[\sum_{i=1}^n w_i\right] + n\mathbb{V}\left[\mathrm{Lap}\left(\frac{d_{max}}{\varepsilon_2}\right)\right]\right) \\
&= \frac{1}{(\mu^*)^2(1-\rho)^2} \left(\mathbb{V}\left[\sum_{i=1}^n w_i\right] + 2n\frac{d_{max}^2}{\varepsilon_2^2}\right),
\end{aligned} \tag{B.6}$$

where the fourth line follows from independence of the added of Laplace noise. Now, we will prove bounds on $\mathbb{V}[\sum_{i=1}^n w_i]$ for $\mathsf{ARRFull}_{\triangle}$, $\mathsf{ARROneNS}_{\triangle}$, and $\mathsf{ARRTwoNS}_{\triangle}$. In the following, we let $S_k(G)$ be the number of k-stars in G and $C_4(G)$ be the number of 4-cycles in G

Bounding the Variance in $\mathsf{ARRFull}_{\triangle}$. In $\mathsf{ARRFull}_{\triangle}$, M_i is defined by (2.7). Thus we have

$$\begin{aligned}
\sum_{i=1}^n w_i &= \sum_{i=1}^n \sum_{\substack{1\leq j<k<i\leq n\\ a_{i,j}=1, a_{i,k}=1}} \mathbf{1}_{(v_j,v_k)\in M_i} \\
&= \sum_{1\leq j<k\leq n} \sum_{k<i\leq n} a_{i,j}a_{i,k}\mathbf{1}_{(v_j,v_k)\in E'} \\
&= \sum_{1\leq j<k\leq n} \mathbf{1}_{(v_j,v_k)\in E'} \sum_{k<i\leq n} a_{i,j}a_{i,k}
\end{aligned}$$

For $j<k$, we introduce the constant $c_{jk} = \sum_{k<i\leq n} a_{i,j}a_{i,k}$. Notice that for any choice of j and k, $\mathbf{1}_{(v_j,v_k)\in E'}$ for $1\leq j\leq k$ are mutually independent, because all edges in E' are mutually independent. Furthermore, the indicator $\mathbf{1}_{(v_j,v_k)\in E'}$ is a Bernoulli random variable with parameter

either μ or $\mu\rho$, and in either case, $\mathbb{V}[\mathbf{1}_{(v_j,v_k)\in E'}] \leq \mu$. We have

$$\begin{aligned}
\mathbb{V}\left[\sum_{i=1}^{n} w_i\right] &= \mathbb{V}\left[\sum_{1\leq j<k\leq n} \mathbf{1}_{(v_j,v_k)\in E'} c_{jk}\right] \\
&= \sum_{1\leq j<k\leq n} \mathbb{V}[\mathbf{1}_{(v_j,v_k)\in E'} c_{jk}] \\
&= \sum_{1\leq j<k\leq n} \mu c_{jk}^2.
\end{aligned}$$

By Lemma 2 (which is shown at the end of Appendix B.7.2), we have $\sum_{1\leq j<k\leq n} c_{jk}^2 \leq 2C_4(G) + S_2(G)$. Plugging into (B.6) (and substituting $\mu^* = \mu$), we obtain

$$\mathbb{V}[\hat{f}_{\triangle}(G)] = \frac{1}{(1-\rho)^2}\left(\frac{1}{\mu}(2C_4(G) + S_2(G)) + 2n\frac{d_{max}^2}{\mu^2\varepsilon_2^2}\right).$$

This establishes the result. □

Bounding the Variance in ARROneNS$_\triangle$. In ARROneNS$_\triangle$, for a fixed $v_i \in V$, we have $(v_j, v_k) \in M_i$ if and only if $j < k < i$, $(v_j, v_k) \in E'$, and $(v_i, v_k) \in E'$ from (2.8). Thus,

$$\begin{aligned}
\sum_{i=1}^{n} w_i &= \sum_{i=1}^{n} \sum_{\substack{1\leq j<k<i\leq n \\ a_{i,j}=1, a_{i,k}=1}} \mathbf{1}_{(v_j,v_k)\in M_i} \\
&= \sum_{1\leq j<k<i\leq n} \mathbf{1}_{(v_j,v_k)\in E'} a_{i,j} a_{i,k} \mathbf{1}_{(v_i,v_k)\in E'}
\end{aligned}$$

Define the random variable $F_{ijk} = \mathbf{1}_{(v_j,v_k)\in E'} \mathbf{1}_{(v_i,v_k)\in E'}$. Substituting, we have

$$\begin{aligned}
\sum_{i=1}^{n} w_i &= \sum_{1\leq j<k<i\leq n} a_{i,j} a_{i,k} F_{ijk} \\
\mathbb{V}\left[\sum_{i=1}^{n} w_i\right] &= \sum_{\substack{1\leq j<k<i\leq n \\ 1\leq j'<k'<i'\leq n}} a_{i,j} a_{i,k} a_{i',j'} a_{i',k'} \operatorname{Cov}(F_{ijk}, F_{i'j'k'}).
\end{aligned}$$

The set $\{i,i',j,j',k,k'\}$ when $j<k<i$ and $j'<k'<i'$ can take between three and six distinct values. If $\{i,i',j,j',k,k'\}$ takes five or more distinct values, then F_{ijk} and $F_{i'j'k'}$ involve distinct edges and are independent random variables. Thus, $\mathrm{Cov}(F_{ijk},F_{i'j'k'})=0$. Otherwise, the events are not independent, and we will use the upper bound $\mathrm{Cov}(F_{ijk},F_{i'j'k'})\le\mathbb{E}[F_{ijk}F_{i'j'k'}]=\Pr[F_{ijk}=F_{i'j'k'}=1]$, which holds because the F_{ijk} have domain $\{0,1\}$. Thus,

$$\begin{aligned}&\mathbb{V}\left[\sum_{i=1}^{n}w_i\right]\\ &\le \sum_{\substack{1\le j<k<i\le n\\ 1\le j'<k'<i'\le n\\ |\{i,j,k,i',j',k'\}|=3\text{ or }4}} a_{i,j}a_{i,k}a_{i',j'}a_{i',k'}\Pr[F_{ijk}=F_{i'j'k'}=1].\end{aligned}$$

Define a choice of $(i,j,k,i',j',k')\in[n]^6$ to be a *valid* choice if $j<k<i$, $j'<k'<i'$ (ordering requirement), $a_{i,k}=a_{i,j}=a_{i',j'}=a_{i',k'}=1$ (edge requirement), and $3\le\{i,i',j,j',k,k'\}\le 4$ (size requirement). We can write the above sum as

$$\mathbb{V}\left[\sum_{i=1}^{n}w_i\right]\le\sum_{i,i',j,j',k,k'\text{ valid}}\Pr[F_{ijk}=F_{i'j'k'}=1].$$

In the above sum, each valid choice implies there exists a subgraph of G that associates each of $\{v_i,v_{i'},v_j,v_{j'},v_k,v_{k'}\}$ with one node in the subgraph and contains edges $(v_i,v_j),(v_i,v_k),(v_{i'},v_{j'}),(v_{i'},v_{k'})$. Conversely, each subgraph of G of three or four nodes can have a certain number of valid choices mapped to it. For each subgraph, we now go over the number of possible valid choices that can map to it:

1. 4-cycle: By ordering, either i or i' is mapped to the node of the 4-cycle with maximal index. WLOG, suppose i is mapped to this node. By edge requirements, i' has an index equal to the opposite node in the 4-cycle. By ordering, there is now one way to map j,j',k,k'. Thus, each 4-cycle can be associated with **2** valid choices.

2. 3-path: Consider the middle node v_ℓ in the 3-path (path graph on 4 nodes) that has the

second-largest index. By ordering, either $i = \ell$ or $i' = \ell$. WLOG, suppose $i = \ell$. Then, by the edge requirement $a_{i',j'} = a_{i',k'} = 1$, the middle node other than v_ℓ is i'. However, this means either $i = j'$ or $i = k'$, and we have $j' > i'$ or $k' > i'$. This violates the order requirement, and therefore there are **0** valid choices.

3. 3-star: By edge requirement, both i, i' map to the central node in the 3-star. j, j', k, k' can map to the other three nodes in any way that satisfies ordering. Suppose the three nodes are v_a, v_b, v_c with $a < b < c$. Only one of the three nodes can be duplicated in this mapping. For example, if a is duplicated, then both j and j' map to v_a, and there are two remaining choices for how to map k and k'. Thus, each S_3 can be associated with **6** valid choices.

4. Triangle: Either i or i' maps to the maximal node in the triangle by ordering. WLOG, suppose i does. By the edge requirement, i' maps to a different node. However, this means $i = k'$ or $i = j'$, so $k' > i'$, contradicting ordering. Thus, there are **0** valid choices.

5. 2-star: By edge requirements, both i and i' map to the central node in the 2-star. We then have just one mapping for the remaining indices. Thus, there is **1** valid choice.

Figure B.7 shows an example of two 4-cycles, six 3-stars, and one 2-stars. We can see that the other possible subgraphs on 3 or 4 nodes are immediately ruled out because they have too many or too few edges, violating edge requirements.

In the following, let $P(i, j, k)$ be the event that $F_{ijk} = F_{i'j'k'} = 1$. Observing Figure 2.4, we can see that in both possible ways in which a valid choice maps to a 4-cycle, then $P(i, j, k)$ holds when at least 3 edges in E' are present. Each edge in E' is independent, and is present with probability at most μ. Thus, $\Pr[P(i, j, k)] \le \mu^3$. Next, if a valid choice maps to a 3-star, then $P(i, j, k)$ implies at least 3 edges in E' are present. Thus, $\Pr[P(i, j, k) = 1] \le \mu^3$. Finally, if a valid choice maps to a 2-star, then $P(i, j, k) = 1$ if and only if 2 edges in E' are present. Thus, $\Pr[P(i, j, k) = 1] \le \mu^2$.

Putting this together,

$$\mathbb{V}\left[\sum_{i=1}^{n} w_i\right] \leq 2C_4(G)\mu^3 + 6S_3(G)\mu^3 + S_2(G)\mu^2.$$

Plugging into (B.6), we get

$$\begin{aligned}
&\mathbb{V}[\hat{f}_{\triangle}(G)] \\
&\leq \frac{1}{(1-\rho)^2}\left(\frac{1}{\mu}(2C_4(G)+6S_3(G)) + \frac{1}{\mu^2}S_2(G) + 2n\frac{d_{max}^2}{\mu^4\varepsilon_2^2}\right).
\end{aligned}$$

This establishes the result. □

Bounding the Variance in $\mathsf{ARRTwoNS}_{\triangle}$. In $\mathsf{ARRTwoNS}_{\triangle}$, for a fixed $v_i \in V$, we have $(v_j, v_k) \in M_i$ if and only if $j < k < i$, $(v_j, v_k) \in E'$, $(v_i, v_k) \in E'$, and $(v_i, v_k) \in E'$ from (2.9). Thus,

$$\begin{aligned}
\sum_{i=1}^{n} w_i &= \sum_{i=1}^{n} \sum_{\substack{1\leq j<k<i\leq n \\ a_{i,j}=1, a_{i,k}=1}} \mathbf{1}_{(v_j,v_k)\in M_i} \\
&= \sum_{1\leq j<k<i\leq n} a_{i,j}a_{i,k}\mathbf{1}_{(v_j,v_k)\in E'}\mathbf{1}_{(v_i,v_k)\in E'}\mathbf{1}_{(v_j,v_k)\in E'}.
\end{aligned}$$

Define the random variable $F_{ijk} = \mathbf{1}_{(v_j v_k)\in E'}\mathbf{1}_{(v_i,v_k)\in E'}\mathbf{1}_{(v_j,v_k)\in E'}$. Following the same steps as those in the proof of $\mathsf{ARROneNS}_{\triangle}$, we have

$$\begin{aligned}
\mathbb{V}\left[\sum_{i=1}^{n} w_i\right] &= \sum_{\substack{1\leq j<k<i\leq n \\ 1\leq j'<k'<i'\leq n}} a_{i,j}a_{i,k}a_{i',j'}a_{i',k'}\,\mathrm{Cov}(F_{ijk}, F_{i'j'k'}) \\
&\leq \sum_{i,i',j,j',k,k' \text{ valid}} \Pr[F_{ijk} = F_{i'j'k'} = 1].
\end{aligned}$$

As we showed in the proof for $\mathsf{ARROneNS}_{\triangle}$, each 4-cycle of G has at most 2 valid choices mapped to it, each 3-star of G has at most 6 valid choices mapped to it, and each 2-star of G has at most one valid choice mapped to it.

In the following, let $P(i,j,k)$ be the event that $F_{ijk} = F_{i'j'k'} = 1$. Observing Figure 2.4, we can see that for each possible mapping of a valid choice to a 4-cycle, five edges must be present in G' in order for $P(i,j,k) = 1$. Thus, $\Pr[P(i,j,k) = 1] \leq \mu^5$. For each possible mapping of a valid choice to a 3-star, five edges must be present in G' in order for $P(i,j,k) = 1$. Thus, $\Pr[P(i,j,k) = 1] \leq \mu^5$. For each possible mapping of a valid choice to a 2-star, three edges must be present in G' in order for $P(i,j,k) = 1$. Thus, $\Pr[P(i,j,k) = 1] \leq \mu^3$.

Plugging into (B.6), we get

$$\begin{aligned}
&\mathbb{V}[\hat{f}_{\triangle}(G)] \\
&\leq \frac{1}{(1-\rho)^2}\left(\frac{1}{\mu}(2C_4(G) + 6S_3(G)) + \frac{1}{\mu^3}S_2(G) + 2n\frac{d_{max}^2}{\mu^6\varepsilon_2^2}\right).
\end{aligned}$$

This establishes the result. $\square$

Lemma 2. *Let $c_{ij} = \sum_{l<i\leq n} a_{l,i}a_{l,j}$. Then,*

$$\sum_{i,j=1,i<j}^{n} c_{ij}^2 \leq 2C_4(G) + S_2(G).$$

Proof.

$$\begin{aligned}
\sum_{i,j=1,i<j}^{n} c_{ij}^2 &= \sum_{i,j=1,i<j}^{n} c_{ij} + \sum_{i,j=1,i<j}^{n} c_{ij}(c_{ij}-1) \\
&= S_2(G) + \sum_{i,j=1,i<j}^{n} c_{ij}(c_{ij}-1).
\end{aligned}$$

Let $C_{i-*-j-*-i}(G)$ be the number of 4-cycles in G such that the first and third nodes are v_i and v_j, respectively $(i<j)$, and the remaining two nodes have smaller indices than i. From middle nodes in 2-paths starting at v_i and ending at v_j, we can choose two nodes as the second and fourth nodes in the 4-cycles. c_{ij} is the number of nodes that have smaller IDs than v_i and are

connected to v_i and v_j. Thus, $C_{i-*-j-*-i}(G) = \binom{c_{ij}}{2}$. Therefore, we have

$$\sum_{i,j=1,i<j}^{n} c_{ij}^2 = S_2(G) + \sum_{i,j=1,i<j}^{n} 2C_{i-*-j-*-i}(G)$$
$$\leq S_2(G) + 2C_4(G).$$

The last inequality comes from the fact that two nodes with the largest indices may not be opposite to each other in some 4-cycles in G. □

B.8 Proof of Statements in Section 2.5

B.8.1 Proof of Theorem 10

Let $\mathbf{a}_i, \mathbf{a}'_i \in \{0,1\}^n$ be two neighbor lists that differ in one bit. Let $t_i{}'$, s'_i, and $w_i{}'$ be respectively the values of t_i (in line 8 of Algorithm 5), s_i (in line 9), and w_i (in line 10) when the neighbor list of user v_i is $\mathbf{a}'_i$. Let $\Delta w_i = |w_i{}' - w_i|$.

We assume that $|\mathbf{a}'_i| = |\mathbf{a}_i| + 1$ without loss of generality. Let $\bar{\mathbf{a}}_i, \bar{\mathbf{a}}'_i \in \{0,1\}^n$ be neighbor lists corresponding to $\mathbf{a}_i$ and $\mathbf{a}'_i$, respectively, after edge clipping. Note that $|\bar{\mathbf{a}}_i| = |\bar{\mathbf{a}}'_i| \leq \tilde{d}_i$. There are three cases for $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$:

1. $\bar{\mathbf{a}}_i$ is identical to $\bar{\mathbf{a}}'_i$ and $|\bar{\mathbf{a}}_i| = |\bar{\mathbf{a}}'_i| = \tilde{d}_i$.
2. $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in one bit and $|\bar{\mathbf{a}}'_i| = |\bar{\mathbf{a}}_i| + 1$.
3. $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in two bits and $|\bar{\mathbf{a}}_i| = |\bar{\mathbf{a}}'_i| = \tilde{d}_i$.

Note that the third case can happen when $|\mathbf{a}'_i| \geq \tilde{d}_i$. For example, assume that $n = 8$, $\tilde{d}_i = 4$, $\mathbf{a}_i = (1,1,0,1,0,1,1,1)$ and $\mathbf{a}'_i = (1,1,1,1,0,1,1,1)$. If we select four "1"s in the order of 3, 1, 4, 6, 8, 2, 7, and 5-th bit in the neighbor list, $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ will be: $\bar{\mathbf{a}}_i = (1,0,0,1,0,1,0,1)$ and $\bar{\mathbf{a}}'_i = (1,0,1,1,0,1,0,0)$, which differ in two bits.

If $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in one bit ($|\bar{\mathbf{a}}'_i| = |\bar{\mathbf{a}}_i| + 1$), then $t_i' - t_i \in [0, \kappa_i]$ and $s'_i - s_i \in [0, \tilde{d}_i]$, hence $\Delta w_i = |(t_i' - t_i) - \mu^* \rho (s'_i - s_i)| \leq \kappa_i$. If $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{a}}'_i$ differ in two bits ($|\bar{\mathbf{a}}_i| = |\bar{\mathbf{a}}'_i| = d_{max}$), then $t_i' - t_i \in [-\kappa_i, \kappa_i]$ and $s_i = s'_i = \binom{\tilde{d}_i}{2}$, hence $\Delta w_i \leq \kappa_i$.

Therefore, we always have $\Delta w_i \leq \kappa_i$ (if $\bar{\mathbf{a}}_i$ is identical to $\bar{\mathbf{a}}'_i$, $\Delta w_i = 0$). Since we add $\mathrm{Lap}(\frac{1}{\varepsilon_0})$ to d_i and $\mathrm{Lap}(\frac{\kappa_i}{\varepsilon_2})$ to w_i^*, the second round provides $(\varepsilon_0 + \varepsilon_2)$-edge LDP. The first round provides ε_1-edge LDP and we use only the lower-triangular part of $\mathbf{A}$. Thus, by sequential composition (Proposition 5) and Proposition 4, $\mathscr{R}_i$ satisfies $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$-edge LDP, and $(\mathscr{R}_1, \ldots, \mathscr{R}_n)$ satisfies $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$-relationship DP. $\square$

B.8.2 Proof of Theorem 11

Recall that $t_{i,j} = |\{(v_i, v_j, v_k) : a_{i,k} = 1, (v_j, v_k) \in M_i, j < k < i\}|$. Let $t'_{i,j} = |\{(v_i, v_j, v_k) : a_{i,k} = 1, (v_j, v_k) \in M_i\}|$. Then $t_{i,j} \leq t'_{i,j}$. Thus we have

$$\Pr(t_{i,j} > \kappa_i) \leq \Pr(t_{i,j} \geq \kappa_i) \leq \Pr(t'_{i,j} \geq \kappa_i).$$

Below we first prove (2.12) and (2.13). Then we prove (2.14).

Proof of (2.12) and (2.13). For each edge (v_i, v_j), we have $\sum_{k \neq i,j} \mathbf{1}_{(v_i, v_k) \in E} \leq \tilde{d}_i$. In $\mathsf{ARRFull}_\triangle$, each edge (v_j, v_k) is included in E' with probability at most μ, and all the events are independent. In $\mathsf{ARROneNS}_\triangle$, each of the edges (v_i, v_k) and (v_j, v_k) is included in E' with probability at most μ, and all the events are independent. Thus, $\Pr(t'_{i,j} \geq \kappa_i)$ is less than or equal to the probability that the number of successes in the binomial distribution $B(\tilde{d}_i, \mu^*)$ ($\mu^* = \mu$ in $\mathsf{ARRFull}_\triangle$ and μ^2 in $\mathsf{ARROneNS}_\triangle$) is larger than or equal to κ_i.

Let $X_{n,p}$ be a random variable representing the number of successes in the binomial distribution $B(n, p)$, and $F(\kappa_i; n, p) = \Pr(X_{n,p} \leq \kappa_i)$; i.e., F is a cumulative distribution function of $B(n,p)$. Since $\kappa_i \geq \mu^* \tilde{d}_i$, we have

$$
\begin{aligned}
&\Pr(t'_{i,j} \geq \kappa_i) \\
&\leq \Pr(X_{\tilde{d}_i,\mu^*} \geq \kappa_i) \\
&= F(\tilde{d}_i - \kappa_i; \tilde{d}_i, 1-\mu^*) \\
&\leq \exp\left[-\tilde{d}_i D\left(\frac{\tilde{d}_i - \kappa_i}{\tilde{d}_i} \parallel 1-\mu^*\right)\right] \text{(by Chernoff bound)} \\
&= \exp\left[-\tilde{d}_i D\left(\frac{\kappa_i}{\tilde{d}_i} \parallel \mu^*\right)\right],
\end{aligned}
$$

which proves (2.12) and (2.13) (as $\Pr(t_{i,j} > \kappa_i) \leq \Pr(t'_{i,j} \geq \kappa_i)$).

Proof of (2.14). Assume that $\kappa_i \geq \mu^2 \tilde{d}_i$ in $\mathsf{ARRTwoNS}_\triangle$. For each edge (v_i, v_j), we have $\sum_{k \neq i,j} \mathbf{1}_{(i,k) \in E} \leq \tilde{d}_i$. In addition, each of the edges (v_i, v_k) and (v_j, v_k) are included in E' with probability at most μ, and all the events are independent.

If (v_i, v_j) is included in E' (which happens with probability at most μ), $\Pr(t'_{i,j} \geq \kappa_i)$ is less than or equal to the probability that the number of successes in the binomial distribution $B(\tilde{d}_i, \mu^2)$ is larger than or equal to κ_i. Otherwise (i.e., if (v_i, v_j) is not included in E'), then $t'_{i,j} = 0$.

Thus, if $\kappa_i \geq \mu^2 \tilde{d}_i$, we have

$$
\begin{aligned}
&\Pr(t'_{i,j} \geq \kappa_i) \\
&\leq \mu \Pr(X_{\tilde{d}_i,\mu^2} \geq \kappa_i) \\
&= \mu F(\tilde{d}_i - \kappa_i; \tilde{d}_i, 1-\mu^2) \\
&\leq \mu \exp\left[-\tilde{d}_i D\left(\frac{\tilde{d}_i - \kappa_i}{\tilde{d}_i} \parallel 1-\mu^2\right)\right] \text{ (by Chernoff bound)} \\
&= \mu \exp\left[-\tilde{d}_i D\left(\frac{\kappa_i}{\tilde{d}_i} \parallel \mu^2\right)\right],
\end{aligned}
$$

and therefore (2.14) holds (as $\Pr(t_{i,j} > \kappa_i) \leq \Pr(t'_{i,j} \geq \kappa_i)$).

If $\mu^3\tilde{d}_i \leq \kappa_i < \mu^2\tilde{d}_i$, (2.14) can be written as: $\Pr(t_{i,j} > \kappa_i) \leq \mu \exp\left[-\tilde{d}_i D\left(\mu^2 \parallel \mu^2\right)\right] = \mu$.

This clearly holds because each edge (v_i, v_j) is included in E' with probability at most μ. $\square$

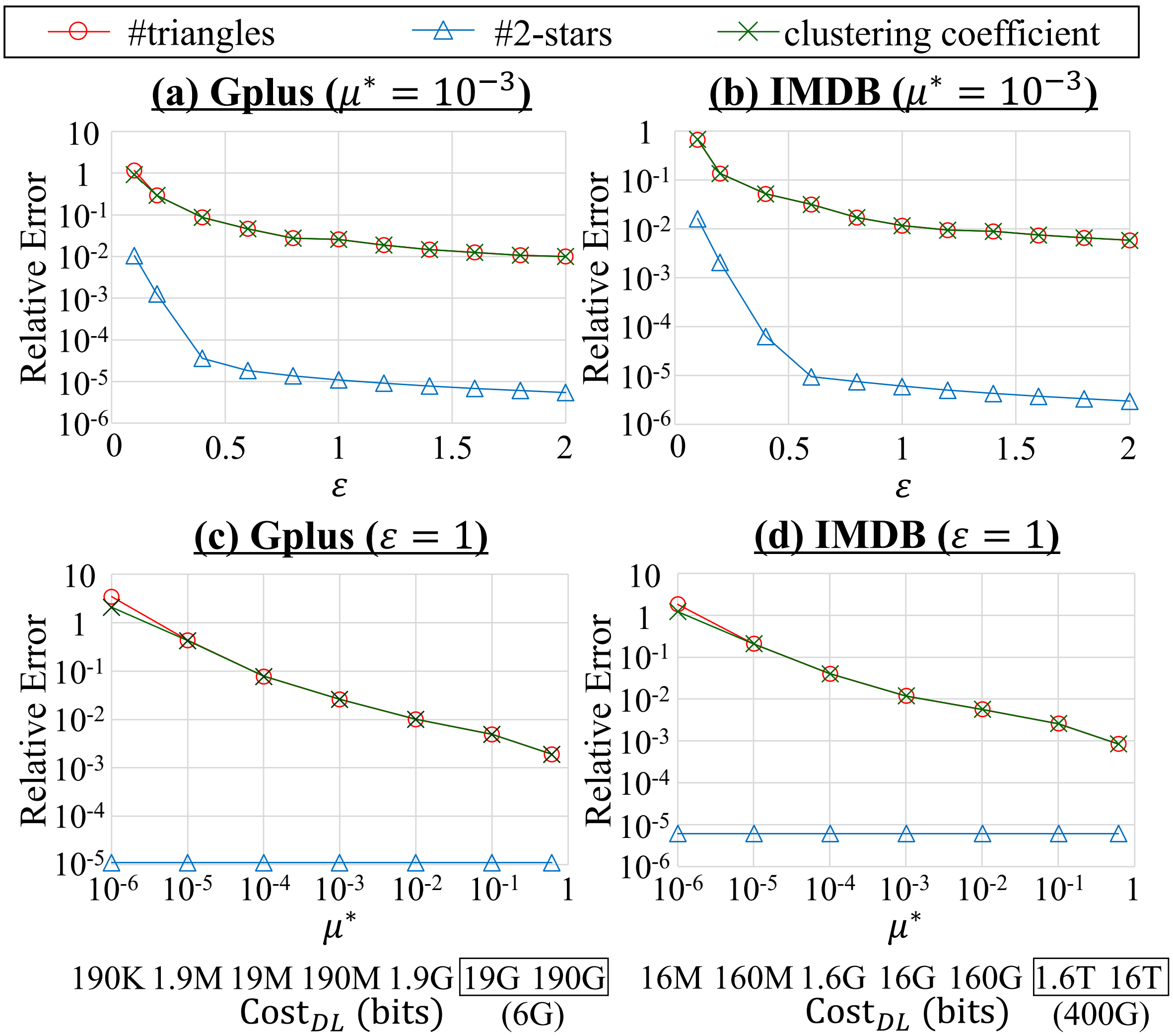


Figure B.3. Relative errors of #triangles, #2-stars, and the clustering coefficient in ARROneNS$_\triangle$ with double clipping. Cost$_{DL}$ is calculated by (2.10) (when $\mu^* \geq 0.1$, Cost$_{DL}$ can be 6 Gbits and 400 Gbits in Gplus and IMDB, respectively).

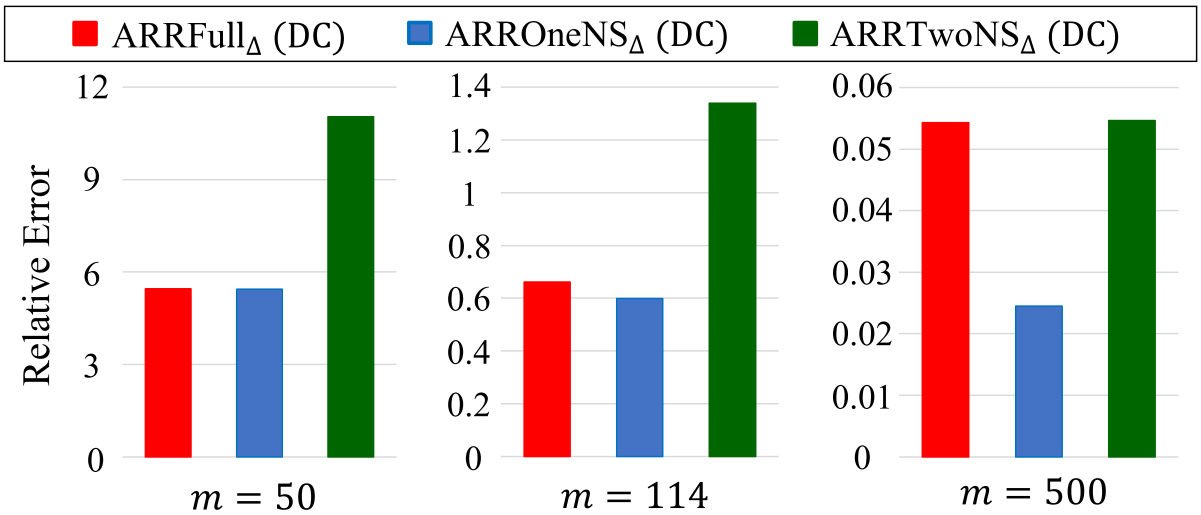


Figure B.4. Relative error of our three algorithms with double clipping in the BA graphs ($n = 107614$, $\varepsilon = 1$, $\mu^* = 10^{-3}$).

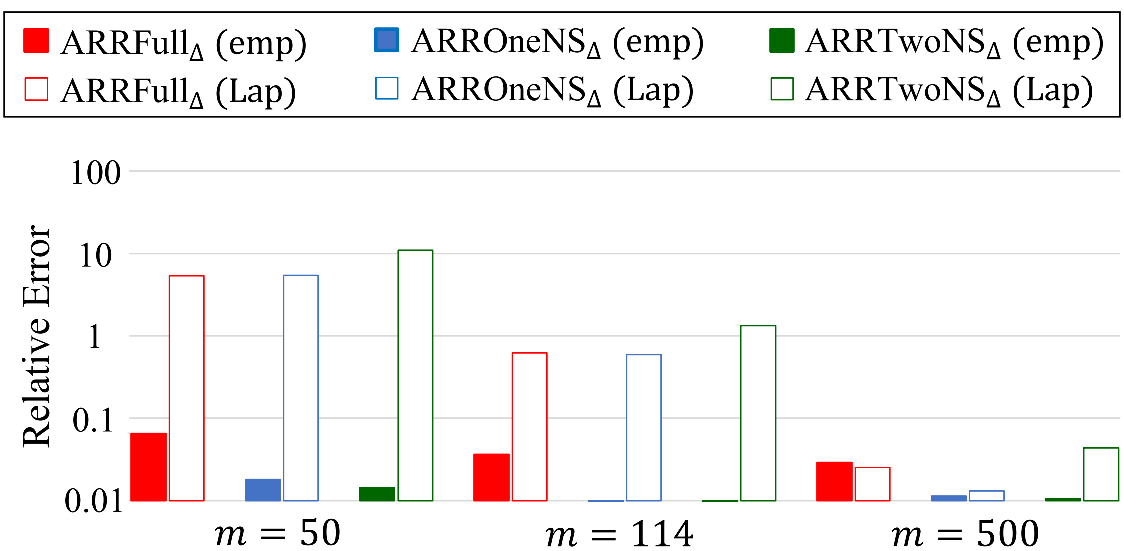


Figure B.5. Relative error of empirical estimation and the Laplacian noise in our three algorithms with double clipping in the BA graphs ($n = 107614$, $\varepsilon = 1$, $\mu^* = 10^{-3}$).

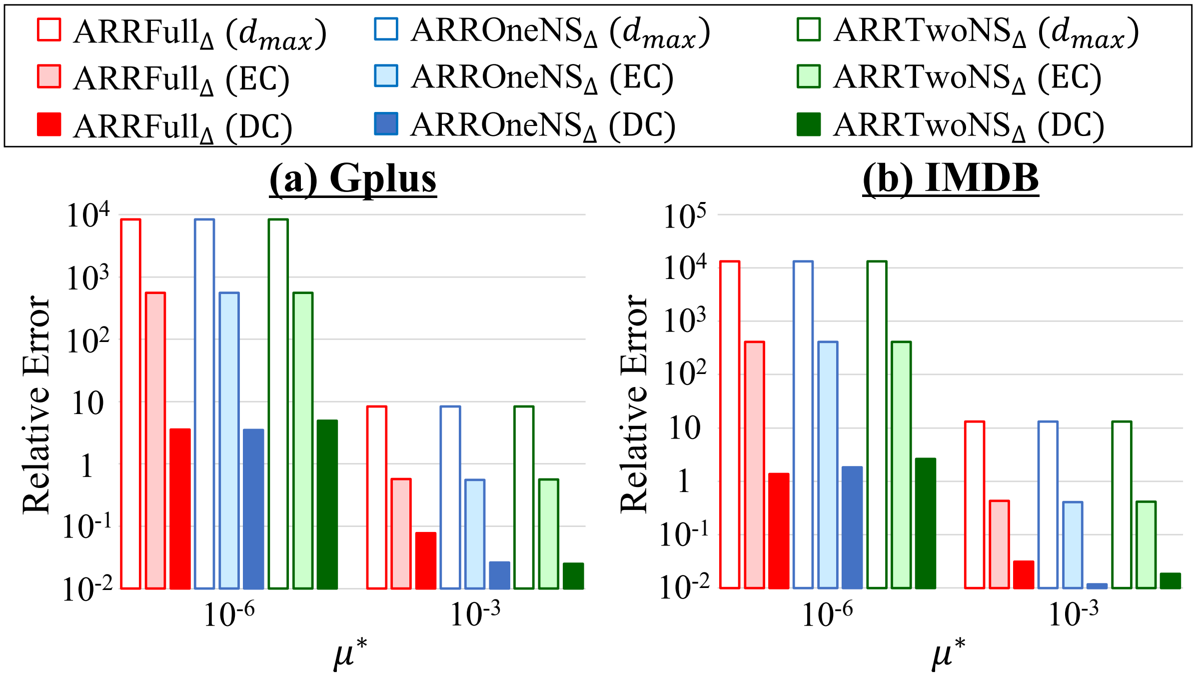


Figure B.6. Relative error of our three algorithms without clipping ("d_{max}"), with only edge clipping ("EC"), and double clipping ("DC") when $\varepsilon = 1$ and $\mu^* = 10^{-6}$ or 10^{-3} ($n = 107614$ in Gplus, $n = 896308$ in IMDB).

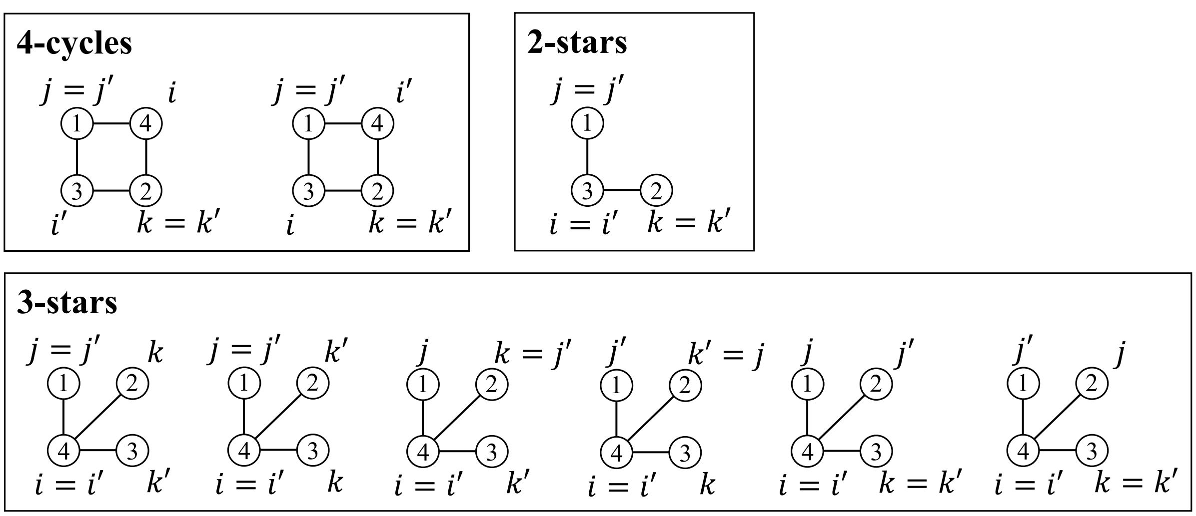


Figure B.7. Examples of two 4-cycles, six 3-stars, and one 2-stars.

Appendix C

C.1 Experiments of the Clustering Coefficient

In Section 3.7, we showed that our triangle counting algorithm $\mathsf{WShuffle}^*_{\triangle}$ accurately estimates the triangle count within one round. We also show that we can accurately estimate the clustering coefficient within one round by using $\mathsf{WShuffle}^*_{\triangle}$.

We calculated the clustering coefficient as follows. We used $\mathsf{WShuffle}^*_{\triangle}$ $(c = 1)$ for triangle counting and the one-round local algorithm in [102] with edge clipping [104] for 2-star counting. The 2-star algorithm works as follows. First, each user v_i adds the Laplacian noise $\mathrm{Lap}(\frac{1}{\varepsilon_1})$ and a non-negative constant $\eta \in \mathbb{R}_{\geq 0}$ to her degree d_i to obtain a noisy degree $\tilde{d}_i = d_i + \mathrm{Lap}(\frac{1}{\varepsilon_1}) + \eta$ with ε_1-edge LDP. If $\tilde{d}_i < d_i$, then v_i randomly removes $d_i - \lfloor \tilde{d}_i \rfloor$ neighbors from her neighbor list. This is called edge clipping in [104]. Then, v_i calculates the number $r_i \in \mathbb{Z}_{\geq 0}$ of 2-stars of which she is a center. User v_i adds $\mathrm{Lap}(\frac{\tilde{d}_i}{\varepsilon_2})$ to r_i to obtain a noisy 2-star count $\tilde{r}_i = r_i + \mathrm{Lap}(\frac{\tilde{d}_i}{\varepsilon_2})$. Because the sensitivity of the k-star count is $\binom{\tilde{d}_i}{k-1}$, the noisy 2-star count $\tilde{r}_i$ provides ε_2-edge LDP. User v_i sends the noisy degree $\tilde{d}_i$ and the noisy 2-star count $\tilde{r}_i$ to the data collector. Finally, the data collector estimates the 2-star count as $\sum_{i=1}^{n} \tilde{r}_i$. By composition, this algorithm provides $(\varepsilon_1 + \varepsilon_2)$-edge LDP. As with [104], we set $\eta = 150$ and divided the total privacy budget ε as $\varepsilon_1 = \frac{\varepsilon}{10}$ and $\varepsilon_1 = \frac{9\varepsilon}{10}$.

Let $\hat{f}^{\triangle}(G)$ be the estimate of the triangle count by $\mathsf{WShuffle}^*_{\triangle}$ and $\hat{f}^{2*}(G)$ be the estimate of the 2-star count by the above algorithm. Then we estimated the clustering coefficient as $\frac{3\hat{f}^{\triangle}(G)}{\hat{f}^{2*}(G)}$.

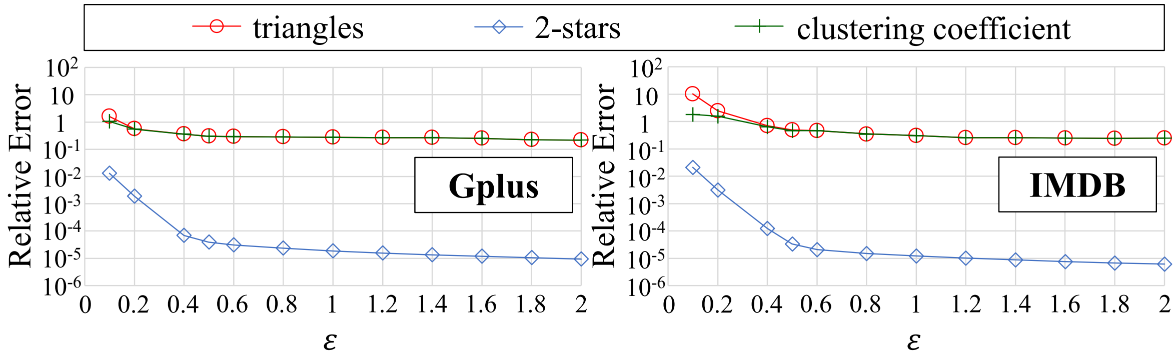


Figure C.1. Relative errors of the triangle count, 2-star count, and clustering coefficient when $\mathsf{WShuffle}^*_\triangle$ and the one-round local 2-star algorithm in [102] with edge clipping are used ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$).

Figure C.1 shows the relative errors of the triangle count, 2-star count, and clustering coefficient in Gplus and IMDB. We observe that the relative error of the clustering coefficient is almost the same as that of the triangle count. This is because the 2-star algorithm is very accurate, as shown in Figure C.1. 2-stars are much easier to count than triangles in the local model, as each user can count her 2-stars. As a result, the error in the clustering coefficient is mainly caused by the error in $\hat{f}^\triangle(G)$, which explains the results in Figure C.1.

In Figure C.1, we use the privacy budget ε for both $\hat{f}^\triangle(G)$ and $\hat{f}^{2*}(G)$. In this case, we need 2ε to calculate the clustering coefficient. However, as shown in Figure C.1, we can accurately estimate the 2-star count with a very small ε; e.g., the relative error is around 10^{-2} when $\varepsilon = 0.1$. Therefore, we can accurately calculate the clustering coefficient with a very small additional budget by using such a small ε for 2-stars.

In summary, our triangle algorithm $\mathsf{WShuffle}^*_\triangle$ is useful for accurately calculating the clustering coefficient within one round.

C.2 Comparison with Two-Round Local Algorithms

In this work, we focus on one-round algorithms because multi-rounds algorithms require a lot of user effort and synchronization. However, it is interesting to see how our one-round algorithms compare with the existing two-round local algorithms [102, 104] in terms of accuracy, as the existing two-round algorithms provide high accuracy. Since they focus on triangle counting, we focus on this task.

We evaluate the two-round local algorithm in [104] because it outperforms [102] in terms of both the accuracy and communication efficiency. The algorithm in [104] works as follows. At the first round, each user v_i obfuscates bits $a_{i,1},\dots,a_{i,i-1}$ for smaller user IDs in her neighbor list $\mathbf{a}_i$ (i.e., lower triangular part of $\mathbf{A}$) by the ARR and sends the noisy neighbor list to the data collector. The data collector constructs a noisy graph $G'=(V,E')$ from the noisy neighbor lists. At the second round, each user v_i downloads some noisy edges $(v_j,v_k)\in E'$ from the data collector and counts noisy triangles (v_i,v_j,v_k) so that only one edge (v_j,v_k) is noisy. User v_i adds the Laplacian noise to the noisy triangle count and sends it to the data collector. Finally, the data collector calculates an unbiased estimate of the triangle count. This algorithm provides ε-edge LDP. The authors in [104] propose some strategies to select noisy edges to download at the second round. We use a strategy to download noisy edges $(v_j,v_k)\in E'$ such that a noisy edge is connected from v_k to v_i (i.e., $(v_i,v_k)\in E'$) because it provides the best performance.

The algorithm in [104] controls the trade-off between the accuracy and the download cost (i.e., the size of noisy edges) at the second round by changing the sampling probability p_0 in the ARR. It is shown in [104] that when $p_0=1$, the MSE is $O(nd_{max}^3)$ and the download cost of each user is $\frac{(n-1)(n-2)}{2}$ bits. In contrast, when $p_0=O(n^{-1/2})$, the MSE is $O(n^2d_{max}^3)$ and the download cost is $O(n\log n)$. We evaluated these two settings. For the latter setting, we set $p_0=\frac{1}{q\sqrt{n}}$ where $q=\frac{e^\varepsilon}{e^\varepsilon+1}$ so that the download cost is $n\log n$ bits. We denote the two-round algorithm with $p_0=1$ and $\frac{1}{q\sqrt{n}}$ by 2R-Large$_\triangle$ and 2R-Small$_\triangle$, respectively. 2R-Large$_\triangle$ requires a larger download cost.

Figure C.2 shows the results. We observe that our WShuffle$^*_\triangle$ is outperformed by 2R-Large$_\triangle$. This is expected, as WShuffle$^*_\triangle$ and 2R-Large$_\triangle$ provide the MSE of $O(n^2)$ and $O(n)$, respectively (when we ignore d_{max}). However, 2R-Large$_\triangle$ is impractical because it requires a too large download cost: 6G and 400G bits per user in Gplus and IMDB, respectively. 2R-Small$_\triangle$ is much more efficient (1.8M and 18M bits in Gplus and IMDB, respectively), and our WShuffle$^*_\triangle$

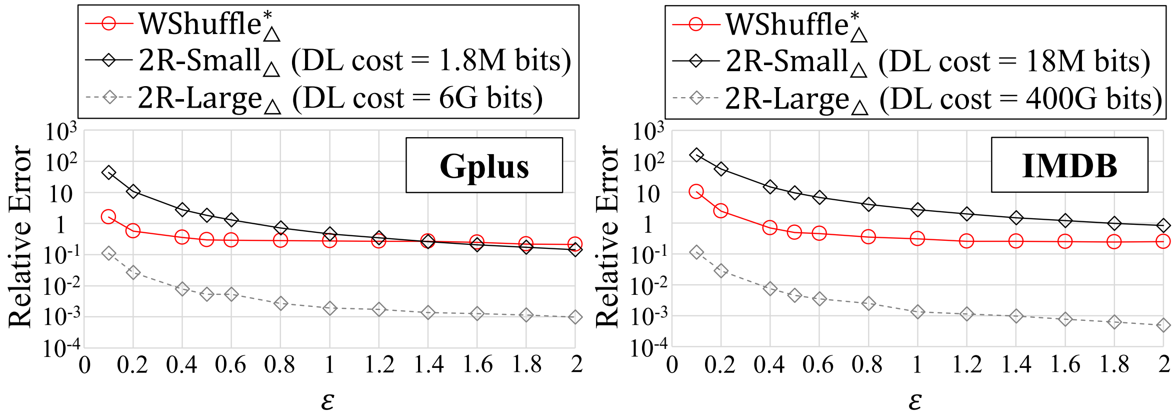


Figure C.2. Comparison with the two-round local algorithm in [104]. The download costs of 2R-Small$_\triangle$ and 2R-Large$_\triangle$ are $n \log n$ and $\frac{(n-1)(n-2)}{2}$ bits, respectively ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$).

is comparable to or outperforms 2R-Small$_\triangle$[1]. This is also consistent with the theoretical results because both WShuffle$^*_\triangle$ and 2R-Small$_\triangle$ provide the MSE of $O(n^2)$.

In summary, our WShuffle$^*_\triangle$ is comparable to the two-round local algorithm in [104] (2R-Small$_\triangle$), which requires a lot of user effort and synchronization, in terms of accuracy.

C.3 Comparison between the Numerical Bound and the Closed-form Bound

In Section 3.7, we used the numerical upper bound in [88] for calculating ε in the shuffle model. Here, we compare the numerical bound with the closed-form bound in Theorem 12.

Figure C.3 shows the results for WShuffle$^*_\triangle$, WShuffle$_\triangle$, and WShuffle$_\square$. We observe that the numerical bound provides a smaller relative error than the closed-form bound when ε is small. However, when $\varepsilon \geq 1$, the relative error is almost the same between the numerical bound and the closed-form bound. This is because when $\varepsilon \geq 1$, the corresponding ε_L is close to the maximum value $\log(\frac{n}{16\log(2/\delta)})$ (= 5.86 in Gplus and 7.98 in IMDB) in both cases. Thus, for a large ε, the closed-form bound is sufficient. For a small ε, the numerical bound is preferable.

[1] As with ARR$_\triangle$, 2R-Large$_\triangle$ and 2R-Small$_\triangle$ provide ε-edge DP (rather than 2ε-edge DP) because it uses only the lower-triangular part of $\mathbf{A}$. However, our conclusion is the same even if we double ε for only WShuffle$^*_\triangle$.

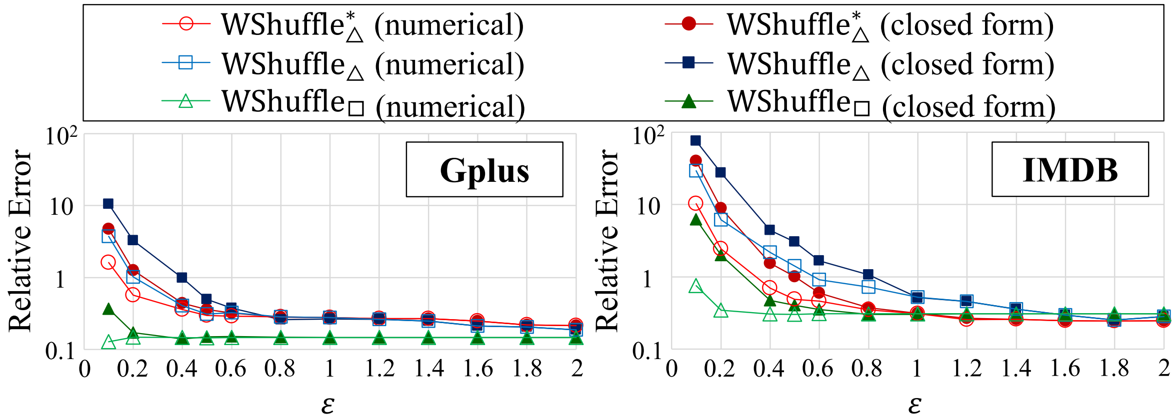


Figure C.3. Numerical bound vs. closed-form bound ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$).

Table C.1. Statistics of Gplus and the BA graphs ($n = 107614$).

	d_{avg}	d_{max}	#triangles	#4-cycles
Gplus	227.4	20127	1.07×10^9	1.42×10^{12}
BA ($m = 100$)	199.8	5361	1.56×10^7	5.31×10^9
BA ($m = 200$)	399.3	7428	9.86×10^7	6.21×10^{10}

C.4 Experiments on the Barabási-Albert Graphs

In Section 3.7, we used Gplus and IMDB as datasets. We also evaluated our algorithms using synthetic datasets based on the BA (Barabási-Albert) graph model [16], which has a power-law degree distribution.

The BA graph model generates a graph by adding new nodes one at a time. Each new node has $m \in \mathbb{N}$ new edges, and each new edge is randomly connected an existing node with probability proportional to its degree. The average degree is almost $d_{avg} = 2m$, and most users' degrees are m. We set $m = 100$ or 200 and used the NetworkX library [96] (barabasi_albert_graph function) to generate a synthetic graph based on the BA model. For the number n of users, we set $n = 107614$ (same as Gplus) to compare the results between Gplus and the BA graphs. Table C.1 shows some statistics of Gplus and the BA graphs. It is well known that the BA model has a low clustering coefficient [101]. Thus, the BA graphs have much smaller triangles and 4-cycles than Gplus.

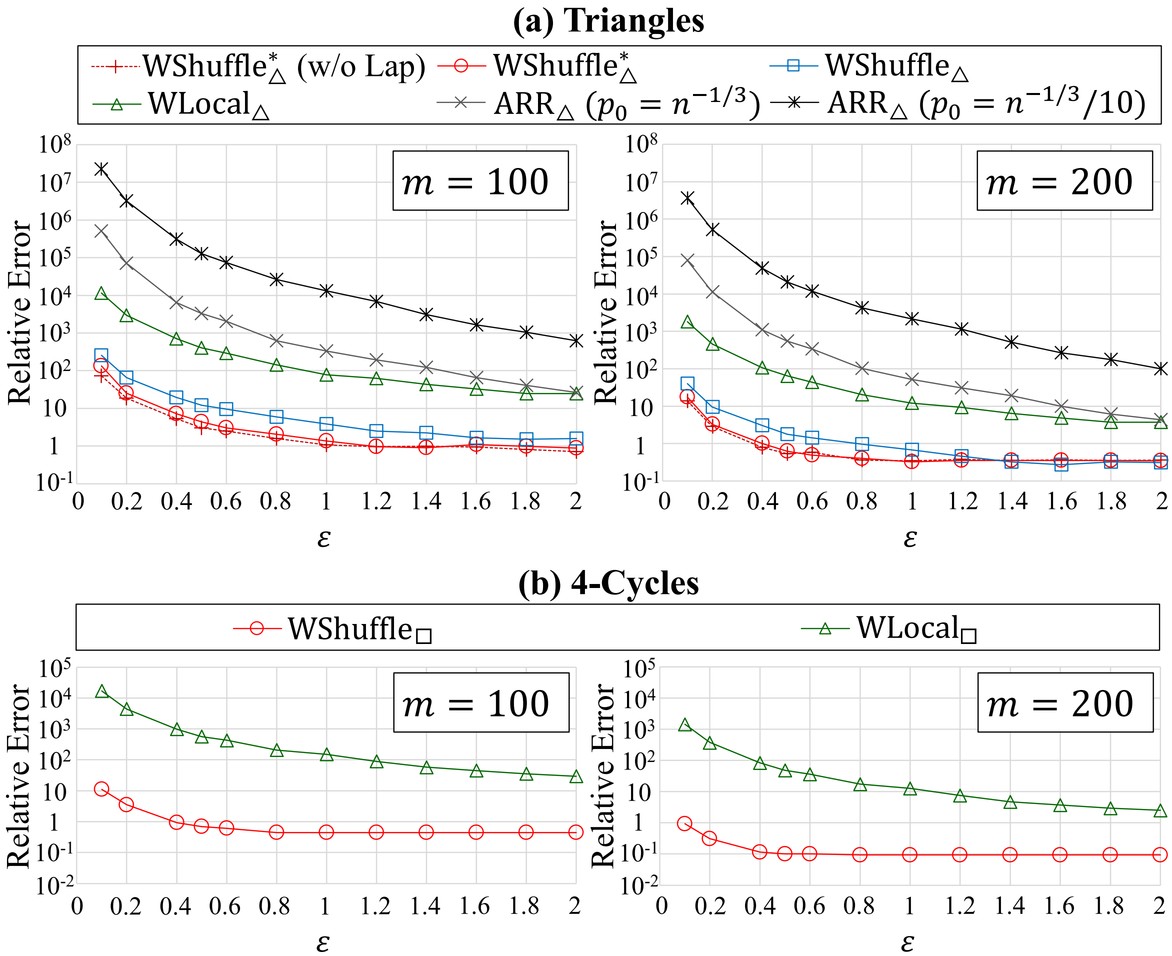


Figure C.4. Relative error in the BA graph data ($n = 107614$, $c = 1$). p_0 is the sampling probability in the ARR.

Figure C.4 shows the results in the BA graphs. We observe that the relative error is smaller when $m = 200$. This is because the BA graph with $m = 200$ includes larger numbers of true triangles and 4-cycles, as shown in Table C.1. In this case, the denominator in the relative error is larger, and consequently the relative error becomes smaller. By Figures 3.6 and C.4, the relative error in the BA graph with $m = 100$ is larger than the relative error in Gplus. The reason for this is the same – Gplus includes larger numbers of triangles and 4-cycles, as shown in Table C.1. These results show that the relative error tends to be smaller in a dense graph that includes a larger number of subgraphs.

Figures C.4 shows that when $\varepsilon = 1$, the relative errors of WShuffle$^*_\triangle$ ($m = 100$), WShuf-

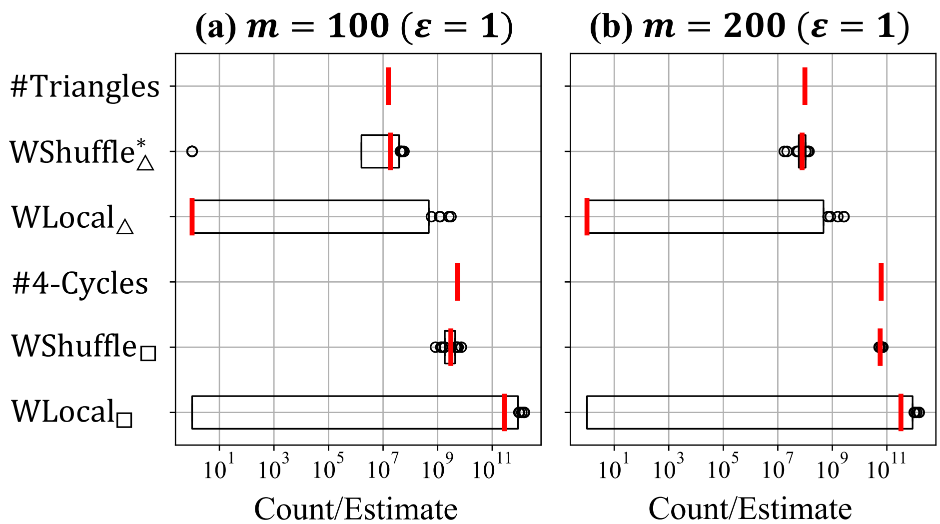


Figure C.5. Box plots of counts/estimates in the BA graph data ($n = 107614$, $c = 1$). #Triangles and #4-Cycles represent the true triangle and 4-cycle counts, respectively. The box plot of each algorithm represents the median (red), lower/upper quartile, and outliers (circles) of 20 estimates. The leftmost values are smaller than 1.

fle$_\square$ ($m = 100$), WShuffle$^*_\triangle$ ($m = 200$), and WShuffle$_\square$ ($m = 200$) are 1.36, 0.447, 0.323, and 0.0928, respectively. Although the relative error of WShuffle$^*_\triangle$ is about 1 when $\varepsilon = 1$ and $m = 100$, we argue that it is still useful for calculating a rough estimate of the triangle count. To explain this, we show box plots of counts or estimates in the BA graphs in Figure C.5. This figure shows that the true triangle count is about 10^7 and that WShuffle$^*_\triangle$ ($m = 100$) successfully calculates a rough estimate ($10^6 \sim 10^8$) in most cases (15 out of 20 cases). WShuffle$_\square$ ($m = 100$), WShuffle$^*_\triangle$ ($m = 200$), and WShuffle$_\square$ ($m = 200$) are much more accurate and successfully calculate an estimate in all cases. In contrast, the local algorithms WLocal$_\triangle$ and WLocal$_\square$ fail to calculate a rough estimate.

In summary, our shuffle algorithms significantly outperform the local algorithms and calculate a (rough) estimate of the triangle/4-cycle count with a reasonable privacy budget (e.g., $\varepsilon = 1$) in the BA graph data.

C.5 Experiments on the Bipartite Graphs

As described in Section 3.1, the 4-cycle count is useful for measuring the clustering tendency in a bipartite graph where no triangles appear. Therefore, we also evaluated our 4-cycle counting algorithms using bipartite graphs generated from Gplus and IMDB.

Specifically, for each dataset, we randomly divided all users into two groups with equal number of users. The number of users in each group is 53807 in Gplus and 448154 in IMDB. Then, we constructed a bipartite graph by removing edges within each group. We refer to the bipartite versions of Gplus and IMDB as the *bipartite Gplus* and *bipartite IMDB*, respectively. Using these datasets, we evaluated the relative errors of $\mathsf{WShuffle}_{\square}$ and $\mathsf{WLocal}_{\square}$. Note that we did not evaluate the triangle counting algorithms, because there are no triangles in these graphs.

Figure C.6 shows the results. We observe that $\mathsf{WShuffle}_{\square}$ significantly outperforms $\mathsf{WLocal}_{\square}$ in these datasets. Compared to Figure 3.6(b), the relative error of $\mathsf{WShuffle}_{\square}$ is a bit larger in the bipartite graph data. For example, when $\varepsilon = 1$, the relative error of $\mathsf{WShuffle}_{\square}$ is 0.147, 0.308, 0.217, and 0.626 in Gplus, IMDB, the bipartite Gplus, and the bipartite IMDB, respectively. This is because the 4-cycle count is reduced by removing edges within each group. In Gplus, the 4-cycle count is reduced from 1.42×10^{12} to 1.77×10^{11}. In IMDB, it is reduced from 2.37×10^{12} to 2.96×10^{11}. Consequently, the denominator in the relative error becomes smaller, and the relative error becomes larger.

Although the relative error of $\mathsf{WShuffle}_{\square}$ with $\varepsilon = 1$ is about 0.6 in the bipartite IMDB, $\mathsf{WShuffle}_{\square}$ still calculates a rough estimate of the 4-cycle count. Figure C.7 shows box plots of counts or estimates in the bipartite graph data. This figure shows that $\mathsf{WLocal}_{\square}$ fails to estimate the 4-cycle count. In contrast, $\mathsf{WShuffle}_{\square}$ successfully calculates a rough estimate of the 4-cycle count in all cases.

In summary, our $\mathsf{WShuffle}_{\square}$ significantly outperforms $\mathsf{WLocal}_{\square}$ and accurately counts 4-cycles in the bipartite graphs as well.

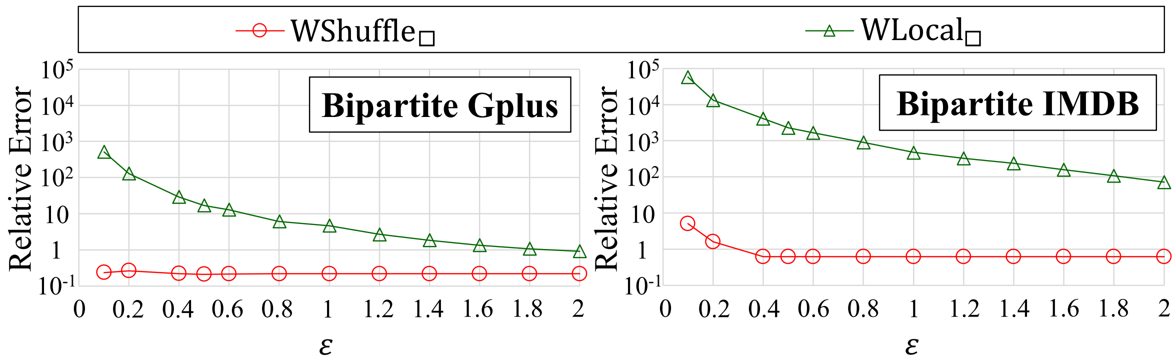


Figure C.6. Relative error in the bipartite graph data ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$).

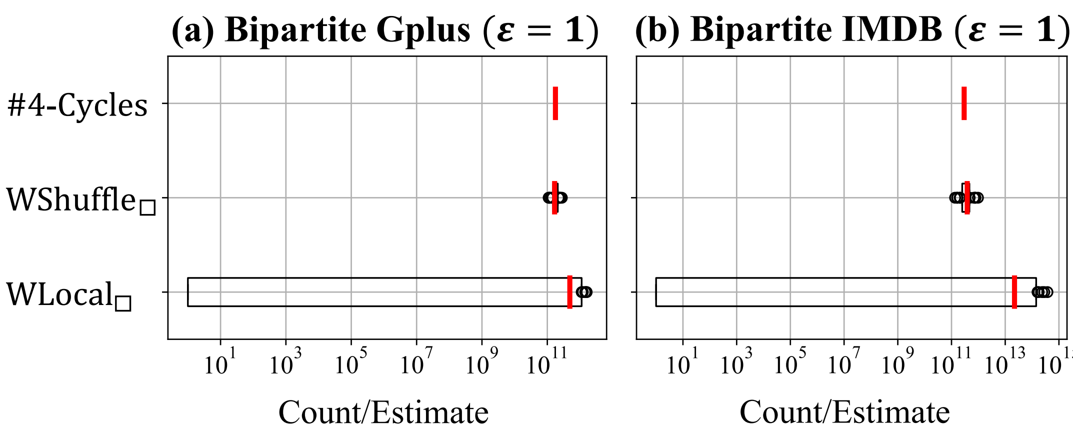


Figure C.7. Box plots of counts/estimates in the bipartite graph data ($n = 107614$ in Gplus, $n = 896308$ in IMDB, $c = 1$). #4-Cycles represents the true 4-cycle count. Each box plot represents the median (red), lower/upper quartile, and outliers (circles) of 20 estimates. The leftmost values are smaller than 1.

C.6 Standard Error of the Average Relative Error

In Section 3.7, we evaluated the average relative error over 20 runs for each algorithm. In this appendix, we evaluate the standard error of the average relative error.

Figure C.8 shows the standard error of the average relative error in Figure 3.6. We observe that the standard error is small. For example, as shown in Table 3.4 (a), the average relative error of WShuffle$^{*}_{\triangle}$ is 0.298 ($\varepsilon = 0.5$) or 0.277 ($\varepsilon = 1$) in Gplus. Figure C.8 shows that the corresponding standard error is 0.078 ($\varepsilon = 0.5$) or 0.053 ($\varepsilon = 1$). Thus, we conclude that 20 runs are sufficient in our experiments.

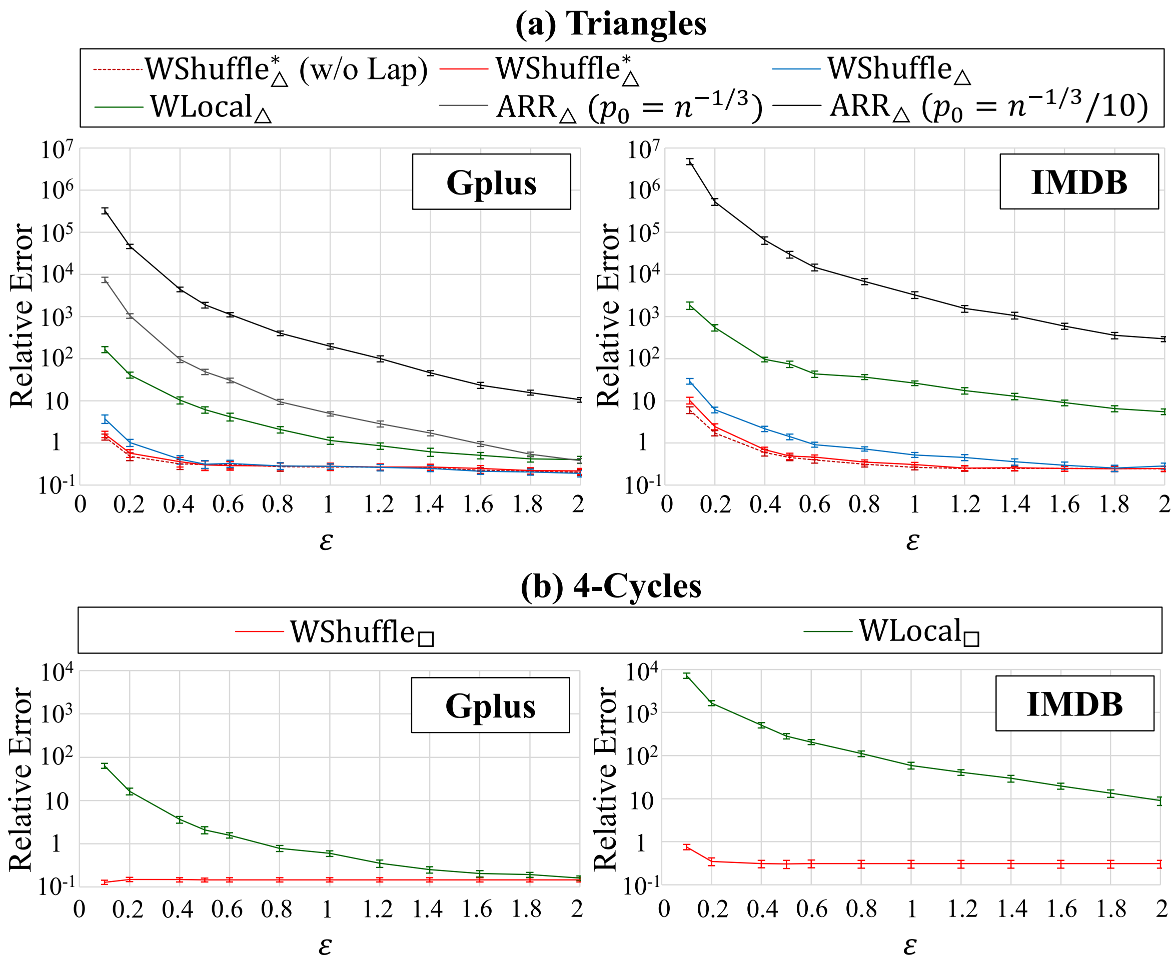


Figure C.8. Standard error of the average relative error in Figure 3.6. Each error bar represents $\pm$ standard error.

C.7 MSE of the Existing One-Round Local Algorithms

Here, we show the MSE of the existing one-round local algorithms $\mathsf{ARR}_\triangle$ [104] and $\mathsf{RR}_\triangle$ [102]. Specifically, we prove the MSE of $\mathsf{ARR}_\triangle$ because $\mathsf{ARR}_\triangle$ includes $\mathsf{RR}_\triangle$ as a special case; i.e., the MSE of $\mathsf{RR}_\triangle$ is immediately derived from that of $\mathsf{ARR}_\triangle$.

We also note that the MSE of $\mathsf{RR}_\triangle$ is proved in [102] under the assumption that a graph is generated from the Erdös-Rényi graph model [16]. However, this assumption does not hold in practice, because the Erdös-Rényi graph does not have a power-law degree distribution. In contrast, we prove the MSE of $\mathsf{ARR}_\triangle$ (hence $\mathsf{RR}_\triangle$) without making any assumption on graphs.

Algorithm. First, we briefly explain $\mathsf{ARR}_\triangle$. In this algorithm, each user v_i obfuscates her neighbor list $\mathbf{a}_i \in \{0,1\}^n$ using the ARR (Asymmetric Randomized Response) whose input domain and output range are $\{0,1\}$. Specifically, the ARR has two parameters $\varepsilon \in \mathbb{R}_{\geq 0}$ and $\mu \in [0, \frac{e^\varepsilon}{e^\varepsilon+1}]$. Given 1 (resp. 0), the ARR outputs 1 with probability μ (resp. $\mu e^{-\varepsilon}$). This mechanism is equivalent to ε-RR followed by edge sampling, which samples each 1 with probability p_0 satisfying $\mu = \frac{e^\varepsilon}{e^\varepsilon+1} p_0$. User v_i applies the ARR to bits $a_{i,1}, \ldots, a_{i,i-1}$ for smaller user IDs in her neighbor list $\mathbf{a}_i$ (i.e., lower triangular part of $\mathbf{A}$) and sends the noisy bits to the data collector. Then, the data collector constructs a noisy graph G^* based on the noisy bits.

The data collector counts triangles, 2-edges (three nodes with two edges), 1-edge (three nodes with one edge), and no-edges (three nodes with no edges) in the noisy graph G^*. Let $m_3^*, m_2^*, m_1^*, m_0^* \in \mathbb{Z}_{\geq 0}$ be the numbers of triangles, 2-edges, 1-edge, and no-edges, respectively, in G^*. Note that $m_3^* + m_2^* + m_1^* + m_0^* = \binom{n}{3}$. Finally, the data collector estimates the number $f^\triangle(G)$ of triangles as follows:

$$\hat{f}^\triangle(G) = \frac{1}{(e^\varepsilon - 1)^3}(e^{3\varepsilon}\hat{m}_3 - e^{2\varepsilon}\hat{m}_2 + e^\varepsilon \hat{m}_1 - \hat{m}_0), \tag{C.1}$$

where

$$\hat{m}_3 = \frac{m_3^*}{p_0^3} \tag{C.2}$$

$$\hat{m}_2 = \frac{m_2^*}{p_0^2} - 3(1-p_0)\hat{m}_3 \tag{C.3}$$

$$\hat{m}_1 = \frac{m_1^*}{p_0} - 3(1-p_0)^2\hat{m}_3 - 2(1-p_0)\hat{m}_2 \tag{C.4}$$

$$\hat{m}_0 = \binom{n}{3} - \hat{m}_3 - \hat{m}_2 - \hat{m}_1. \tag{C.5}$$

$\mathsf{RR}_\triangle$ is a special case of $\mathsf{ARR}_\triangle$ where $\mu = \frac{e^\varepsilon}{e^\varepsilon+1}$ ($p_0 = 1$), i.e. without edge sampling.

Privacy and Time Complexity. The ARR is equivalent to ε-RR followed by edge sampling, as explained above. Therefore, $\mathsf{ARR}_\triangle$ provides ε-edge LDP by the post-processing invariance [73].

The time complexity of $\mathsf{ARR}_\triangle$ is dominated by counting the number m_3 of triangles in the noisy graph G^*. The expectation of m_3 is upper bounded as $\mathbb{E}[\mu_3^*] \leq \mu^3 n^3$, as each user-pair has an edge in G^* with probability at most μ. Thus, the time complexity of $\mathsf{ARR}_\triangle$ can be expressed as $O(\mu^3 n^3)$. This is $O(n^2)$ when $\mu^3 = O(\frac{1}{n})$.

MSE. Below, we analyze the MSE of $\mathsf{ARR}_\triangle$. First, we show that $\mathsf{ARR}_\triangle$ provides an unbiased estimate[2]:

Theorem 25. *In $\mathsf{ARR}_\triangle$, $\mathbb{E}[\hat{f}^\triangle(G)] = f^\triangle(G)$.*

Proof. Let $m_3, m_2, m_1, m_0 \in \mathbb{Z}_{\geq 0}$ be the numbers of triangles, 2-edges, 1-edge, and no-edges, respectively, in the noisy graph G' obtained by applying only ε-RR. Because the ARR independently samples each edge with probability p_0, we have:

$$\mathbb{E}[m_3^*] = p_0^3 m_3 \tag{C.6}$$

$$\mathbb{E}[m_2^*] = 3p_0^2(1-p_0)m_3 + p_0^2 m_2 \tag{C.7}$$

$$\mathbb{E}[m_1^*] = 3p_0(1-p_0)^2 m_3 + 2p_0(1-p_0)m_2 + p_0 m_1. \tag{C.8}$$

By (C.1), we have:

$$\begin{aligned}
&\mathbb{E}[\hat{f}^\triangle(G)] \\
&= \frac{1}{(e^\varepsilon - 1)^3}(e^{3\varepsilon}\mathbb{E}[\hat{m}_3] - e^{2\varepsilon}\mathbb{E}[\hat{m}_2] + e^\varepsilon \mathbb{E}[\hat{m}_1] - \mathbb{E}[\hat{m}_0]) \\
&= \frac{1}{(e^\varepsilon - 1)^3}(e^{3\varepsilon}\mathbb{E}[\hat{m}_3] - e^{2\varepsilon}\mathbb{E}[\hat{m}_2] + e^\varepsilon \mathbb{E}[\hat{m}_1] - \mathbb{E}[\hat{m}_0]).
\end{aligned} \tag{C.9}$$

By (C.2), (C.3), (C.4), (C.5), (C.6), (C.7), and (C.8), we have:

$$\begin{aligned}
\mathbb{E}[\hat{m}_3] &= \tfrac{\mathbb{E}[m_3^*]}{p_0^3} = \mathbb{E}[m_3] \\
\mathbb{E}[\hat{m}_2] &= \tfrac{\mathbb{E}[m_2^*]}{p_0^2} - 3(1-p_0)\mathbb{E}[\hat{m}_3] \\
&= 3(1-p_0)m_3 + m_2 - 3(1-p_0)m_3 \\
&= \mathbb{E}[m_2]
\end{aligned}$$

[2]It is informally explained in [104] that the estimate of $\mathsf{ARR}_\triangle$ is unbiased. We formalize their claim.

$$\begin{aligned}
\mathbb{E}[\hat{m}_1] &= \tfrac{\mathbb{E}[m_1^*]}{p_0} - 3(1-p_0)^2\mathbb{E}[\hat{m}_3] - 2(1-p_0)\mathbb{E}[\hat{m}_2] \\
&= 3(1-p_0)^2\mathbb{E}[m_3] + 2(1-p_0)\mathbb{E}[m_2] + \mathbb{E}[m_1] \\
&\quad - 3(1-p_0)^2\mathbb{E}[m_3] - 2(1-p_0)\mathbb{E}[m_2] \\
&= \mathbb{E}[m_1] \\
\mathbb{E}[\hat{m}_0] &= \tbinom{n}{3} - \mathbb{E}[\hat{m}_3] - \mathbb{E}[\hat{m}_2] - \mathbb{E}[\hat{m}_1] \\
&= \tbinom{n}{3} - \mathbb{E}[m_3] - \mathbb{E}[m_2] - \mathbb{E}[m_1] \\
&= \mathbb{E}[m_0].
\end{aligned}$$

Thus, the equality (C.9) can be written as follows:

$$\begin{aligned}
&\mathbb{E}[\hat{f}^{\triangle}(G)] \\
&= \frac{1}{(e^{\varepsilon}-1)^3}(e^{3\varepsilon}\mathbb{E}[m_3] - e^{2\varepsilon}\mathbb{E}[m_2] + e^{\varepsilon}\mathbb{E}[m_1] - \mathbb{E}[m_0]).
\end{aligned} \tag{C.10}$$

Finally, we use the following lemma:

Lemma 3. *[Proposition 2 in [102]]*

$$\mathbb{E}\left[\tfrac{e^{3\varepsilon}}{(e^{\varepsilon}-1)^3}m_3 - \tfrac{e^{2\varepsilon}}{(e^{\varepsilon}-1)^3}m_2 + \tfrac{e^{\varepsilon}}{(e^{\varepsilon}-1)^3}m_1 - \tfrac{1}{(e^{\varepsilon}-1)^3}m_0\right] = f_{\triangle}(G). \tag{C.11}$$

See [102] for the proof of Lemma 3. By (C.10) and Lemma 3, we have $\mathbb{E}[\hat{f}^{\triangle}(G)] = f_{\triangle}(G)$. □

Next, we show the MSE (= variance) of $\mathsf{ARR}_{\triangle}$:

Theorem 26. *When we treat ε as a constant, $\mathsf{ARR}_{\triangle}$ provides the following utility guarantee:*

$$\mathit{MSE}(\hat{f}^{\triangle}(G)) = \mathbb{V}[\hat{f}^{\triangle}(G)] = O\left(\frac{n^4}{\mu^6}\right).$$

By Theorem 26, the MSE of $\mathsf{ARR}_{\triangle}$ is $O(n^6)$ when we set $\mu^3 = O(\frac{1}{n})$ so that the time complexity is $O(n^2)$. The MSE of $\mathsf{RR}_\triangle$ ($\mu = 1$) is $O(n^4)$. Below, we prove Theorem 26.

Proof. Let $d_3 = \frac{e^{3\varepsilon}}{(e^\varepsilon-1)^3}$, $d_2 = -\frac{e^{2\varepsilon}}{(e^\varepsilon-1)^3}$, $d_1 = \frac{e^\varepsilon}{(e^\varepsilon-1)^3}$, and $d_0 = -\frac{1}{(e^\varepsilon-1)^3}$. Then, by (C.1), we have:

$$
\begin{aligned}
\mathbb{V}[\hat{f}^\triangle(G)] &= \mathbb{V}[d_3\hat{m}_3 + d_2\hat{m}_2 + d_1\hat{m}_1 + d_0\hat{m}_0] \\
&= d_3^2\mathbb{V}[\hat{m}_3] + d_2^2\mathbb{V}[\hat{m}_2] + d_1^2\mathbb{V}[\hat{m}_1] + d_0^2\mathbb{V}[\hat{m}_0] \\
&\quad + \sum_{i \neq j} d_i d_j \operatorname{Cov}(\hat{m}_i, \hat{m}_j).
\end{aligned} \tag{C.12}
$$

By the Cauchy-Schwarz inequality,

$$
\begin{aligned}
|\operatorname{Cov}(\hat{m}_i, \hat{m}_j)| &\leq \sqrt{\mathbb{V}[\hat{m}_i]\mathbb{V}[\hat{m}_j]} \\
&\leq \max\{\mathbb{V}[\hat{m}_i], \mathbb{V}[\hat{m}_j]\} \\
&\leq \mathbb{V}[\hat{m}_i] + \mathbb{V}[\hat{m}_j].
\end{aligned} \tag{C.13}
$$

Therefore, we have:

$$
\mathbb{V}[\hat{f}^\triangle(G)] = O(\mathbb{V}[\hat{m}_3] + \mathbb{V}[\hat{m}_2] + \mathbb{V}[\hat{m}_1] + \mathbb{V}[\hat{m}_0]). \tag{C.14}
$$

Below, we upper bound $\mathbb{V}[\hat{f}^\triangle(G)]$ in (C.14) by bounding $\mathbb{V}[m_3^*], \ldots, \mathbb{V}[m_0^*]$ and then $\mathbb{V}[\hat{m}_3], \ldots, \mathbb{V}[\hat{m}_0]$. Let $T_{i,j,k} \in \{0,1\}$ be a random variable that takes 1 if and only if v_i, v_j, and v_k form a triangle in the noisy graph G^*. Then we have:

$$
\begin{aligned}
\mathbb{V}[m_3^*] &= \mathbb{V}\left[\sum_{i,j,k} T_{i,j,k}\right] \\
&= \sum_{i<j<k} \sum_{i'<j'<k'} \operatorname{Cov}[T_{i,j,k}, T_{i',j',k'}].
\end{aligned}
$$

If v_i, v_j, v_k and $v_{i'}, v_{j'}, v_{k'}$ intersect in zero or one node, then $T_{i,j,k}$ and $T_{i',j',k'}$ are independent and their covariance is 0. There are only $O(n^4)$ choices of v_i, v_j, v_k and $v_{i'}, v_{j'}, v_{k'}$ that intersect in two or more nodes, as there can only be 4 distinct nodes. Therefore, we have $\mathbb{V}[m_3^*] = O(n^4)$. Similarly, we can prove $\mathbb{V}[m_2^*] = \mathbb{V}[m_1^*] = \mathbb{V}[m_0^*] = O(n^4)$ by regarding $T_{i,j,k}$ as a random variable that takes 1 if and only if v_i, v_j, and v_k form a 2-edge, 1-edge, and no-edges, respectively. In summary, we have:

$$\mathbb{V}[m_3^*] = \mathbb{V}[m_2^*] = \mathbb{V}[m_1^*] = \mathbb{V}[m_0^*] = O(n^4). \tag{C.15}$$

By (C.15) and $\mu = \frac{e^\varepsilon}{e^\varepsilon+1} p_0$, we can upper bound the variance of $\hat{m}_3$ in (C.2) as follows:

$$\mathbb{V}[\hat{m}_3] = \frac{\mathbb{V}[m_3^*]}{p_0^6} = O\left(\frac{n^4}{\mu^6}\right).$$

As with (C.12), (C.13), and (C.14), we can upper bound the variance of $\hat{m}_2$ in (C.3) by using the Cauchy-Schwarz inequality as follows:

$$\begin{aligned}
&\mathbb{V}[\hat{m}_2] \\
&= \mathbb{V}\left[\tfrac{m_2^*}{p_0^2} - \tfrac{3(1-p_0)}{p_0^3} m_3^*\right] \\
&= \frac{\mathbb{V}[m_2^*]}{p_0^4} + \frac{9(1-p_0)^2 \mathbb{V}[m_3^*]}{p_0^6} - \frac{6(1-p_0)\,\mathrm{Cov}(m_2^*, m_3^*)}{p_0^5} \\
&\leq \frac{\mathbb{V}[m_2^*]}{p_0^4} + \frac{9(1-p_0)^2 \mathbb{V}[m_3^*]}{p_0^6} + \frac{6(1-p_0)(\mathbb{V}[m_2^*] + \mathbb{V}[m_3^*])}{p_0^5} \\
&= O\left(\frac{n^4}{\mu^6}\right)
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
&\mathbb{V}[\hat{m}_1] \\
&= \mathbb{V}\left[\tfrac{m_1^*}{p_0} + \tfrac{3(1-p_0)^2}{p_0^3} m_3^* - \tfrac{2(1-p_0)}{p_0^2} m_2^*\right]
\end{aligned}$$

$$= O\left(\frac{n^4}{\mu^6}\right)$$

$$\mathbb{V}[\hat{m}_0]$$

$$= \mathbb{V}[\hat{m}_3 + \hat{m}_2 + \hat{m}_1]$$

$$= \mathbb{V}\left[\tfrac{m_3^*}{p_0^3} + \tfrac{m_2^*}{p_0^2} - \tfrac{3(1-p_0)}{p_0^3} m_3^* + \tfrac{m_1^*}{p_0} + \tfrac{3(1-p_0)^2}{p_0^3} m_3^* - \tfrac{2(1-p_0)}{p_0^2} m_2^*\right]$$

$$= O\left(\frac{n^4}{\mu^6}\right).$$

In summary,

$$\mathbb{V}[\hat{m}_3] = \mathbb{V}[\hat{m}_2] = \mathbb{V}[\hat{m}_1] = \mathbb{V}[\hat{m}_0] = O\left(\frac{n^4}{\mu^6}\right). \tag{C.16}$$

By (C.14) and (C.16), $\mathbb{V}[\hat{f}^{\triangle}(G)] = O\left(\frac{n^4}{\mu^6}\right)$. □

C.8 Proofs of Statements in Section 5

For these proofs, we will write $f_{i,\sigma}(G)$ as a shorthand for $f_{\sigma(i),\sigma(i+1)}$ and $\hat{f}_{i,\sigma}(G)$ as a shorthand for $\hat{f}_{\sigma(i),\sigma(i+1)}$.

C.8.1 Proof of Theorem 13

From (3.4), the quantity $\hat{f}^{\triangle}_{i,j}(G)$ can be written as follows:

$$\hat{f}^{\triangle}_{i,j}(G) = \frac{1}{2}(\hat{f}^{(1)}_{i,j}(G) + \hat{f}^{(2)}_{i,j}(G)), \tag{C.17}$$

where

$$\hat{f}^{(1)}_{i,j}(G) = \frac{(z_{i,j} - q)\sum_{k \in I_{-(i,j)}}(y_k - q_L)}{(1-2q)(1-2q_L)} \tag{C.18}$$

$$\hat{f}^{(2)}_{i,j}(G)) = \frac{(z_{j,i} - q)\sum_{k \in I_{-(i,j)}}(y_k - q_L)}{(1-2q)(1-2q_L)}, \tag{C.19}$$

and each variable y_k represents the output of the RR for the existence of wedge w_{i-k-j} (see Algorithm 6). We call $\hat{f}^{(1)}_{i,j}(G)$ and $\hat{f}^{(2)}_{i,j}(G)$ the first and second estimates, respectively.

First Estimate. Since variables $z_{i,j}$ and y_k in (C.18) are independent, we have

$$\mathbb{E}[\hat{f}^{(1)}_{i,j}(G)] = \frac{\mathbb{E}[z_{i,j}-q]\sum_{k\in I_{-(i,j)}}\mathbb{E}[y_k-q_L]}{(1-2q)(1-2q_L)}.$$

First, suppose $(v_i,v_j)\notin E$. Then, $z_{i,j}=1$ with probability q, and $\mathbb{E}[z_{i,j}-q]=0$. This means $\mathbb{E}[\hat{f}^{(1)}_{i,j}(G)] = f^{\triangle}_{i,j}(G) = 0$. Second, suppose $(v_i,v_j)\in E$. We have $z_{i,j}=1$ with probability $1-q$. We have $\mathbb{E}[z_{i,j}-q]=1-2q$. For any $k\in I_{-(i,j)}$, if $w_{i-k-j}=0$, then we have $\mathbb{E}[y_k-q_L]=0$. If $w_{i-k-j}=1$, then $\mathbb{E}[y_k-q_L]=1-2q_L$. Written concisely, we can say $\mathbb{E}[y_k-q_L]=(1-2q_L)w_{i-k-j}$. Putting this together, we have

$$\begin{aligned}
\mathbb{E}[\hat{f}^{(1)}_{i,j}(G)] &= \frac{(1-2q)\sum_{k\in I_{-(i,j)}}(1-2q_L)w_{i-k-j}}{(1-2q)(1-2q_L)}\\
&= \sum_{k\in I_{-(i,j)}} w_{i-k-j}\\
&= f^{\triangle}_{i,j}(G).
\end{aligned}$$

Thus, $\mathbb{E}[\hat{f}^{(1)}_{i,j}(G)] = f^{\triangle}_{i,j}(G)$ holds for both cases.

Second and Average Estimates. Similarly, we can prove that $\mathbb{E}[\hat{f}^{(2)}_{i,j}(G)] = f^{\triangle}_{i,j}(G)$ holds. Then, by (C.17), $\mathbb{E}[\hat{f}^{\triangle}_{i,j}(G)] = f^{\triangle}_{i,j}(G)$ holds. □

C.8.2 Proof of Theorem 14

Recall that $\hat{f}^{\triangle}_{i,j}(G)$ is an average of the first estimate $\hat{f}^{(1)}_{i,j}(G)$ and second estimate $\hat{f}^{(2)}_{i,j}(G)$; see (C.17), (C.18), and (C.19). We bound the variance of the first estimate.

First Estimate. Let $H=(z_{i,j}-q)\sum_{k\in I_{-(i,j)}}(y_k-q_L)$. Using the law of total variance, we have

$$\mathbb{V}[\hat{f}^{(1)}_{i,j}(G)]$$

$$= \frac{1}{(1-2q)^2(1-2q_L)^2}\left(\mathbb{E}_z[\mathbb{V}_y[H|z_{i,j}]] + \mathbb{V}_z[\mathbb{E}_y[H|z_{i,j}]]\right),$$

where $\mathbb{E}_z$ (resp. $\mathbb{E}_y$) represents the expectation over $z_{i,j}$ (resp. y_k). $\mathbb{V}_z$ (resp. $\mathbb{V}_y$) represents the variance over $z_{i,j}$ (resp. y_k).

We upper bound $\mathbb{V}_z[\mathbb{E}_y[H|z_{i,j}]]$ first. Let $E_y = \mathbb{E}_y[\sum_{k\in I_{-(i,j)}} y_k - q_L]$. When $z_{i,j} = 0$, we have

$$\mathbb{E}_y[H] = -q\mathbb{E}_y[\textstyle\sum_{k\in I_{-(i,j)}} y_k - q_L] = -qE_y.$$

When $z_{i,j} = 1$, we have

$$\mathbb{E}_y[H] = (1-q)\mathbb{E}_y[\textstyle\sum_{k\in I_{-(i,j)}} y_k - q_L] = (1-q)E_y.$$

The difference between these two quantities is E_y. Since $z_{i,j}$ is a Bernoulli random variable with bias q on either 0 or 1, we have $\mathbb{V}_z[\mathbb{E}_y[H|z_{i,j}]] = E_y^2 q(1-q)$. Recalling that $\mathbb{E}_y[y_k - q_L] = (1-2q_L)w_{i-k-j}$ (see the proof of Theorem 13), by linearity of expectation we have that

$$\begin{aligned} E_y &= \sum_{k\in I_{-(i,j)}} (1-2q_L)w_{i-k-j} \\ &\leq (1-2q_L)d_{max}. \end{aligned}$$

Thus,

$$\mathbb{V}_z[\mathbb{E}_y[H|z_{i,j}]] \leq q(1-2q_L)^2 d_{max}^2.$$

Now, we upper bound $\mathbb{E}_z[\mathbb{V}_y[H|z_{i,j}]]$. When $z_{i,j} = 0$, we have $H = -q\sum_{k\in I_{-(i,j)}}(y_k - q)$, which is a sum of $n-2$ Bernoulli random variables. Thus, $\mathbb{V}_y[H|z_{i,j}] = q^2(n-2)q_L(1-q_L) \leq q^2 q_L n$. When $z_{i,j} = 1$, we have by a similar argument that $\mathbb{V}_y[H|z_{i,j}] \leq (1-q)^2 q_L n$. Regardless of the value of $z_{i,j}$, both values attainable by $\mathbb{V}_y[H|z_{i,j}]$ are at most nq_L. Thus,

$$\mathbb{E}_z[\mathbb{V}_y[H|z_{i,j}]] \leq nq_L.$$

Putting all this together, we have the following upper-bound:

$$\mathbb{V}[\hat{f}_{i,j}^{(1)}(G)] \leq \frac{nq_L + q(1-2q_L)^2 d_{max}^2}{(1-2q)^2(1-2q_L)^2}.$$

Second and Average Estimates. Similarly, we can prove the same upper-bound for the second estimate $\hat{f}_{i,j}^{(2)}(G)$. Using (C.17) and Lemma 4 at the end of Appendix C.8.2, we have $\mathbb{V}[\hat{f}_{i,j}^{\triangle}(G)] \leq \mathbb{V}[\hat{f}_{i,j}^{(1)}(G)]$, and the result follows.

Effect of Shuffling. When ε and δ are constants and $\varepsilon_L \geq \log n + O(1)$, we have $nq_L = \frac{n}{e^{\varepsilon_L+1}} = O(1)$ and $q_L = O(1/n)$, and the bound becomes

$$\mathbb{V}[\hat{f}_{i,j}^{\triangle}(G)] \leq O(d_{max}^2).$$

$\square$

Lemma 4. *Let X, Y be two real-valued random variables. Then $\mathbb{V}[X+Y] \leq 4\max\{\mathbb{V}[X], \mathbb{V}[Y]\}$.*

Proof. We have $\mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y] + 2Cov(X,Y)$. By the Cauchy-Schwarz inequality, we have $Cov(X,Y) \leq \sqrt{\mathbb{V}[X]\mathbb{V}[Y]} \leq \max\{\mathbb{V}[X], \mathbb{V}[Y]\}$. Our result follows by observing $\mathbb{V}[X] + \mathbb{V}[Y] \leq 2\max\{\mathbb{V}[X], \mathbb{V}[Y]\}$. $\square$

C.8.3 Proof of Theorem 15

First, we show that WSLE meets the desired privacy requirements. Let $x_k = w_{i-k-j}$. In step 1 of WSLE, for each $k \in I_{-(i,j)}$, user v_k sends $y_k = \mathscr{R}_{\varepsilon_L}^W(x_k)$ to the shuffler. Then, the shuffler sends $\{y_{\pi(k)} | k \in I_{-(i,j)}\}$ to the data collector. Thus, by Theorem 12, $\{x_k | k \in I_{-(i,j)}\}$ is protected with (ε, δ)-DP, where $\varepsilon = f(n-2, \varepsilon_L, \delta)$. Note that changing $a_{k,i}$ will change x_k if and only if $a_{k,j} = 1$. Thus, for any $k \in I_{-(i,j)}$, $a_{k,i}$ and $a_{k,j}$ are protected with (ε, δ)-DP.

In step 1, user v_i (resp. v_j) sends $z_{i,j} = \mathscr{R}^W_\varepsilon(a_{i,j})$ (resp. $z_{j,i} = \mathscr{R}^W_\varepsilon(a_{j,i})$) to the data collector. Since $\mathscr{R}^W_\varepsilon$ provides ε-DP, $a_{i,j}$ and $a_{j,i}$ are protected with ε-DP.

Putting all this together, each element of the i-th and j-th columns in the adjacency matrix $\mathbf{A}$ is protected with (ε, δ)-DP. Thus, by Proposition 6, WSLE provides (ε, δ)-element-level DP and $(2\varepsilon, 2\delta)$-edge DP.

$\mathsf{WShuffle}_\triangle$ interacts with $\mathbf{A}$ by calling WSLE on

$$(v_{\sigma(1)}, v_{\sigma(2)}), \ldots, (v_{\sigma(2t-1)}, v_{\sigma(2t)}).$$

Each of these calls use disjoint elements of $\mathbf{A}$, and thus each element of $\mathbf{A}$ is still protected by (ε, δ)-level DP and by $(2\varepsilon, 2\delta)$-edge DP. □

C.8.4 Proof of Theorem 16

Notice that the number of triangles in a graph can be computed as

$$\begin{aligned} 6f^\triangle(G) &= \sum_{1 \le i,j \le n, i \ne j} f^\triangle_{i,j}(G) \\ &= n(n-1)\mathbb{E}_\sigma[f^\triangle_{i,\sigma}(G)], \end{aligned} \tag{C.20}$$

where $i \in [n]$ is arbitrary and σ is a random permutation on $[n]$. The constant 6 appears because each triangle appears six times when summing up $f^\triangle_{i,j}(G)$; e.g., a triangle (v_1, v_2, v_3) appears in $f^\triangle_{1,2}(G)$, $f^\triangle_{2,1}(G)$, $f^\triangle_{1,3}(G)$, $f^\triangle_{3,1}(G)$, $f^\triangle_{2,3}(G)$, and $f^\triangle_{3,2}(G)$.

Note that there are two kinds of randomness in $\hat{f}^\triangle(G)$: randomness in choosing a permutation σ and randomness in Warner's RR. By (3.7), the expectation can be written as follows:

$$\mathbb{E}_{\sigma,RR}[\hat{f}^{\triangle}(G)] = \frac{n(n-1)}{6t} \sum_{i=1,3,\ldots}^{2t-1} \mathbb{E}_{\sigma,RR}[\hat{f}^{\triangle}_{i,\sigma}(G)].$$

By the law of total expectation, we have

$$\begin{aligned}
\mathbb{E}_{\sigma,RR}[\hat{f}^{\triangle}_{i,\sigma}(G)] &= \mathbb{E}_{\sigma}[\mathbb{E}_{RR}[\hat{f}^{\triangle}_{i,\sigma}(G)|\sigma]] \\
&= \mathbb{E}_{\sigma}[f^{\triangle}_{i,\sigma}(G)] \;\; \text{(by Theorem 13)} \\
&= \frac{6}{n(n-1)} f^{\triangle}(G) \;\; \text{(by (C.20))}.
\end{aligned}$$

Putting all together,

$$\begin{aligned}
\mathbb{E}_{\sigma,RR}[\hat{f}^{\triangle}(G)] &= \frac{n(n-1)}{6t} \sum_{i=1,3,\ldots}^{2t-1} \mathbb{E}_{\sigma,RR}[\hat{f}^{\triangle}_{i,\sigma}(G)] \\
&= \frac{n(n-1)}{6t} \sum_{i=1,3,\ldots}^{2t-1} \frac{6}{n(n-1)} f^{\triangle}(G) \\
&= f^{\triangle}(G).
\end{aligned}$$

□

C.8.5 Proof of Theorem 17

Because $\hat{f}^{\triangle}(G)$ is unbiased, $\mathsf{MSE}(\hat{f}^{\triangle}(G)) = \mathbb{V}[\hat{f}^{\triangle}(G)]$. By (3.10), the variance can be written as follows:

$$\begin{aligned}
&\mathbb{V}_{\sigma,RR}[\hat{f}^{\triangle}(G)] \\
&= \frac{n^2(n-1)^2}{36t^2} \mathbb{V}_{\sigma,RR}\left[\sum_{i=1,3,\ldots}^{2t-1} \hat{f}^{\triangle}_{i,\sigma}(G)\right] \\
&= \frac{n^2(n-1)^2}{36t^2} \mathbb{V}_{\sigma}\left[\sum_{i=1,3,\ldots}^{2t-1} \mathbb{E}_{RR}\left[\hat{f}^{\triangle}_{i,\sigma}(G)\middle|\sigma\right]\right] \qquad \text{(C.21)}
\end{aligned}$$

$$+\frac{n^2(n-1)^2}{36t^2}\mathbb{E}_\sigma\left[\sum_{i=1,3,\ldots}^{2t-1}\mathbb{V}_{RR}\left[\hat{f}^{\triangle}_{i,\sigma}(G)\middle|\sigma\right]\right], \tag{C.22}$$

where in the step we used the law of total variance and the independence of each $\hat{f}^{\triangle}_{i,\sigma}(G)$ given a fixed σ.

Bounding (C.21)**:** We can write

$$\mathbb{V}_\sigma\left[\sum_{i=1,3,\ldots}^{2t-1}\mathbb{E}_{RR}\left[\hat{f}^{\triangle}_{i,\sigma}(G)\middle|\sigma\right]\right]=\mathbb{V}_\sigma\left[\sum_{i=1,3,\ldots}^{2t-1}f^{\triangle}_{i,\sigma}(G)\right]. \tag{C.23}$$

Each random variable $f^{\triangle}_{i,\sigma}(G)$ represents a uniform draw from the set $\{f^{\triangle}_{i,j}(G) : i,j\in[n], i\neq j\}$ without replacement. Applying Lemma 5, the variance in (C.23) is upper bounded by $t\mathbb{V}_\sigma[f_{1,\sigma}(G)]$. We can upper bound this final term in the following:

$$\begin{aligned}
&\mathbb{V}_\sigma\left[f^{\triangle}_{1,\sigma}(G),\right]\\
&\leq \mathbb{E}_\sigma[(f^{\triangle}_{1,\sigma}(G))^2]\\
&=\frac{1}{n(n-1)}\sum_{1\leq i,j\leq n, i\neq j}(f^{\triangle}_{i,j}(G))^2\\
&=\frac{1}{n(n-1)}\sum_{(i,j)\in E}(f^{\triangle}_{i,j}(G))^2 \ \text{ (because } f^{\triangle}_{i,j}(G)=0 \text{ for } (i,j)\notin E)\\
&\leq \frac{d^3_{max}}{n-1} \ \text{ (because } |E|\leq nd_{max} \text{ and } f^{\triangle}_{i,j}(G)\leq d_{max}).
\end{aligned} \tag{C.24}$$

Plugging this into (C.23), we obtain

$$\mathbb{V}_\sigma\left[\sum_{i=1,3,\ldots}^{2t-1}\mathbb{E}_{RR}\left[\hat{f}^{\triangle}_{i,\sigma}(G)\middle|\sigma\right]\right]\leq\frac{td^3_{max}}{n-1}.$$

Bounding (C.22)**:** For any value of i and permutation σ, we have from Theorem 14 that

$$\mathbb{V}_{RR}[\hat{f}^{\triangle}_{i,\sigma}(G)|\sigma]\leq err_{\mathsf{WSLE}}(n,d_{max},q,q_L).$$

Thus,

$$\mathbb{E}_\sigma\left[\sum_{i=1,3,\ldots}^{2t-1}\mathbb{V}_{RR}\left[\hat{f}_{i,\sigma}^{\triangle}(G)\middle|\sigma\right]\right]\leq t\cdot err_{\mathsf{WSLE}}(n,d_{max},q,q_L).$$

Putting it together: Plugging the upper bounds in, we obtain a final bound of

$$\mathbb{V}[\hat{f}^{\triangle}(G)]\leq\frac{n^4}{36t}err_{\mathsf{WSLE}}(n,d_{max},q,q_L)+\frac{n^3}{36t}d_{max}^3.$$

Finally, when ε and δ are constants, $\varepsilon_L=\log(n)+O(1)$, and $t=\lfloor\frac{n}{2}\rfloor$,

$$err_{\mathsf{WSLE}}(n,d_{max},q,q_L)=O(d_{max}^2),$$

and we obtain

$$\mathbb{V}[\hat{f}^{\triangle}(G)]=O(n^3d_{max}^2+n^2d_{max}^3).$$

We can verify that $n^3d_{max}^2\geq n^2d_{max}^3$ for all values of d_{max} between 1 and n, and thus the bound simplifies to $O(n^3d_{max}^2)$. $\square$

Lemma 5. *Let $\mathscr{X}$ be a finite subset of real numbers of size n. Suppose $X_1,X_2,\ldots,X_k$ for $k\leq n$ are sampled uniformly from $\mathscr{X}$ without replacement. Then,*

$$\mathbb{V}[X_1+\cdots+X_k]\leq k\mathbb{V}[X_1].$$

Proof. We have

$$\mathbb{V}[X_1+\cdots+X_k]=\sum_{i,j=1}^{k}Cov(X_i,X_j).$$

We are done by observing each X_i has the same distribution, and so $\mathbb{V}[X_i]=\mathbb{V}[X_1]$, and by showing $Cov(X_i,X_j)\leq 0$ when $i\neq j$. To prove the latter statement, let $\mathscr{X}=\{x_1,\ldots,x_n\}$ with $x_1\leq x_2\leq\cdots\leq x_n$. Notice that for any $i\neq j$, the distribution $X_i|X_j$ is uniformly distributed on $\mathscr{X}\setminus\{X_j\}$. This implies that $\mathbb{E}[X_i|X_j=x_1]\geq\mathbb{E}[X_i|X_j=x_2]\geq\cdots\geq\mathbb{E}[X_i|X_j=x_n]$. Let $y_\ell=\mathbb{E}[X_i|X_j=x_\ell]$ for $i,j\in[n]$. We have $y_1\geq y_2\geq\cdots\geq y_n$ for any $i\in[n]$.

Because X_j is uniformly distributed across $\mathscr{X}$, we have $\mathbb{E}[X_j] = \frac{1}{n}\sum_{\ell=1}^{n} x_i$. Next, we can observe $\mathbb{E}[X_i] = \sum_{\ell=1}^{n}\mathbb{E}[X_i|X_j = x_\ell]\Pr[X_j = x_\ell] = \frac{1}{n}\sum_{\ell=1}^{n} y_\ell$. Finally, we have $\mathbb{E}[X_iX_j] = \sum_{\ell=1}^{n} x_\ell$ $\mathbb{E}[X_i|X_j = x_\ell]\Pr[X_j = x_\ell] = \frac{1}{n}\sum_{\ell=1}^{n} x_\ell y_\ell$. Using Chebyshev's sum inequality, we are able to deduce that $\mathbb{E}[X_iX_j] \leq \mathbb{E}[X_i]\mathbb{E}[X_j]$, implying $Cov(X_i, X_j) \leq 0$. □

C.8.6 Proof of Theorem 18

$\mathsf{WShuffle}_\triangle^*$ interacts with $\mathbf{A}$ in the same way as $\mathsf{WShuffle}_\triangle$, plus the additional degree estimates $\tilde{d}_i$. The first calculation is protected by (ε_2, δ)-element DP by Theorem 15. The noisy degrees are calculated with the Laplace mechanism, which provides a protection of $(\varepsilon_1, 0)$-element DP. Using composition, the entire computation provides $(\varepsilon_1 + \varepsilon_2, \delta)$-element DP, and by Proposition 6, the computation also provides $(2(\varepsilon_1 + \varepsilon_2), 2\delta)$-edge DP. □

C.8.7 Proof of Theorem 19

Let $f_*^\triangle$ be the estimator returned by $\mathsf{WShuffle}_\triangle^*$ to distinguish it from that returned by $\mathsf{WShuffle}_\triangle$. Define $V^+ = \{i : i \in [n], \tilde{d}_i \geq c\tilde{d}_{avg}\}$, and let $V^- = [n] \setminus V^+$. The randomness in $\mathsf{WShuffle}_\triangle^*$ comes from randomized response, from the choice of σ, and from the choice of D. Note that $i \in D$ if and only if $\mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+}$. For any V^+ and V^-,

$$
\begin{aligned}
&\mathbb{E}[\hat{f}_*^\triangle(G)|V^+, V^-] \\
&= \mathbb{E}_{\sigma,RR}\left[\frac{n(n-1)}{6t}\sum_{i\in D}\hat{f}_{i,\sigma}^\triangle(G)\middle| V^+, V^-\right] \\
&= \mathbb{E}_{\sigma,RR}\left[\frac{n(n-1)}{6t}\sum_{i=1,3,\ldots}^{2t-1}\mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+}\hat{f}_{i,\sigma}^\triangle(G)\middle| V^+, V^-\right] \\
&= \frac{n(n-1)}{6t}\sum_{i=1,3,\ldots}^{2t-1}\mathbb{E}_{\sigma,RR}[\mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+}\hat{f}_{i,\sigma}^\triangle(G)|V^+, V^-] \\
&= \frac{n(n-1)}{6t}\sum_{i=1,3,\ldots}^{2t-1}\mathbb{E}_{\sigma}[\mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+}f_{i,\sigma}^\triangle(G)|V^+, V^-],
\end{aligned}
\tag{C.25}
$$

where the last line uses the fact that for a fixed i,σ, $\mathbb{E}_{RR}[\hat{f}_{i,\sigma}(G)] = f_{i,\sigma}(G)$ (Theorem 13). Using the inequality $f^{\triangle}_{i,\sigma}(G) \leq \min\{d_{\sigma(i)}, d_{\sigma(i+1)}\}$, and the fact that $f^{\triangle}_{i,\sigma}(G) = 0$ unless $(v_{\sigma(i)}, v_{\sigma(i+1)}) \in E$, we obtain

$$
\begin{aligned}
&|\mathbb{E}_{\sigma}[f^{\triangle}_{i,\sigma}(G) - \mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+} f^{\triangle}_{i,\sigma}(G)|V^+,V^-]| \\
&\leq \mathbb{E}_{\sigma}[\min\{d_{\sigma(i)}, d_{\sigma(i+1)}\}\mathbf{1}_{v_{\sigma(i)},v_{\sigma(i+1)}\in E}\mathbf{1}_{\sigma(i)\in V^-\vee\sigma(i+1)\in V^-} \\
&\qquad |V^+,V^-].
\end{aligned}
$$

Expanding the second equation, we obtain

$$
\begin{aligned}
&\frac{1}{n(n-1)} \sum_{1\leq i,j\leq n, i\neq j} \min\{d_i,d_j\}\mathbf{1}_{(i,j)\in E}\mathbf{1}_{i\in V^-\vee j\in V^-} \\
&= \frac{1}{n(n-1)}\left(\sum_{j\in V^-,(i,j)\in E}\min\{d_i,d_j\}\right. \\
&\qquad\left. + \sum_{i\in V^-, j\in V^+,(i,j)\in E}\min\{d_i,d_j\}\right) \\
&\leq \frac{1}{n(n-1)}\left(\sum_{j\in V^-,(i,j)\in E} d_j + \sum_{i\in V^-, j\in V^+,(i,j)\in E} d_i\right) \\
&\leq \frac{1}{n(n-1)}\left(\sum_{j\in V^-} d_j^2 + \sum_{i\in V^-} d_i^2\right) \\
&\leq \frac{2}{n(n-1)} d^{(2)}_{sum,-},
\end{aligned}
$$

where $d^{(2)}_{sum,-} = \sum_{i\in V^-} d_i^2$.

Applying the triangle inequality,

$$
\begin{aligned}
&\left|\sum_{i=1,3,\ldots}^{2t-1}\mathbb{E}_{\sigma}[\mathbf{1}_{\sigma(i)\in V}\mathbf{1}_{\sigma(i+1)\in V} f^{\triangle}_{i,\sigma}(G)|V] - \sum_{i=1,3,\ldots}^{2t-1}\mathbb{E}_{\sigma}[f^{\triangle}_{i,\sigma}(G)]\right| \\
&\leq \frac{2t}{n(n-1)} d^{(2)}_{sum,-}
\end{aligned}
$$

Plugging in (C.25) and (C.20), we obtain

$$\left|\frac{6t}{n(n-1)}\mathbb{E}[\hat{f}_*^{\triangle}(G)|V^+,V^-] - \frac{6t}{n(n-1)}f^{\triangle}(G)\right| \leq \frac{2t}{n(n-1)}d_{sum,-}^{(2)}$$
$$\left|\mathbb{E}[\hat{f}_*^{\triangle}(G)|V^+,V^-] - f^{\triangle}(G)\right| \leq \frac{1}{3}d_{sum,-}^{(2)}. \tag{C.26}$$

Marginalizing over all V^+ and V^-, and applying the triangle inequality again, we obtain

$$\left|\mathbb{E}[\hat{f}_*^{\triangle}(G)] - f^{\triangle}(G)\right| \leq \frac{1}{3}\mathbb{E}[d_{sum,-}^{(2)}].$$

Now, we have

$$\begin{aligned}\mathbb{E}[d_{sum,-}^{(2)}] &= \sum_{i=1}^{n} d_i^2 \Pr[i \in V^-] \\ &\leq n(cd_{avg})^2 + \sum_{i\in[n],d_i\geq cd_{avg}} d_i^2 \Pr[i \in V^-].\end{aligned}$$

In the second line, we split the sum into those nodes where $d_i \geq cd_{avg}$ (of which there are only n^{α}, since $c \geq \lambda$), and other nodes. In order for i to be in V^- for a node such that $d_i \geq cd_{avg}$, we must have that $Lap(\frac{1}{\varepsilon_1}) \leq d_i - cd_{avg}$. The probability of this occuring is at most $e^{-(d_i - cd_{avg})\varepsilon_1}$. Using calculus, we can show the expression $d_i^2 e^{-(d_i-cd_{avg})\varepsilon_1}$ is maximized when $d_i = cd_{avg}$ (when $cd_{avg} \geq \frac{2}{\varepsilon_1}$), and when $d_i = \frac{2}{\varepsilon_1}$ (otherwise).

In the first case, we have

$$\sum_{i\in[n],d_i\geq cd_{avg}} d_i^2 \Pr[i \in V^-] \leq n^{\alpha}(cd_{avg})^2.$$

In the second, we have

$$\sum_{i\in[n],d_i\geq cd_{avg}} d_i^2 \Pr[i \in V^-] \leq n^{\alpha}\left(\frac{2}{\varepsilon_1}\right)^2 e^{-2+cd_{avg}\varepsilon_1} \leq n^{\alpha}\left(\frac{2}{\varepsilon_1}\right)^2.$$

Thus, our overall bound becomes

$$\begin{aligned}\mathbb{E}[d_{sum,-}^{(2)}] &= \sum_{i=1}^{n} d_i^2 \Pr[i \in V^-] \\ &\leq n(cd_{avg})^2 + n^{\alpha}\left(\frac{2}{\varepsilon_1}\right)^2,\end{aligned}$$

and therefore

$$\left|\mathbb{E}[\hat{f}_*^{\triangle}(G)] - f^{\triangle}(G)\right| \leq \frac{nc^2 d_{avg}^2}{3} + \frac{4n^{\alpha}}{3\varepsilon_1^2}.$$

□

C.8.8 Proof of Theorem 20

Define $f_*^{\triangle}$, V^+, and V^- be as they are defined in Appendix C.8.7. Using the law of total variance, we have

$$\mathbb{V}[\hat{f}_*^{\triangle}(G)] = \mathbb{E}_{V^+,V^-}\mathbb{V}[\hat{f}_*^{\triangle}(G)|V^+,V^-] + \mathbb{V}_{V^+,V^-}\mathbb{E}[\hat{f}_*^{\triangle}(G)|V^+,V^-]. \tag{C.27}$$

From (C.26), $\mathbb{E}[\hat{f}_*^{\triangle}(G)|V^+,V^-]$ is always in the range $[f^{\triangle}(G) - \frac{\overline{d}}{3}, f^{\triangle}(G) + \frac{\overline{d}}{3}]$, where $\overline{d} = \max_{V^- \subseteq [n]} \sum_{i \in V^-} d_i^2 \leq \sum_{i=1}^{n} d_i^2 \leq nd_{max}^2$. Because the maximum variance of a variable bounded between $[A,B]$ is $\frac{(B-A)^2}{4}$, the second term of (C.27) can be written as

$$\mathbb{V}_{V^+,V^-}\mathbb{E}[\hat{f}_*^{\triangle}(G)|V^+,V^-] \leq \frac{(\sum_{i=1}^{n} d_i^2)^2}{9} \leq \frac{n^2 d_{max}^4}{9}.$$

Again using the law of total variance on $\mathbb{V}[\hat{f}_*^{\triangle}(G)|V^+,V^-]$, we obtain, similar to (C.21) and (C.22):

$$\begin{aligned}&\left(\frac{6t}{n(n-1)}\right)^2 \mathbb{V}[\hat{f}_*^{\triangle}(G)|V^+,V^-] \\ &= \mathbb{V}_{\sigma,RR}\left[\sum_{i=1,3,\dots}^{2t-1} \mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+}\hat{f}_{i,\sigma}^{\triangle}(G) \middle| V^+,V^-\right]\end{aligned}$$

$$= \mathbb{V}_\sigma \left[\sum_{i=1,3,\dots}^{2t-1} \mathbf{1}_{\sigma(i)\in V^+} \mathbf{1}_{\sigma(i+1)\in V^+} \mathbb{E}_{RR} \left[\hat{f}^{\triangle}_{i,\sigma}(G) \middle| \sigma \right] \middle| V^+, V^- \right] \tag{C.28}$$

$$+ \mathbb{E}_\sigma \left[\sum_{i=1,3,\dots}^{2t-1} \mathbf{1}_{\sigma(i)\in V^+} \mathbf{1}_{\sigma(i+1)\in V^+} \mathbb{V}_{RR} \left[\hat{f}^{\triangle}_{i,\sigma}(G) \middle| \sigma \right] \middle| V^+, V^- \right] \tag{C.29}$$

Bounding (C.28)**:** Similar to the process for bounding (C.21), we have that $\mathbb{E}_{RR}[\hat{f}^{\triangle}_{i,\sigma}(G)|\sigma] = f^{\triangle}_{i,\sigma}(G)$. The collection $\{\mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+} f^{\triangle}_{i,\sigma}(G)\}$ for $i \in \{1,3,\dots,2t-1\}$ consists of draws without replacement from the set

$$\mathscr{T} = \{\mathbf{1}_{i\in V^+}\mathbf{1}_{j\in V^+} f^{\triangle}_{i,j}(G) : i,j \in [n], i \neq j\},$$

We can then apply Lemma 5 for an upper bound of

$$t\mathbb{V}_\sigma[\mathbf{1}_{\sigma(1)\in V^+}\mathbf{1}_{\sigma(2)\in V^+} f_{1,\sigma}(G)].$$

Using the same line of reasoning as that to obtain (C.24), we obtain an upper bound of

$$\begin{aligned}
&\mathbb{V}_\sigma[\mathbf{1}_{\sigma(1)\in V^+}\mathbf{1}_{\sigma(2)\in V^+} f^{\triangle}_{1,\sigma}(G)] \\
&\leq \mathbb{E}_\sigma[\mathbf{1}_{\sigma(1)\in V^+}\mathbf{1}_{\sigma(2)\in V^+} f^{\triangle}_{1,\sigma}(G)^2] \\
&\leq \frac{1}{n(n-1)} \sum_{(v_i,v_j)\in E, i\in V^+, j\in V^+} f^{\triangle}_{i,j}(G)^2 \\
&\leq \frac{1}{n(n-1)} |V^+| d^3_{max},
\end{aligned}$$

with the last line holding because there are at most $|V^+|d_{max}$ edges within V^+. Thus, (C.28) is at most $\frac{t|V^+|d^3_{max}}{n(n-1)}$.

Bounding (C.29)**:** Similar to the process for bounding (C.22), we use Theorem 14, along with the fact that $\mathbb{E}_\sigma[\mathbf{1}_{\sigma(i)\in V^+}\mathbf{1}_{\sigma(i+1)\in V^+}] = \frac{|V^+|(|V^+|-1)}{n(n-1)} \leq \frac{|V^+|^2}{n^2}$, to obtain an upper bound of

$$\frac{t|V^+|^2}{n^2} err_{\mathsf{WSLE}}(n, d_{max}, q, q_L).$$

Putting it together: Summing together (C.28) and (C.29), manipulating constants, and taking an expectation over V, we obtain

$$\begin{aligned}
&\mathbb{E}_{V^+,V^-}\mathbb{V}[\hat{f}_*^{\triangle}(G)|V^+,V^-] \\
&\leq \frac{n^2 d_{max}^3 \mathbb{E}[|V^+|]}{36t} + \frac{n^2\mathbb{E}[|V^+|^2]}{36t} err_{\mathsf{WSLE}}(n, d_{max}, q, q_L).
\end{aligned}$$

Plugging back into (C.27), we have

$$\begin{aligned}
&\mathbb{V}[\hat{f}_*^{\triangle}(G)] \leq \frac{n^2 d_{max}^4}{9} + \\
&\frac{n^2\mathbb{E}[|V^+|^2]}{36t} err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) + \frac{n^2 d_{max}^3 \mathbb{E}[|V^+|]}{36t}.
\end{aligned}$$

We are given that there are just n^α nodes with degrees higher than λd_{avg}. Let L_i be the Laplace random variable added to d_i. We have $\Pr[L_i \geq T] \leq e^{-T\varepsilon_1}$ for $T > 0$. Observe

$$|V^+| \leq n^\alpha + \sum_{i \in [n], d_i \leq \lambda d_{avg}} \mathbf{1}_{L_i \geq (c-\lambda) d_{avg}}.$$

We have $\mathbb{E}[\mathbf{1}_{L_i \geq (c-\lambda) d_{avg}}] = e^{-(c-\lambda)\varepsilon_1 d_{avg}}$, which by the condition on c, is at most $n^{\alpha-1}$. Thus, $\mathbb{E}[|V^+|] \leq n^\alpha + n \cdot n^{\alpha-1} = 2n^\alpha$. Similarly,

$$\begin{aligned}
\mathbb{E}[|V^+|^2] &\leq n^{2\alpha} + 2n^\alpha \mathbb{E}[|V^+|] \\
&\quad + 2\sum_{i,j \in [n], i \neq j, d_i, d_j \leq \lambda d_{avg}} n^{2(\alpha-1)} + \sum_{i \in [n], d_i \leq \lambda d_{avg}} n^{\alpha-1} \\
&\leq n^{2\alpha} + 4n^{2\alpha} + 2n^{2\alpha} + n^\alpha \\
&\leq 8n^{2\alpha}.
\end{aligned}$$

Plugging in, we obtain

$$\mathbb{V}[\hat{f}_*^{\triangle}(G)] \leq \frac{n^2 d_{max}^4}{9} + \\ \frac{2n^{2+2\alpha}}{9t} err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) + \frac{n^{2+\alpha} d_{max}^3}{36t}.$$

When ε and δ, are treated as constants, $\varepsilon_L = \log n + O(1)$, and $t = \lfloor \frac{n}{2} \rfloor$, then $err_{\mathsf{WSLE}}(n, d_{max}, q, q_L) = O(d_{max}^2)$, and

$$\begin{aligned} \mathbb{V}[\hat{f}_*^{\triangle}(G)] &\leq O(n^2 d_{max}^4 + n^{1+2\alpha} d_{max}^2 + n^{1+\alpha} d_{max}^3) \\ &= O(n^2 d_{max}^4 + n^{1+2\alpha} d_{max}^2). \end{aligned}$$

We can remove the third term because it is always smaller than the first term. □

C.9 Proofs of Statements in Section 6

In the following, we define $f_{i,\sigma}^{\wedge}(G) = f_{\sigma(2i),\sigma(2i-1)}^{\wedge}(G)$ and $\hat{f}_{i,\sigma}^{\wedge}(G) = \hat{f}_{\sigma(2i),\sigma(2i-1)}^{\wedge}(G)$ as a shorthand. Similarly, we do the same with $f_{i,\sigma}^{\square}$, $\hat{f}_{i,\sigma}^{\square}$ by replacing $\wedge$ with $\square$.

C.9.1 Proof of Theorem 21

$\mathsf{WShuffle}_{\square}$ interacts with $\mathbf{A}$ in the same way as $\mathsf{WShuffle}_{\triangle}$. The subsequent processes (lines 7-10 in Algorithm 10) are post-processing on $\{y_{\pi_i(k)} | k \in I_{-(\sigma(i),\sigma(i+1))}\}$. Thus, by the post-processing invariance [73], $\mathsf{WShuffle}_{\square}$ provides (ε, δ)-element DP and $(2\varepsilon, 2\delta)$-element DP in the same way as $\mathsf{WShuffle}_{\triangle}$ (see Appendix C.8.3 for the proof of DP for $\mathsf{WShuffle}_{\triangle}$). □

C.9.2 Proof of Theorem 22

Notice that the number of 4-cycles can be computed as

$$4 f^{\square}(G) = \sum_{1 \leq i,j \leq n, i \neq j} f_{i,j}^{\square}(G) \tag{C.30}$$

$$= n(n-1)\mathbb{E}_{\sigma}[f^{\square}_{i,\sigma}(G)], \tag{C.31}$$

where the 4 appears because for every 4-cycle, there are 4 choices for diagonally opposite nodes.

We will show that $\hat{f}^{\wedge}_{i,j}(G) = \sum_{k\in I_{-(i,j)}} \frac{y_k - q_L}{1-2q_L}$ is an unbiased estimate of $f^{\wedge}_{i,j}(G)$. Since we use ε_L-RR in WS, we have

$$\mathbb{E}[y_k] = (1-q_L)w_{i-k-j} + q_L(1-w_{i-k-j}).$$

Thus, we have:

$$\begin{aligned}\mathbb{E}\left[\frac{y_k - q_L}{1-2q_L}\right] &= \frac{(1-q_L)w_{i-k-j} - q_L w_{i-k-j}}{1-2q_L}\\ &= w_{i-k-j}.\end{aligned}$$

The sum of these is clearly the number of wedges connected to users v_i and v_j. Therefore, $\mathbb{E}[\hat{f}^{\wedge}_{i,j}(G)] = f^{\wedge}_{i,j}(G)$.

Furthermore, $(1-2q_L)\hat{f}_{i,j}(G)$ is a sum of $(n-2)$ Bernoulli random variables shifted by $(n-2)q_L$. Each Bernoulli r.v. has variance $q_L(1-q_L)$, and thus $\mathbb{V}[\hat{f}_{i,j}(G)] = \frac{(n-2)q_L(1-q_L)}{(1-2q_L)^2}$. This information is enough to verify that

$$\begin{aligned}\mathbb{E}[\hat{f}^{\wedge}_{i,j}(G)^2] &= \mathbb{E}[\hat{f}^{\wedge}_{i,j}(G)]^2 + \mathbb{V}[\hat{f}^{\wedge}_{i,j}(G)]\\ &= f^{\wedge}_{i,j}(G)^2 + \frac{(n-2)q_L(1-q_L)}{(1-2q_L)^2}.\end{aligned}$$

Putting this together and plugging into (3.15), we obtain

$$\begin{aligned}\mathbb{E}_{RR}\left[\hat{f}^{\square}_{i,j}(G)\right] &= \mathbb{E}_{RR}\left[\frac{\hat{f}^{\wedge}_{i,j}(G)(\hat{f}^{\wedge}_{i,j}(G)-1)}{2} - \frac{n-2}{2}\frac{q_L(1-q_L)}{(1-2q_L)^2}\right]\\ &= \frac{f^{\wedge}_{i,j}(G)(f^{\wedge}_{i,j}(G)-1)}{2},\end{aligned} \tag{C.32}$$

In the above equation, we emphasize that the randomness in the expectation is over the randomized response used by the estimator $\hat{f}$. From (C.32), the estimate $\hat{f}^{\square}(G)$ satisfies:

$$
\begin{aligned}
\mathbb{E}[\hat{f}^{\square}(G)] &= \mathbb{E}_{\sigma}[\mathbb{E}_{RR}[\hat{f}^{\square}(G)|\sigma]] \\
&= \frac{n(n-1)}{4t}\sum_{i=1}^{t}\mathbb{E}_{\sigma}\left[\mathbb{E}_{RR}\left[\frac{\hat{f}_i(G)(\hat{f}_i(G)-1)}{2} - \frac{n-2}{2}\frac{q_L(1-q_L)}{(1-2q_L)^2}\middle|\sigma\right]\right] \\
&= \frac{n(n-1)}{4t}\sum_{i=1}^{t}\mathbb{E}_{\sigma}\left[\frac{\hat{f}_{i,\sigma}(G)(\hat{f}_{i,\sigma}(G)-1)}{2}\right] \\
&= \frac{n(n-1)}{4}\mathbb{E}_{\sigma}\left[f_{i,\sigma}^{\square}(G)\right] \\
&= f^{\square}(G).
\end{aligned}
$$

□

C.9.3 Proof of Theorem 23

From (3.16), we have

$$
\mathbb{V}[\hat{f}^{\square}(G)] = \left(\frac{n(n-1)}{4t}\right)^2 \mathbb{V}\left[\sum_{i=1}^{t}\hat{f}_{i,\sigma}^{\square}(G)\right].
$$

Using the law of total variance, along with the fact that the $\hat{f}_{i,\sigma}^{\square}(G)$ are mutually independent given σ, we write

$$
\begin{aligned}
\mathbb{V}_{\sigma,RR}\left[\sum_{i=1}^{t}\hat{f}_{i,\sigma}^{\square}(G)\right] &= \mathbb{E}_{\sigma}\left[\sum_{i=1}^{t}\mathbb{V}_{RR}\left[\hat{f}_{i,\sigma}^{\square}(G)\middle|\sigma\right]\right] && \text{(C.33)} \\
&\quad + \mathbb{V}_{\sigma}\left[\sum_{i=1}^{t}\mathbb{E}_{RR}\left[\hat{f}_{i,\sigma}^{\square}(G)\middle|\sigma\right]\right]. && \text{(C.34)}
\end{aligned}
$$

We will now shift our attention to upper bounding the terms on the left- and right-hand sides of the above sum.

Upper bounding (C.33)**:** Our analysis will apply to any fixed σ, and we will assume for the rest of the proof that σ is a fixed constant permutation. By (3.15), we have

$$\mathbb{V}_{RR}[\hat{f}^{\square}_{i,\sigma}(G)|\sigma] = \mathbb{V}_{RR}\left[\frac{\hat{f}^{\wedge}_{i,\sigma}(G)(\hat{f}^{\wedge}_{i,\sigma}(G)-1)}{2}\middle|\sigma\right].$$

Each term can be upper bounded using the fact that $\mathbb{V}[X+Z] \leq 4\max\{\mathbb{V}[X],\mathbb{V}[Z]\}$ for random variables X,Z. Thus, for any i, we have $\mathbb{V}[\frac{\hat{f}^{\wedge}_{i,\sigma}(G)(\hat{f}^{\wedge}_{i,\sigma}(G)-1)}{2}] \leq \max\{\mathbb{V}[\hat{f}^{\wedge}_{i,\sigma}(G)^2], \mathbb{V}[\hat{f}^{\wedge}_{i,\sigma}(G)]\}$. In Appendix C.9.2, we showed

$$\mathbb{V}[\hat{f}^{\wedge}_{i,\sigma}(G)] \leq \frac{nq_L(1-q_L)}{(1-2q_L)^2}. \tag{C.35}$$

To bound $\mathbb{V}[\hat{f}^{\wedge}_{i,\sigma}(G)^2]$, first we define $I = I_{-(\sigma(2i),\sigma(2i-1))}$ as a shorthand and plug in (3.14):

$$\mathbb{V}[\hat{f}^{\wedge}_{i,\sigma}(G)^2] = \frac{1}{(1-2q_L)^4}\mathbb{V}\left[\sum_{k,\ell\in I} y_k y_\ell\right].$$

We can write

$$\begin{aligned}
\mathbb{V}\left[\sum_{k,\ell\in I} y_k y_\ell\right] &= \sum_{k,\ell,k',\ell'\in I} Cov(y_k y_\ell, y_{k'} y_{\ell'}) \\
&= 4 \sum_{k,\ell,k' \text{ distinct in } I} Cov(y_k y_\ell, y_{k'} y_\ell) \\
&\quad + 4 \sum_{k,\ell \text{ distinct in } I} Cov(y_k y_\ell, y_k^2) \\
&\quad + \sum_{k\in I} Cov(y_k^2, y_k^2),
\end{aligned}$$

where in the second equality we have canceled the covariances equal to 0 due to independence. Note that the coefficient in the first term is 4 because $Cov(y_k y_\ell, y_{k'} y_\ell)$ captures $Cov(y_k y_\ell, y_{k'} y_\ell)$, $Cov(y_k y_\ell, y_\ell y_{k'})$, $Cov(y_\ell y_k, y_{k'} y_\ell)$, and $Cov(y_\ell y_k, y_\ell y_{k'})$. In other words, there are four possible combinations depending on the positions of two y_ℓ's. Similarly, the coefficient in the second term is 4 because $Cov(y_k y_\ell, y_k^2)$ captures $Cov(y_k y_\ell, y_k^2)$, $Cov(y_\ell y_k, y_k^2)$, $Cov(y_k^2, y_k y_\ell)$, and $Cov(y_k^2, y_\ell y_k)$.

Now, we have that

$$
\begin{aligned}
Cov(y_k y_\ell, y_{k'} y_\ell) &= \mathbb{E}[y_k y_\ell y_{k'} y_\ell] - \mathbb{E}[y_k y_\ell]\mathbb{E}[y_{k'} y_\ell] \\
&= \mathbb{E}[y_\ell^2]\mathbb{E}[y_k]\mathbb{E}[y_{k'}] - \mathbb{E}[y_\ell]^2\mathbb{E}[y_k]\mathbb{E}[y_{k'}] \\
&= \mathbb{E}[y_k]\mathbb{E}[y_{k'}]\mathbb{V}[y_\ell].
\end{aligned}
$$

We unconditionally have that $\mathbb{V}[y_\ell] \leq q_L$, and there are at most d_{max} choices for k such that $\mathbb{E}[y_k] = 1 - q_L$, and the remaining choices satisfy $\mathbb{E}[y_k] = q_L$. Finally, there are at most n choices for ℓ in I. Thus,

$$
\begin{aligned}
&\sum_{k,\ell,k' \text{ distinct in } I} Cov(y_k y_\ell, y_{k'} y_\ell) \\
&\quad \leq \sum_{k,\ell,k' \text{ distinct in } I} \mathbb{E}[y_k]\mathbb{E}[y_{k'}]\mathbb{V}[y_\ell] \\
&\quad \leq n q_L (d_{max}^2 (1 - q_L)^2 + 2 d_{max}(n - d_{max})(1 - q_L) q_L \\
&\qquad + (n - d_{max})^2 q_L^2) \\
&\quad \leq n q_L (d_{max}^2 + 2 n d_{max} q_L + n^2 q_L^2) \\
&\quad \leq n q_L (d_{max} + n q_L)^2.
\end{aligned}
$$

Now, using the fact that y_k is zero-one valued, we have

$$
Cov(y_k y_\ell, y_k^2) = Cov(y_k y_\ell, y_k) = \mathbb{E}[y_\ell]\mathbb{V}(y_k).
$$

There are at most n choices for k, and there are at most d_{max} choices such that $\mathbb{E}[y_\ell] = 1 - q_L$, and the remaining choices satisfy $\mathbb{E}[y_\ell] = q_L$. Thus,

$$
\sum_{k,\ell \text{ distinct in } I} Cov(y_k y_\ell, y_k^2)
$$

$$
\begin{aligned}
&\leq \sum_{k,\ell \text{ distinct in } I} \mathbb{E}[y_\ell]\mathbb{V}[y_k] \\
&\leq nq_L(d_{max}(1-q_L)+(n-d_{max})q_L) \\
&\leq nq_L(d_{max}+nq_L).
\end{aligned}
$$

Finally, $\mathbb{V}[y_k^2] = \mathbb{V}[y_k] \leq q_L$, and so $\sum_{k\in I} Cov(y_k^2, y_k^2) \leq nq_L$. Thus,

$$
\begin{aligned}
\mathbb{V}\left[\sum_{k,\ell\in I} y_k y_\ell\right] &\leq \big(4nq_L(d_{max}+nq_L)^2 \\
&\qquad + 4nq_L(d_{max}+nq_L)+nq_L\big) \\
&\leq 9nq_L(d_{max}+nq_L)^2.
\end{aligned}
$$

This implies

$$
\mathbb{V}[\hat{f}^{\wedge}_{i,\sigma}(G)^2] \leq \frac{9nq_L(d_{max}+nq_L)^2}{(1-2q_L)^4}. \tag{C.36}
$$

We clearly have that (C.36) is bigger than (C.35), and thus (C.36) is an upper bound for $\mathbb{V}_{RR}[\hat{f}^{\square}_{i,\sigma}(G)|\sigma]$. Thus, (C.33) is upper bounded by $\frac{9ntq_L(d_{max}+nq_L)^2}{(1-2q_L)^4}$.

Upper bounding (C.34)**:** By (3.16), we can write

$$
\mathbb{V}_\sigma\left[\sum_{i=1}^{t} \mathbb{E}_{RR}\left[\hat{f}^{\square}_{i,\sigma}(G)\middle|\sigma\right]\right] = \mathbb{V}_\sigma\left[\sum_{i=1}^{t} f^{\square}_{i,\sigma}(G)\right].
$$

When σ is chosen randomly, the random variables $f_{i,\sigma}(G)$ for $1 \leq i \leq t$ are a uniform sampling without replacement from the set

$$
\mathscr{F} = \left\{ f^{\square}_{i,j}(G) : i,j \in [n], i \neq j \right\}.
$$

Applying Lemma 5, we have

$$\mathbb{V}_\sigma\left[\sum_{i=1}^{t} f_{i,\sigma}^{\square}(G)\right] \leq t\mathbb{V}_\sigma[f_{1,\sigma}^{\square}(G)].$$

Now, we have

$$\begin{aligned}
\mathbb{V}_\sigma[f_{1,\sigma}^{\square}(G)] &\leq \mathbb{E}_\sigma[f_{1,\sigma}^{\square}(G)^2] \\
&= \frac{1}{4}\mathbb{E}_\sigma\left[(f_{i,\sigma}^{\wedge}(G)(f_{i,\sigma}^{\wedge}(G)-1))^2\right] \\
&\leq \frac{1}{4}\mathbb{E}_\sigma\left[f_{i,\sigma}^{\wedge}(G)^4\right] \\
&= \frac{1}{4n(n-1)}\sum_{1\leq i,j\leq n, i\neq j} f_{i,j}^{\wedge}(G)^4.
\end{aligned}$$

Let E^2 be the set of node pairs for which there exists a 2-hop path between them in G. We have $|E^2| \leq nd_{max}^2$. Now, we can write

$$\begin{aligned}
\frac{1}{4n(n-1)}\sum_{1\leq i,j\leq n, i\neq j} f_{i,j}^{\wedge}(G)^4 &= \frac{1}{4n(n-1)}\sum_{(i,j)\in E^2} f_{i,j}^{\wedge}(G)^4 \\
&\leq \frac{1}{4n(n-1)}\sum_{(i,j)\in E^2} d_{max}^4 \\
&\leq \frac{d_{max}^6}{4(n-1)}.
\end{aligned}$$

This allows to conclude that the variance of (C.34) is at most $\frac{td_{max}^6}{4(n-1)}$.

Putting it together: Substituting in (C.33) and (C.34), we obtain

$$\begin{aligned}
\mathbb{V}[\hat{f}^{\square}(G)] &\leq \frac{n^2(n-1)^2}{16t^2}\left(\frac{td_{max}^6}{4(n-1)} + \frac{9ntq_L(d_{max}+nq_L)^2}{(1-2q_L)^4}\right) \\
&\leq \frac{9n^5q_L(d_{max}+nq_L)^2}{16t(1-2q_L)^4} + \frac{n^3d_{max}^6}{64t}.
\end{aligned}$$

$\square$

Appendix D

D.1 Background Cntd.

Data: $l \in \{0,1\}^n$ - True adjacency list
Result: $q \in \{0,1\}^n$ - Reported (noisy) adjacency list
$\rho = \frac{1}{1+e^{\varepsilon}}$;
for $i \in [n]$ **do**
  $q[i] = RR_\rho(l[i])$;
end
return q

Algorithm 18: Definition of randomized response $RR_\rho : \{0,1\}^n \mapsto \{0,1\}^n$

D.2 Proofs

First, we introduce notation and preliminary results used in our proofs.

D.2.1 Notation

In this section, for a graph G with vertices $[n]$, we let $d_i(S)$ for $S \subseteq [n]$ denote the number of neighbors of node i in the set S. We will often abuse notation for a set $\mathscr{S}$ of users by also letting $\mathscr{S}$ be the indices of the users in the set. Thus, we may let $i \in \mathscr{S}$ be the index of some user in $\mathscr{S}$. Finally, we sometimes refer to user U_i simply as user i.

D.2.2 Preliminary Results

We will heavily make use of the following concentration result:

Lemma 6. *Let $X_1, \ldots, X_n$ denote independent random variables such that $X_i \sim \mathrm{Bernoulli}(p_i)$. Let $v = \sum_{i=1}^{n} p_i(1-p_i)$, and $X = \sum_{i=1}^{n} X_i$. Then,*

$$\Pr[|X - \mathbb{E}[X]| \geq \max\{1.5 \ln \frac{2}{\delta}, \sqrt{2v \ln \frac{2}{\delta}}\}] \leq \delta.$$

Proof. Center the random variables so that $Z_i = X_i - p_i$; the variance v does not change. We know by Bernstein's inequality that for all $t \geq 0$,

$$\Pr[Z \geq t] \leq \exp\left(\frac{-t^2}{2(v + t/3)}\right) \leq \exp\left(-\max\left\{\frac{t^2}{2v}, \frac{3t}{2}\right\}\right).$$

Thus, if $t \geq \max\{\frac{3}{2} \ln \frac{2}{\delta}, \sqrt{2v \ln \frac{2}{\delta}}\}$, then $\Pr[Z \geq t] \leq \frac{\delta}{2}$. Applying the argument to $-Z$, we obtain the two-sized bound.

□

Next, we observe the following facts about randomized response.

Fact 2. *If user $i \in \mathscr{H}$, then $\mathbb{E}[q_i[j]] = \rho + (1-2\rho)d_i(j)$.*

Fact 3. *If users $i, j \in \mathscr{H}$, then $\mathbb{E}[q_i[j]q_j[i]] = \rho^2 + (1-2\rho)d_i(j)$.*

D.2.3 Proof of Theorem 1

Recall that in the Laplace mechanism, a user's degree estimate $\hat{d_i}$ is simply $d_i + L_i$, where $L_i \sim Lap(\frac{1}{\varepsilon})$ is a Laplace random variable generated by the user.

Correctness. The correctness guarantee follows from the concentration of Laplace distribution: Each Laplace random variable L_i satisfies $|\Pr[|L_i| \geq t] \leq e^{-t\varepsilon}$. Setting $t = \frac{1}{\varepsilon} \ln \frac{n}{\delta}$ and applying the union bound, each of the n Laplace variables will satisfy $|L_i| \leq \frac{1}{\varepsilon} \ln \frac{\delta}{n}$ with probability $1 - \delta$, and if this holds, then $|d_i - \hat{d}_i| \leq \frac{1}{\varepsilon} \ln \frac{\delta}{n}$ for honest users.

Tight Soundness. Consider the empty graph. A malicious user U_i may report $n-1$, the maximum possible degree, and thus $\hat{d}_i = n-1$ while $d_i = 0$.

D.2.4 Proof of Theorem 2

Correctness.

As defined in *SimpleRR*, the estimator $count_i^1$ is given by

$$count_i^1 = (\sum_{j<i} q_j[i] + \sum_{i<j} q_i[j]) \tag{D.1}$$

We may alternatively split the above sum into honest bits and malicious bits as $count_i^1 = hon_i + mal_i$. Here,

$$hon_i = \sum_{j<i, j\in\mathscr{H}} q_j[i] + \sum_{i<j} q_i[j]$$
$$mal_i = \sum_{j<i, j\in\mathscr{M}} q_j[i].$$

Since all bits in the sum hon_i are honest, by Fact 2 we have $\mathbb{E}[hon_i] = \rho|\mathscr{H}_i| + (1-2\rho)d_i(\mathscr{H}_i)$, where $\mathscr{H}_i = \mathscr{H} \cup \{1,2,\ldots,i-1\}$.

Furthermore, $0 \leq mal_i \leq |\mathscr{M}_i|$, where $\mathscr{M}_i = [n] \setminus \mathscr{H}_i$. This implies $|mal_i - E_{mal,i}| \leq |\mathscr{M}_i|$, where $E_{mal,i} = \rho|\mathscr{M}_i| + (1-2\rho)d_i(\mathscr{M}_i)$. By Lemma 6 and a union bound, with probability $1-\delta$, we have for all $i \in \mathscr{H}_i$ that

$$\begin{aligned}
\left|hon_i - \mathbb{E}[hon_i] + mal_i - E_{mal,i}\right| &\leq \sqrt{2\rho n \ln \frac{2n}{\delta}} + |\mathscr{M}_i| \\
\implies \left|count_i^1 - \rho n - (1-2\rho)d_i\right| &\leq \sqrt{2\rho n \ln \frac{2n}{\delta}} + m \\
\implies |\hat{d}_i - d_i| &\leq \frac{1}{1-2\rho}\sqrt{2\rho n \ln \frac{2n}{\delta}} + \frac{m}{1-2\rho}.
\end{aligned}$$

Tight Soundness. Consider the empty graph, and suppose that user n is malicious. Since this user reports all his edges, he may report $q_i[j] = 1$ for all $j < 1$. Thus, $\hat{d}_n \geq n-1$, but $d_n = 0$, showing $n-1$-tight soundness.

D.2.5 Proof of Theorem 3

Recall the key quantities defined in *RRCheck* (Algorithm 12):

$$count_i^{11} = \sum_{j \in [n] \setminus i} q_i[j] q_j[i] \tag{D.2}$$

$$count_i^{01} = \sum_{j \in [n] \setminus i} (1 - q_i[j]) q_j[i]. \tag{D.3}$$

We now prove correctness.

Correctness. It will be helpful to split $count_i^{11} = hon_i^{11} + mal_i^{11}$, where $hon_i^{11} = \sum_{j \in \mathscr{H} \setminus i} q_i[j] q_j[i]$ and $mal_i^{11} = \sum_{j \in \mathscr{M} \setminus i} q_{ij} q_{ji}$. We define hon_i^{01} and mal_i^{01} similarly such that they satisfy $count_i^{01} = hon_i^{01} + mal_i^{01}$. We break the proof into two claims: showing that honest users receive an accurate estimate and that they are not disqualified.

Claim 1. *We have*

$$\Pr[\forall U_i \in \mathscr{H}.\ |\hat{d}_i - d_i| \geq \frac{m + 2\sqrt{\rho n \ln \frac{4n}{\delta}}}{1 - 2\rho}] \leq \frac{\delta}{2}.$$

Proof. Let $U_i \in \mathscr{H}$. Then, hon_i^{11} is a sum of $h-1$ Bernoulli random variables with $p = \rho^2$ or $(1-\rho)^2$. By Fact 3, we have

$$\mathbb{E}[hon_i^{11}] = \rho^2 (h-1) + (1 - 2\rho) d_i(\mathscr{H})$$

Now, v defined in Lemma 6 satisfies $(h-1)\rho^2 \leq v \leq (h-1)(1-(1-\rho)^2) \leq 2(h-1)\rho$. Applying the Lemma and a union bound, we have with probability at least $1 - \frac{\delta}{2}$ that for all $i \in \mathscr{H}$,

$$|hon_i^{11} - \mathbb{E}[hon_i^{11}]| \leq 2\sqrt{(h-1)\rho \ln \frac{4n}{\delta}}. \tag{D.4}$$

On the other hand, we have that $0 \leq mal_i^{11} \leq m$, so if we let $E_{mal,i}^{11} = \rho^2 m + (1 - 2\rho)d_i(\mathscr{M})$ (defined for convenience later), then $|mal_i^{11} - E_{mal,i}^{11}| \leq m$.

Applying the triangle inequality, the following holds over all $i \in \mathscr{H}$:

$$
\begin{aligned}
&|hon_i^{11} - \mathbb{E}[hon_i^{11}] + mal_i^{11} - E_{mal,i}^{11}| \leq m + 2\sqrt{\rho n \ln \tfrac{4n}{\delta}} \\
\implies & |count_i^{11} - \rho^2(n-1) - (1-2\rho)d_i| \leq m + 2\sqrt{\rho n \ln \tfrac{4n}{\delta}} \\
\implies & |\hat{d}_i - d_i| \leq \frac{m + 2\sqrt{\rho n \ln \frac{4n}{\delta}}}{1 - 2\rho}
\end{aligned}
$$

This proves the claim. $\square$

Next, we show that honest users are not likely to be disqualified.

Claim 2. *We have*

$$\Pr[\forall U_i \in \mathscr{H}.\ |count_i^{01} - \rho(1-\rho)(n-1)| \geq \tau] \leq \frac{\delta}{2},$$

where $\tau = m + \sqrt{2\rho n \ln \frac{4n}{\delta}}$

Proof. Let U_i be honest. Then, the quantity hon_i^{01} consists of $h-1$ Bernoulli random variables drawn from $\rho(1-\rho)$. We have

$$\mathbb{E}[hon_i^{01}] = \rho(1-\rho)(h-1).$$

As defined in Lemma 6, v satisfies $\frac{1}{2}(h-1)\rho \leq P \leq (h-1)\rho$. Applying the Lemma and a union bound, we have with probability $1 - \frac{\delta}{2}$ that for all $i \in \mathscr{H}$,

$$|hon_i^{01} - \mathbb{E}[hon_i^{01}]| \leq \sqrt{2\rho(h-1) \ln \tfrac{4n}{\delta}} \tag{D.5}$$

Noticing that $|mal_i^{01} - m\rho(1-\rho)| \leq m$, we have by the triangle inequality that

$$|count_i^{01} - \rho(1-\rho)(n-1)| \geq m + \sqrt{2\rho n \ln \frac{4n}{\delta}}.$$

This concludes the proof. $\square$

Putting it together, $\left(m(\frac{e^\varepsilon+1}{e^\varepsilon-1}) + \sqrt{n}\frac{2\sqrt{(e^\varepsilon+1)\ln\frac{4n}{\delta}}}{e^\varepsilon-1}, \delta\right)$-correctness follows.

Soundness.

When player i is a malicious player, we can still prove a tight bound on $count_i^{11} + count_i^{01}$, and this combined with the check in *SimpleRR* means that his degree estimate will be accurate.

Claim 3. *We have*

$$\Pr[\forall i \in \mathscr{M}.\ |count_i^{11} + count_i^{01} \\ - (1-2\rho)d_i - \rho(n-1)| \leq \tau] \geq 1-\delta,$$

where $\tau = m + \sqrt{2\rho n \ln \frac{4n}{\delta}}$.

Proof. Observe that $count_i^{11} + count_i^{01} = \sum_{j=1, j\neq i}^n q_j[i]$. Let hon_i^1 denote the sum of the $q_j[i]$ where j is honest, and mal_i^1 denote the sum of the malicious players. By Fact 2, we have $\mathbb{E}[hon_i^1] = d_i(\mathscr{H})(1-2\rho) + h\rho$. Applying a union bound over Lemma 6, for all $i \in \mathscr{M}$, we have with probability at most δ that

$$|hon_i^1 - \mathbb{E}[hon_i^1]| \geq \sqrt{2\rho n \ln \frac{2m}{\delta}} \tag{D.6}$$

Because $|mal_i^1 - (1-2\rho)d_i(\mathscr{M}) - \rho(m-1)| \leq m$, the claim follows from the triangle inequality. $\square$

To conclude the proof, consider any malicious user $i \in \mathscr{M}$ is not disqualified ($\hat{d}_i \neq \perp$), as if he is then the soundness event trivially happens. Thus, it must be true that $|count_i^{01} - (n -$

$1)\rho(1-\rho)| \leq \tau$. However, given this and the event in Claim 3 holds, it follows by the triangle inequality that

$$|count_i^{11} - (1-2\rho)d_i - \rho^2(n-1)| \leq 2\tau$$
$$|\hat{d}_i - d_i| \leq \frac{2\tau}{1-2\rho}$$

This establishes $\left(2m(\frac{e^\varepsilon+1}{e^\varepsilon-1}) + 4\sqrt{n}\frac{\sqrt{(e^\varepsilon+1)\ln\frac{4n}{\delta}}}{e^\varepsilon-1}, \delta\right)$-soundness.

D.2.6 Proof of Theorem 4

Correctness. Let honest U_i share an edge with all malicious users in $\mathscr{M}$. Now, all the malicious users can lie and report 0 for U_i, i.e., set $q_j[i] = 0 \forall j \in \mathscr{M}$. This deflates U_i's degree by m.

Soundness. Let malicious U_i share an edge with all users in the graph. Consider the attack where U_i and $U_j, j \in \mathscr{M} \setminus i$ report 0 for their edges, and additionally U_i reports 0 for $\min\{m-1, n-m\}$ additional honest users. In this way, U_i can deflate its degree estimate by $\min(2m-1, n)$.

D.2.7 Proof of Theorem 5

Correctness. By Claim 2, the first check in *Hybrid* will not set $\hat{d}_i = \perp$ for any honest user with probability at least $1 - \frac{\delta}{4}$. The variables $\hat{d}_i^{rr}$ in *Hybrid* behave identically to $\hat{d}_i$ in *RRCheck*. By Claim 1 we have for all users, $|\hat{d}_i^{rr} - d_i| \leq \frac{m+2\sqrt{\rho n \ln\frac{8n}{\delta}}}{1-2\rho}$, with probability at least $1 - \frac{\delta}{4}$.

By concentration of Laplace random variables, we have for all $i \in \mathscr{H}$ that $|\hat{d}_i^{lap} - d_i| \leq \frac{1}{\varepsilon}\ln\frac{2n}{\delta}$ with probability at least $1 - \frac{\delta}{2}$, and by the triangle inequality we have $|\hat{d}_i^{lap} - \hat{d}_i^{rr}| \leq \frac{m+2\sqrt{\rho n \ln\frac{8n}{\delta}}}{1-2\rho} + \frac{1}{\varepsilon}\ln\frac{2n}{\delta}$. Thus, the second check will not set $\hat{d}_i = \perp$ assuming these events hold, and the estimator $\hat{d}_i$ satisfies the correctness bound of $\hat{d}_i^{lap}$.

Soundness. Following the same argument we saw in the soundness proof of Theorem 3, we can have that, with probability at least $1 - \frac{\delta}{2}$, for all malicious users $i \in \mathscr{M}$, we have $|\tilde{d}_i^{rr} -$

$d_i| \leq \frac{2\tau}{1-2\rho}$. Suppose that $\hat{d_i}$ is not set to be $\perp$. This implies that $|\tilde{d}_i^{rr} - \tilde{d}_i^{lap}| \leq \frac{2\tau}{1-2\rho} + \frac{1}{\varepsilon}\log\frac{2n}{\delta}$. By the triangle inequality, this implies

$$|\tilde{d}_i^{rr} - d_i| \leq \frac{4\tau}{1-2\rho} + \frac{1}{\varepsilon}\log\frac{2n}{\delta}.$$

This establishes $(\frac{4\tau}{1-2\rho} + \frac{1}{\varepsilon}\log\frac{2n}{\delta}, \delta)$-soundness.

D.2.8 Proof of Theorem 6

Correctness. The correctness guarantee follows in the same way as Theorem 1.

Soundness. Consider a malicious user U_i, and let m_i be the malicious degree estimate sent by U_i, with $0 \leq m_i \leq n-1$. The estimator is given by $\hat{d_i} = m_i + \eta, \eta \sim Lap(\frac{1}{\varepsilon})$. Thus, $\Pr[|d_i - m_i - \eta| \geq n-1] \leq \Pr[\eta > 0] \leq \frac{1}{2}$.

D.2.9 Proof of Theorem 7

Correctness. We follow the correctness proof of Theorem 2, with the following change. Observe that mal_i consists of $|\mathscr{M}_i|$ Bernoulli random variables of mean either ρ or $1-\rho$. Thus, with probability $1-\frac{\delta}{2}$, we have $|mal_i - \mathbb{E}[mal_i]| \leq \sqrt{2m\ln\frac{4m}{\delta}}$ for all $i \in \mathscr{M}$.

Thus, we can show $|mal_i - E_{mal,i}| \leq (1-2\rho)|\mathscr{M}_i|$, where $E_{mal,i} = \rho|\mathscr{M}_i| + (1-2\rho)d_i(M_i)$. Finishing the proof, we can show

$$|\hat{d_i} - d_i| \leq \frac{1}{1-2\rho}(\sqrt{2\rho n\ln\frac{4n}{\delta}} + \sqrt{2m\ln\frac{4m}{\delta}}) + m.$$

Soundness.

In order for $|d_i - \hat{d_i}| = n-1$, it is necessary for $|count_i^1 - \rho(n-1) - (1-2\rho)d_i| \geq (1-2\rho)(n-1)$. We have $count_i^1$ is a sum of $n-1$ Bernoulli random variables of mean either ρ or $1-\rho$, so it can be written as $\mu + Z_i$, where Z_i is approximately a normal random variable of mean 0. Observe that, since μ and $\rho(n-1)+(1-2\rho)d_i$ are in the interval $[\rho(n-1),(1-\rho)(n-1)]$, it is impossible for the difference $\mu - \rho(n-1) + (1-2\rho)d_i$ to exceed $(1-2\rho)(n-1)$ unless Z_i has the correct sign, which happens with probability at most $\frac{1}{2}$. This establishes $(n-1,\frac{1}{2})$-soundness.

D.2.10 Proof of Theorem 8

Correctness. Our proof follows that of Theorem 3. We are able to prove stronger versions of the claims.

Claim 4. *We have*

$$\Pr[\forall i \in \mathscr{H}.\ |\hat{d}_i - d_i| \geq m + \frac{\sqrt{8\max\{\rho n, m\}\ln\frac{8n}{\delta}}}{1-2\rho}] \leq \frac{\delta}{2}.$$

Proof. We can control hon_i^{11} in exactly the same way as in Claim 1, so (D.4) holds with probability $1-\frac{\delta}{4}$, for all $i \in \mathscr{H}$. On the other hand, we know that mal_i^{11} is now a sum of $d_i(\mathscr{M})$ Bernoulli random variables with bias either $(1-\rho)^2$ or $(1-\rho)\rho$, plus a sum of $m-d_i(\mathscr{M})$ Bernoulli random variables with bias either $\rho(1-\rho)$ or ρ^2. Thus,

$$\begin{aligned}\rho(1-2\rho)d_i(\mathscr{M}) + \rho^2 m &\leq \mathbb{E}[mal_i^{11}] \\ &\leq (1-\rho)(1-2\rho)d_i(\mathscr{M}) + \rho(1-\rho)m.\end{aligned}$$

From this, we can show $|\mathbb{E}[mal_i^{11}] - E_{mal,i}^{11}| \leq (1-2\rho)m$, where $E_{mal,i}^{11} = \rho^2 m + (1-2\rho)d_i(\mathscr{M})$. Applying Hoeffding's inequality, we conclude that with probability at least $1-\frac{\delta}{4}$, for all $i \in \mathscr{H}$,

$$|mal_i^{11} - \mathbb{E}[mal_i^{11}]| \geq \sqrt{2m\ln\frac{8n}{\delta}}$$

Thus, $|mal_i^{11} - E_{mal,i}^{11}| \leq (1-2\rho)m + \sqrt{2m\ln\frac{8n}{\delta}}$. Applying the triangle inequality, we obtain

$$\begin{aligned}&\Pr[|hon_i^{11} + mal_i^{11} - \mathbb{E}[hon_i^{11}] - E_{mal,i}^{11}| \\ &\qquad \geq \sqrt{2m\ln\frac{8n}{\delta}} + (1-2\rho)m + 2\sqrt{\rho n\ln\frac{8n}{\delta}}] \leq \frac{\delta}{2}.\end{aligned}$$

The result follows in the same way as in Claim 1. $\square$

Claim 5. *We have*

$$\Pr[\forall i \in \mathscr{H}.\ |count_i^{01} - \rho(1-\rho)(n-1)| \geq \tau] \leq \tfrac{\delta}{2},$$

where $\tau = m(1-2\rho) + \sqrt{8\max\{\rho n, m\}\ln\frac{8n}{\delta}}$.

Proof. We can follow the same line of reasoning as Claim 2 and conclude that (D.5) holds. Similar to Claim 4, we can show that $|mal_i^{01} - \rho(1-\rho)m| \leq (1-2\rho)m + \sqrt{2m\ln\frac{8n}{\delta}}$ with probability at least $\frac{\delta}{4}$, and applying the triangle inequality, we see

$$\Pr[|count_i^{01} - \rho(1-\rho)n| \geq m(1-2\rho) + \sqrt{2m\ln\tfrac{8n}{\delta}} + \sqrt{2\rho n\ln\tfrac{8n}{\delta}}] \leq \tfrac{\delta}{2}.$$

$\square$

The $(2m + \frac{4\sqrt{2\max\{\rho n, m\}\ln\frac{8n}{\delta}}}{1-2\rho}, \delta)$-correctness guarantee follows from the union bound over the two claims.

Soundness. When player i is a malicious player, he is still subject to the following claim:

Claim 6. *We have*

$$\Pr[\forall i \in \mathscr{M}.\ |count_i^{11} + count_i^{01} - (1-2\rho)d_i - \rho(n-1)| \leq \tau|] \geq 1-\delta,$$

where $\tau = m(1-2\rho) + \sqrt{8\max\{\rho n, m\}\ln\frac{8n}{\delta}}$.

Proof. Observe that $count_i^{11} + count_i^{01} = \sum_{j=1, j\neq i}^{n} q_j[i] = hon_i^1 + mal_i^1$. With the same argument as in Claim 3, we know that (D.6) holds. Similarly, each random variable in mal_i^1 comes from either Bernoulli(ρ) or Bernoulli($1-\rho$), and thus with probability at least $1-\frac{\delta}{2}$, for all $i \in \mathscr{M}$

$$|mal_i^1 - \mathbb{E}[mal_i^1]| \leq \sqrt{2m \ln \frac{4m}{\delta}}$$

Since $\mathbb{E}[mal_i^1] \in [\rho m, (1-\rho)m]$, This implies that $|mal_i^1 - \rho m| \leq (1-2\rho)m + \sqrt{2m \ln \frac{4m}{\delta}}$. Thus, the claim follows. □

Having established this claim, we can prove $(2m + 4\sqrt{2\max\{\rho n, m\} \ln \frac{8n}{\delta}}, \delta)$-soundness using an identical method as in the proof of soundness for Theorem 3.

D.2.11 Proof of Theorem 9

Correctness. As input manipulation attacks are a subset of response manipulation attacks, the same correctness guarantee as Theorem 5 holds.

Soundness. The proof of this is similar to the soundness proof of Theorem 5, using previous results in Theorem 8.

D.3 Evaluation Cntd.

In this section, we describe the specific implementations of the attacks we use for our evaluation in Section 4.9.

D.3.1 Attacks Against *RRCheck*

Degree Inflation Attacks

Let $U_t, t \in \mathscr{M}$ denote the target malicious user.

Input Poisoning. In this attack, the non-target malicious users set the bit for U_t to be 1. The target malicious user constructs his input by setting 1 for all other malicious users. They also report 1 for honest users to which they share an edge.

For honest users to which U_t does not share an edge, U_t flips some of the bits to 1 with the hopes of artificially increasing his degree. He does this for a r_1-fraction of these neighbors.

Data: $l \in \{0,1\}^n$ - True adjacency list, t - Target honest user
Result: $q \in \{0,1\}^n$ - Reported adjacency list
Select $r_1 \in [0,1]$;
$\mathscr{H}_1 = \{i \in \mathscr{H} | l[i] = 1\}$;
$\triangleright$ $\mathscr{H}_1$ is the set of honest users with a mutual edge;
$\mathscr{H}_0 = \mathscr{H} \setminus \mathscr{H}_1$;
$\triangleright$ $\mathscr{H}_0$ is the set of honest users without a mutual edge;
$F \in_R \mathscr{H}_0, |F| = r_1|\mathscr{H}_0|$;
$\triangleright$ Randomly sample r_1 fraction of the users in $\mathscr{H}_0$;
$l' = \{0,0,\cdots,0\}$;
for $i \in \mathscr{H}_1 \cup \mathscr{M} \cup F$ **do**
    $l'[i] = 1$;
end
for $i \in [n]$ **do**
    $q[i] = RR_\rho(l'[i])$;
end
return q

Algorithm 19: Definition of input poisoning attack $A^{inp}_{RRCheck} : \{0,1\}^n \mapsto \{0,1\}^n$

See Algorithm 19 for the details; we term this attack $A^{inp}_{RRCheck}$. Note that if $r_1 = 0$, then the malicious user is being completely honest for these users and will not inflate his degree, and if $r_1 = 1$, then he lies about each of these users and will likely be caught. Thus, his strategy is to pick a value in between 0 and 1, and in the experiments we found that $r_1 = 15\%$ was a good tradeoff point.

Response Poisoning. For response poisoning, the non-target malicious first find a plausible response by applying RR_ρ to their data. They then set the bit for U_t to be 1, indicating they are connected to this user.

The target malicious user constructs his response by first applying RR_ρ to his data to compute a plausible response. Then, he flips his bits to malicious users to 1, and for honest users, he takes a r_1-fraction of the 0s in his response and flips them to 1. The quantity r_1 is a tradeoff parameter with the same intuition as for $A^{inp}_{RRCheck}$. The details of this attack appear in Algorithm 20, and it is termed $A^{resp}_{RRCheck}$.

Data: $l \in \{0,1\}^n$ - True adjacency list
Result: $q \in \{0,1\}^n$ - Reported adjacency list
Select $r_1 \in [0,1]$;
$q = \mathsf{RR}_\rho(l)$;
$\mathscr{I}_1 = \{i \in \mathscr{H} | q[i] = 1\}$;
▷ $\mathscr{H}_1$ is the set of honest users with an edge in q;
$\mathscr{I}_0 = \mathscr{H} \setminus \mathscr{I}_1$;
$F \in_R \mathscr{I}_0, |F| = r_1|\mathscr{I}_0|$;
▷ Randomly sample r_1 fraction of the users in $\mathscr{I}_0$;
for $i \in \mathscr{I}_1 \cup \mathscr{M} \cup F$ **do**
  $q[i] = 1$;
end
return q

Algorithm 20: Definition of response poisoning attack $A^{resp}_{RRCheck} : \{0,1\}^n \mapsto \{0,1\}^n$

Degree Deflation Attacks

Let $U_t, t \in \mathscr{H}$ denote the target honest user.

Input Poisoning. Here, every malicious user constructs his input acting honestly for non-target users and setting a 0 for U_t.

Response Poisoning. Every malicious user acts honestly for non-target users by applying randomized response to their input. They finally send a 0 for their connection to U_t.

D.3.2 Attacks Against *Hybrid*

Degree Inflation Attacks

Let $U_t, t \in \mathscr{M}$ be the target malicious user.

Input Poisoning. The non-target malicious users flip their edge to U_t to a 1 as they do in $A^{inp}_{RRCheck}$. They send an honest estimate of their degree $\tilde{d}^{Lap}$ as this does not affect the target.

The target malicious user crafts his input adjacency list q as he did in $A^{inp}_{RRCheck}$. For his estimate $\tilde{d}^{Lap}_t$, he computes the expected value of $\tilde{d}^{rr}_t$ given that he submitted q while the other users either submit $RR_\rho(l_i)$ or $RR_\rho(1)$, depending if they are honest or malicious. Specifically,

Data: $l \in \{0,1\}^n$ - True adjacency list
Result: $q \in \{0,1\}^n$ - Reported adjacency list, $\tilde{d}^{lap}$ - Reported noisy degree estimate
Select $r_2 \in [0,1]$;
$\rho = \frac{1}{1+e^{c\varepsilon}}$;
    ▷ c is the constant used in Alg. 14 to divide the budget between the RR and Laplace steps;
$q \leftarrow A^{inp}_{RRCheck}(l, c\varepsilon)$;
$\tilde{count}^{11} \leftarrow m(1-\rho)^2 + \mathbb{E}[\sum_{i\in\mathscr{H}} q_i RR_\rho(l_i)]$;
$e^{rr,inp} = \frac{\tilde{count}^{11} - \rho^2 n}{1-2\rho}$;
$\hat{d}^{Lap} = e^{rr,inp} + r_2 \frac{\tau}{1-2\rho} + \eta$ where $\eta \sim Lap(\frac{1}{(1-c)\varepsilon})$;
return $q, \tilde{d}^{Lap}$

Algorithm 21: Definition of input poisoning attack $A^{inp}_{Hybrid} : \{0,1\}^n \mapsto \{0,1\}^n$

the expected value is given by

$$e^{rr,inp} = \frac{m(1-\rho)^2 + \mathbb{E}[\sum_{i\in\mathscr{H}} q_i RR_\rho(l_i)] - \rho^2 n}{1-2\rho}.$$

He finally sets $\tilde{d}^{rr}_t = e^{rr,inp}_t + r_2 \frac{\tau}{1-2\rho}$ where $r_2 \in [0,1]$, which again trades off between how much cheating is possible and getting flagged. During the trials, we used $q_2 = 0.1$ as this did not significantly increase the target's chance of being rejected as $\perp$. This attack, termed A^{inp}_{Hybrid}, appears in Algorithm 21.

Response Poisoning. The non-target malicious users flip their edge to U_t to a 1 as they do in $A^{inp}_{RRCheck}$. They send an honest estimate of their degree $\tilde{d}^{Lap}$ as this does not affect the target.

The target malicious user crafts his response adjacency list q as he did in $A^{resp}_{RRCheck}$. For his estimate $\tilde{d}^{Lap}_t$, he computes the expected value of $\tilde{d}^{rr}_t$ given that he submitted q while the other users either submit $RR_\rho(l_i)$ or 1, depending if they are honest or malicious. This expected value is given by

$$e^{rr,resp} = \frac{m + \mathbb{E}[\sum_{i\in\mathscr{H}} q_i RR_\rho(l_i) - \rho^2 n]}{1-2\rho}.$$

He finally sets $\tilde{d}^{rr}_t = e^{rr,resp} + r_2 \frac{\tau}{1-2\rho}$ where $r_2 \in [0,1]$ serves a similar tradeoff purpose as for A^{inp}_{Hybrid}.

Data: $l \in \{0,1\}^n$ - True adjacency list
Result: $q \in \{0,1\}^n$ - Reported adjacency list, $\tilde{d}^{lap}$ - Reported noisy degree estimate
1 Select $r_2 \in [0,1]$;
2 $q = A^{resp}_{RRCheck}(l, \varepsilon)$;
3 $\rho = \frac{1}{1+e^{c\varepsilon}}$;
4 $\triangleright$ c determines how the privacy budget is divided between the two types of response as in Alg. 14;
5 $\tilde{count}^{11} \leftarrow m + \mathbb{E}[\sum_{i \in \mathscr{H}} q_i RR_\rho(l_i)]$;
6 $e^{rr,resp} = \frac{m + \tilde{count}^{11} - \rho^2 n}{1-2\rho}$;
7 $\tilde{d}^{Lap} = e^{rr,resp} + r_2 \frac{\tau}{1-2\rho}$;
8 **return** $q, \tilde{d}^{Lap}$

Algorithm 22: Definition of response poisoning attack $A^{resp}_{Hybrid} : \{0,1\}^n \mapsto \{0,1\}^n$.

Degree Deflation Attacks

Let $U_t, t \in \mathscr{H}$ represent the honest target.

Input Poisoning. For the adjacency list, all the malicious users follow the same protocol as for *RRCheck*. For the degree, all the malicious users follow the Laplace mechanism truthfully as these values are immaterial for estimating the degree of the target honest user.

Response Poisoning. For the adjacency list, all the malicious users follow the same protocol as for $RRCheck(\cdot)$. For the degree, all the malicious users follow the Laplace mechanism truthfully as these values are immaterial for estimating the degree of the target honest user.

D.3.3 Configurations Ctnd.

Our theoretical results suggested setting $\tau = m + C\sqrt{\rho n}$, where C is a constant that is obtained from Chernoff's bounds, for the different input and response manipulation attacks. The constant C is not tight, and for the practical interest of using as small a threshold as possible, we sought to set τ as small as possible so as not to falsely flag any honest user. Note that lower the threshold, lower is the permissible skew (α_1 and α_2 for correctness and soundness, respectively) introduced by poisoning, thereby improving the robustness of our protocols. We ran preliminary experiments using 50 runs of each protocol on both graphs, and we found that at all values of ε, setting $\tau = m + 0.4\sqrt{\rho n}$ (for $m = 40$) and $\tau = m + 0.1\sqrt{\rho n}$ (for $m = 1500$) did not result in any false positives. Thus, we used these smaller thresholds in our experiments, and throughout the experiments there were no false positives.

Appendix E

E.1 Related Work

Differential Privacy

Differential privacy [72] has recently become the gold standard of privacy used by institutions such as the US census [71] and large tech companies [83]. In a nutshell, DP algorithms provide plausible deniability for the input data of any user. There is a vast literature on DP algorithms for a disparate range of problems and many different models for differential privacy [72, 145, 39, 173, 137, 71] (we refer to [73] for a survey).

Among this rapidly growing literature, our work builds on multiple work on differentially privacy, namely DP PCA algorithms [75], DP Johnson Lindenstrauss projections [31], DP cut sparsification in graphs [79] as well as DP stochastic block model reconstruction (reviewed later).

Private graph algorithms

Especially relevant to this work is the area of differential privacy in graphs. DP has been declined in graph problems both as the edge-level [80, 79] and node-level model [122]. The most related work in this area is that on graph cut approximation [79, 10], as well as that of graph clustering with DP in correlation clustering model [32, 52].

Hierarchical Clustering

As we discussed in the introduction, hierarchical clustering has been studied for decades in multiple fields. For this reason, a significant number of algorithms for hierarchical clustering have been introduced [157]. Up until recently [57], most work on hierarchical clustering has been heuristic in nature, defining algorithms based on procedures without specific theoretical guarantees in terms of approximation. Most well-known among such algorithms are the linkage-based ones [106, 20]. [57] introduced for the first time a combinatorial approximation objective for hierarchical clustering which is the one studied in this paper. Since this work, many authors have designed algorithms for variants of the problem [53, 54, 37, 154, 4, 38] exploring maximization/minimization versions of the problem on dissimilarity/similarity graphs.

Limited work has been devoted to DP hierarchical clustering algorithms. One paper [214] initiates private clustering via MCMC methods, which are not guaranteed to be polynomial time. Follow-up work [125] shows that sampling from the Boltzmann distribution (essentially the exponential mechanism [145] in DP) produces an approximation to the maximization version of Dasgupta's function, which is a different problem formulation. Again, this algorithm is not provably polynomial time.

Private flat clustering

Contrary to hierarchical clustering, the area of private *flat* clustering on metric spaces has received large attention. Most work in this area has focus on improving the privacy-approximation trade-off [91, 14] and on efficiency [99, 51, 50].

Stochastic block models

The Stochastic Block Model (SBM) is a classic model for random graphs with planted partitions which has received significant attention in the literature. Most work in this area has focus on providing exact or approximate recovery of communities for increasingly more difficult regimes of the model [95, 150, 149, 87, 63, 135]. Specifically for our work, we focus on a variant of the model which has nested ground-truth communities arranged in a hierarchical fashion. This model has received attention for hierarchical clustering [53].

The study of private algorithms for SBMs is instead very recent and no work has addressed private recovery for hierarchical SBMS. One of the only results known for private (non-hierarchical) SBMs is the work of [179] which provides a quasi-polynomial time algorithm for some regimes of the model. This paper require either non-poly time or $\varepsilon \in \Omega(\log(|V|))$. Finally, very recently and currently to our work, the manuscript of [40] has been published. This work provides strong approximation guarantees using semi-definite programming for recovering SBM communities. None of these papers can be used directly to approximate hierarchical clustering on HSBMs. For this reason in Section 5.6 we design a hierarchical clustering algorithm (Algorithm 15) which uses as subroutine a DP SBM community detection algorithm. Moreover, we show a novel algorithm for SBMs (Algorithm 16) (independent to that of [40]) which is of practical interest as it does not require procedure with large polynomial dependency on the size of the input, such solving a complex semi-definite program.

E.2 Omitted proofs from Section 5.4

E.2.1 Proof of Lemma 2

We start with the following lemma:

Lemma 4. *Let G_1, G_2 be two graphs drawn uniformly at random from $\mathscr{P}(n,5)$. Let $\alpha = \frac{1}{100}$. The probability that there exists a balanced cut (A,B) which misses at most $\frac{\alpha}{5}n$ of the cycles for both G_1, G_2 is at most $2^{-0.4n}$.*

Proof. Let (A,B) be any balanced cut with $|A| = \beta n$, for $\frac{1}{3} \leq \beta \leq \frac{2}{3}$. Let $\mathscr{E}_1(A,B)$ be the event that (A,B) misses at most $\frac{\alpha}{X}n$ cycles in G_1, and define $\mathscr{E}_2(A,B)$ similarly for G_2. We observe the desired probability can be upper bounded by

$$\sum_{(A,B) \text{ a balanced cut}} \Pr[\mathscr{E}_1(A,B)] \Pr[\mathscr{E}_2(A,B)]. \tag{E.1}$$

In the above sum, the balanced cuts (A,B) are fixed, and the graphs G_1, G_2 are generated independently. We consider an equivalent random process, where $G_1 \in \mathscr{P}(n,5)$ is fixed, and then (A,B) is generated by picking a uniformly random string $S \in \{0,1\}^n$ with βn 1s. There are $\binom{n}{\beta n}$ possible strings. We will now upper bound the number of strings for which $\mathscr{E}_1(A,B)$ holds. When $\mathscr{E}_1(A,B)$ holds, we can choose c cycles which are monochromatic 1s, where c is a non-negative integer such that $5c < n$, plus $\frac{\alpha n}{5}$ cycles which are not necessarily monochromatic. Within these $\frac{\alpha n}{5}$ cycles, there are αn vertices from which we can choose $d \leq \alpha n$ remaining 1s. The total number of 1s is $5c+d$, and thus $5c+d = \beta n$. Thus, the total number of admissible strings is at most

$$\sum_{5c+d=\beta n, d\leq \alpha n} \binom{n/5}{c}\binom{n/5}{\alpha n/5}\binom{\alpha n}{d}.$$

We make the simple observation that $\binom{\alpha n}{d} \leq 2^{\alpha n}$. Furthermore, we observe that there are $\frac{\alpha n}{5}$ admissible choices of c, d. In the following, we use the fact that $2^{H_2(\beta)n - \ln n} \leq \binom{n}{\beta n} \leq 2^{H_2(\beta)n}$, where $H_2(p)$ is the binary entropy function. We upper bound the number of admissible strings with

$$\begin{aligned}\frac{\alpha n}{5} \max_{(\beta-\alpha)n \leq 5c \leq \beta n} \binom{n/5}{c}\binom{n/5}{\alpha n/5} 2^{\alpha n} &\leq \frac{\alpha n}{5} \max_{(\beta-\alpha)n \leq 5c \leq \beta n} 2^{H_2(5c/n)n/5} 2^{H_2(\alpha)n/5} 2^{\alpha n} \\ &\leq n 2^{H_2(\beta)n/5} 2^{H_2(\alpha)n/5} 2^{\alpha n}.\end{aligned}$$

Dividing this number by $\binom{n}{\beta n}$, the total possible number of strings, we obtain

$$\begin{aligned}\Pr[\mathscr{E}_1(A,B)] &\leq \frac{n 2^{(H_2(\beta)+H_2(\alpha))n/5 + \alpha n}}{2^{H_2(\beta)n - \ln n}} \\ &\leq 2^{\left(\frac{H_2(\beta)+H_2(\alpha)}{5} + \alpha - H_2(\beta)\right)n + \ln n} \\ &\leq 2^{-0.7n},\end{aligned}$$

where the last line follows from the fact that $\frac{1}{3} \leq \beta \leq \frac{2}{3}$ and that $\alpha = \frac{1}{100}$ so that $H_2(\alpha) \leq 0.081$. By a similar argument, we have $\Pr[A_2(B)] \leq 2^{-0.7n}$.

Thus, (E.1) can be upper bounded by

$$2^n \Pr[\mathscr{E}_1(A,B)] \Pr[\mathscr{E}_2(A,B)] \leq 2^n 2^{-2\times 0.7n} \leq 2^{-0.4n}.$$

$\square$

Having shown the result for two random graphs, we apply the union bound to show that for exponentially many random graphs, it is unlikely that any tree can cluster more than one graph in the family well. We now prove Lemma 2.

Proof. Let $\mathscr{F}$ consist of $2^{0.2n}$ graphs generated uniformly at random $\mathscr{P}(n,5)$. For each pair of graphs G_1, G_2, we have by Lemma 4 every balanced cut will miss at least $\frac{\alpha}{5}n$ cycles in either G_1 or G_2 with probability $1-2^{-0.4n}$. By the union bound applied $\frac{1}{2}2^{0.4n}$ times for each pair of graphs, we have with probability $\frac{1}{2}$ that every balanced cut will miss at least $\frac{\alpha}{5}n$ cycles in all but at most one graph in $\mathscr{F}$.

Every tree can be mapped to a balanced cut, so by Lemma 1, any tree will cost at least $\frac{4\alpha}{15}n^2 \geq \frac{n^2}{400}$ on all but at most one member of $\mathscr{F}$. This allows us to conclude that the sets $\mathscr{B}(G,r)$ are disjoint for all $G \in \mathscr{F}$. $\square$

E.3 Omitted proofs from Section 5.5

E.3.1 Proof of Theorem 16

First, we state a theorem about private graph sparsification.

Theorem 16. *There is a polynomial-time, (ε,δ)-edge differentially private algorithm which, on input graph $G=(V,E,w)$, outputs a graph G' which with probability 0.9 is a $(z, O(nz))$-approximation to cut queries in G, where $z = O(\frac{\log^2 \frac{1}{\delta}}{\varepsilon}\frac{\log n}{\sqrt{n}})$.*

Proof. We apply an edge sparsification algorithm of [10], which given a graph with Laplacian L, outputs a graph with Laplacian L' with $O(\frac{n}{\gamma^2})$ edges such that

$$(1-\gamma)((1-z)L+zL_n) \preceq L' \preceq (1+\gamma)((1-z)L+zL_n),$$

where L_n is the Laplacian of an unweighted K_n. The value of the cut $w(S,\overline{S})$ is given by by $\mathbf{1}_S^T L \mathbf{1}_S$; therefore, we have

$$(1-\gamma)((1-z)w(S,\overline{(S)})-z|S|(n-|S|)) \leq w'(S,\overline{S}) \leq (1+\gamma)((1-z)w(S,\overline{S})+z|S|(n-|S|))$$

Using the fact that $|S|(n-|S|) \leq n\min\{|S|, n-|S|\}$ and letting $\gamma \to 0$, we estabish that G' is a (z,nz) approximation to cut queries in G. □

Next, we reduce the cost to a sum of cuts. This idea appeared in [4].

Lemma 5. *Suppose G' is an (α_n, β_n)-approximation to cut queries in G for some $\alpha < 1$. Let T' be any tree which satisfies $\omega_{G'}(T') \leq a_n \omega^*_{G'}$. Then,*

$$\omega_G(T') \leq (1+2\alpha_n)a_n\omega^*_G + (4a_n+2)\beta_n n^2.$$

For the revenue objective, let T' be any tree which satisfies $\omega^{MW}_{G'}(T') \geq a_n \omega^{MW*}_{G'}$. Then,

$$\omega^{MW}_G(T') \geq (1-2\alpha_n)a_n\omega^{MW*}_G - 2(a_n+1)\beta_n n^2 - 2(a_n+1)\alpha_n n^3.$$

A proof of this lemma appears in the next section.

Finally, we are ready to prove the theorem.

Proof. (Of Theorem 16): First, release a private graph G' using Theorem 16, which is a (z,nz)-cut approximation with probability at least 0.9, where $z = O(\frac{\log^2 \frac{1}{\delta}}{\varepsilon}\frac{\log n}{\sqrt{n}})$. We use the black box hierarchical clustering algorithm, which finds a tree such that $\mathbb{E}[\omega_G(T')] \leq a_n \omega^*_G$. Then, we apply Lemma 5, obtaining

$$\mathbb{E}[\omega_G(T')] \leq (1+2z)a_n\omega_G^* + (4a_n+2)zn^3.$$

For the revenue objective, our black box hierarchical clustering finds a tree T' such that $\mathbb{E}[\omega_G^{\mathsf{MW}}(G')] \geq a_n\omega_G^{\mathsf{MW}*}$. We apply Lemma 5, obtaining

$$\omega_G^{\mathsf{MW}}(T') \geq (1-2z)a_n\omega_G^{\mathsf{MW}*} - 4(a_n+1)zn^3.$$

□

E.3.2 Proof of Lemma 5

We start with the well-known representation of $\omega_G(T)$ [57]:

$$\omega_G(T) = \sum_{S\to(S_1,S_2) \text{ in } T} |S|w(S_1,S_2),$$

where the sum is indexed by internal splits of T, which splits a set S of leaves into two parts S_1,S_2. Using the identity $w(S_1,S_2) = \frac{1}{2}w(S_1,\overline{S_1}) + \frac{1}{2}w(S_2,\overline{S_2}) - \frac{1}{2}w(S,\overline{S})$, we substitute:

$$\omega_G(T) = \frac{1}{2}\sum_{S\to(S_1,S_2) \text{ in } T} |S|w(S_1,\overline{S_1}) + |S|w(S_2,\overline{S_2}) - |S|w(S,\overline{S})$$

In the above sum, if we assign cuts to their respective nodes, then we obtain the following: The root node is assigned $-|S|w(S,\overline{S}) = 0$. Each internal node S_1 which is not a leaf node or the root is assigned $|S|w(S_1,\overline{S_1}) - |S_1|w(S_1,\overline{S_1}) = |S_2|w(S_1,\overline{S_1})$, where $S \to (S_1,S_2)$ is the parent split of S_1. Finally, each leaf node S_1 is assigned $|S|w(S_1,\overline{S_1}) = |S_2|w(S_1,\overline{S_1}) + w(S_1,\overline{S_1})$, using the fact that $|S_1| = 1$. This brings us to the following decomposition [4]:

$$\omega_G(T) = \underbrace{\sum_{S \to (S_1, S_2) \text{ in } T} |S_2| w(S_1, \overline{S_1}) + |S_1| w(S_2, \overline{S_2})}_{\omega_G^1(T)} + \underbrace{\sum_{i=1}^{n} w(v, \overline{v})}_{\omega_G^2}.$$

We refer to the leftmost term of the above as $\omega_G^1(T)$, and the rightmost term as ω_G^2. Observe the second quantity does not depend on T. Now, for any tree T, we have

$$\begin{aligned}
\omega_{G'}^1(T) \le \sum_{S \to (S_1, S_2) \text{ in } T} & \Big(|S_2| ((1+\alpha_n) w_G(S_1, \overline{S_1}) + \beta_n \min\{|S_1|, n - |S_1|\}) \\
& + |S_1| ((1+\alpha_n) w_G(S_2, \overline{S_2}) + \beta_n \min\{|S_2|, n - |S_2|\}) \Big) \\
& \le (1+\alpha_n) \omega_G^1(T) + \beta_n \sum_{S \to (S_1, S_2) \text{ in } T} |S_2| \min\{|S_1|, n - |S_1|\} + |S_1| \min\{|S_2|, n - |S_2|\} \\
& \le (1+\alpha_n) \omega_G^1(T) + \beta_n \sum_{S \to (S_1, S_2) \text{ in } T} 2|S_1||S_2| \\
& \le (1+\alpha_n) \omega_G^1(T) + \beta_n n^2,
\end{aligned}$$

where the final line comes from an induction argument: if $f(n) \le \max_{1 \le i \le n} f(i) f(n-i) + 2\beta i(n-i)$, then we can show via induction that $f(n) \le \frac{n^2 \beta}{2}$. By a similar process, we can show the following inequalities

$$(1-\alpha_n)\omega_G^1(T) - \beta_n n^2 \le \omega_{G'}^1(T) \le (1+\alpha_n) w_G^1(T) + \beta_n n^2 \tag{E.2}$$

$$(1-\alpha_n)\omega_G^2 - \beta_n n \le \omega_{G'}^2 \le (1+\alpha_n)\omega_G^2 + \beta_n n \tag{E.3}$$

This implies that

$$(1-\alpha_n)\omega_G(T) - 2\beta_n n^2 \le \omega_{G'}(T) \le (1+\alpha_n)\omega_G(T) + 2\beta_n n^2.$$

This allows us to derive that

$$\begin{aligned}
\omega_G(T') &\leq (1+\alpha_n)\omega_{G'}(T') + 2\beta_n n^2 \\
&\leq (1+\alpha_n)a_n\omega_{G'}(T^*) + 2\beta_n n^2 \\
&\leq (1+\alpha_n)a_n((1+\alpha_n)\omega^*_G + 2\beta_n n^2) + 2\beta_n n^2 \\
&\leq (1+2\alpha_n)a_n\omega^*_G + (4a_n+2)\beta_n n^2
\end{aligned}$$

Plugging T^*, the optimal tree for G, into the above, we obtain that $\omega^*_{G'} \leq (1+\alpha_n)\omega^*_G + 2\beta_n n^2$, and therefore,

$$\omega_{G'}(T') \leq a_n(1+\alpha_n)\omega^*_G + 2a_n\beta_n n^2.$$

We also have that $(1-\alpha_n)\omega_G(T') - 2\beta_n n^2 \leq \omega_{G'}(T')$, and we obtain our result by rearranging.

E.3.3 Proof of Lemma 3

Using a general lemma about the exponential mechanism [145], we are able to prove a bound on the algorithm error.

Lemma 6. *Let $f(X,Y)$ be a function with sensitivity 1 in X. Suppose we run the exponential mechanism $M : \mathscr{X} \to \mathscr{Y}$ with finite range $\mathscr{Y}$ using utility function $u_X(Y) = f(X,Y)$. Let $OPT(X) = \min_{Y\in\mathscr{Y}} u_X(Y)$. If our privacy budget is ε, then for each $X \in \mathscr{X}$, we have*

$$\Pr[u_X(M(X)) \leq OPT(X) + 2\frac{\log(|\mathscr{Y}|)}{\varepsilon}] \geq 1 - \frac{1}{|\mathscr{Y}|}.$$

Proof. Let $\mathscr{Z} = \{Y \in \mathscr{Y} : u_X(Y) \leq OPT(X) + 2\frac{\log(|\mathscr{Y}|)}{\varepsilon}\}$. We are guaranteed that the optimal element, Z^*, with $u_X(Z^*) = OPT(X)$, is in $\mathscr{Z}$. We want to lower bound the quantity $\Pr[M(X) \in \mathscr{Z}]$. Observe that

$$\Pr[M(X) \in \mathscr{Z}] = \frac{\sum_{Z\in\mathscr{Z}} e^{-\varepsilon u_X(Z)/2}}{\sum_{Z\in\mathscr{Z}} e^{-\varepsilon u_X(Z)/2} + \sum_{Y\in\mathscr{Y}, Y\notin\mathscr{Z}} e^{-\varepsilon u_X(Y)/2}}$$

$$\geq \frac{e^{-\varepsilon u_X(Z^*)/2}}{e^{-\varepsilon u_X(Z^*)/2} + \sum_{Y\in\mathscr{Y}, Y\notin\mathscr{Z}} e^{-\varepsilon u_X(Y)/2}}$$
$$= \frac{e^{-\varepsilon OPT(X)/2}}{e^{-\varepsilon OPT(X)/2} + \sum_{Y\in\mathscr{Y}, Y\notin\mathscr{Z}} e^{-\varepsilon u_X(Y)/2}}.$$

The second line holds because the function $g(z) = \frac{z}{z+K}$ for $K > 0$ is decreasing as $z \to 0$. The bottom sum can be upper bounded with $|\mathscr{Y}|e^{-\varepsilon(OPT(X)+2\log(|\mathscr{Y}|)/\varepsilon)/2} \leq \frac{1}{|\mathscr{Y}|}e^{-\varepsilon OPT(X)/2}$. Thus, we are left with

$$\Pr[M(X) \in \mathscr{Z}] \geq \frac{1}{1+1/|\mathscr{Y}|} \geq 1 - \frac{1}{|\mathscr{Y}|}.$$

□

For hierarchical clustering, our algorithm is a corollary of the previous result:

Proof. We apply the exponential mechanism with utility function $u_G(T) = -\frac{1}{n}\omega_G(T)$, which has sensitivity 1. The range of the algorithm is the space of trees with n nodes; there are at most n^n trees of this size. By Lemma 6, the utility satisfies $\Pr[\frac{\omega_G^*}{n} \leq \frac{\omega_G(M(G))}{n} + 2\frac{n\log n}{\varepsilon}] \geq 1 - o(1)$, and hence the algorithm is a $(1, O(\frac{n^2\log n}{\varepsilon}))$-approximation.

For the revenue objective, we apply the exponential mechanism with utility function $u_G(T) = \frac{1}{2n}\omega_G^{\mathsf{MW}}(T)$, which has sensitivity 1. By Lemma 3, the utility satisfies $\Pr[\frac{\omega_G^{\mathsf{MW}}(M(G))}{2n} \leq \frac{\omega_G^{\mathsf{MW}*}}{2n} + 2\frac{n\log n}{\varepsilon}] \geq 1 - o(1)$. This establishes $(1, O(\frac{n^2\log n}{\varepsilon}))$-approximation. □

E.4 Omitted proofs from Section 5.6

E.4.1 Proof of Theorem 13

In order to prove this theorem, we will show that DPHCBlocks finds a $(1+o(1))$-approximate ground-truth tree, and then appeal to a result showing the such trees are approximately optimal with high probability [54]:

Lemma 7. *(Lemma 5.10 from [54]) Let G be a graph drawn from* $\mathrm{HSBM}(B,P,f)$*, where* $p_{min} = \min_{i\in B\cup N} f(i) \geq \omega(\sqrt{\frac{\log n}{n}})$*. Let* (B,P',f') *be a* γ*-approximate ground-truth tree. Then,* *with probability* $1-2^{-n}$*, we have*

$$\text{cost}_G(P') \leq \gamma(1+o(1))\text{cost}^*_G,$$

We now show that DPHCBlocks outputs an approximate ground-truth tree. We introduce a high-probability event and prove that if it happens, then the output is an approximate ground-truth tree.

Our event $\mathscr{E}$ states that $sim(B_i, B_j)$ as used in DPHCBlocks is a good estimate for $f(LCA_P(B_i, B_j))$. Intuitively, this makes sense, as if one had access to $f(LCA_P(B_i, B_j))$, then it would be easy to construct P (or an equivalent tree) using single linkage. Formally, we let $\mathscr{E}$ denote the event that there exists α such that for all B_i, B_j,

$$\left|sim(B_i, B_j) - f(LCA_P(B_i, B_j))\right| \leq \alpha f(LCA_P(B_i, B_j)). \tag{E.4}$$

The following lemma shows that $\mathscr{E}$ occurs with high probability.

Lemma 8. *If* $|B_i| \geq n^{2/3}$ *for all* i, j*,* $\varepsilon \geq \frac{1}{n^{1/2}}$*, and* $f(x) \geq \frac{\log n}{n^{1/2}}$*, then the event* $\mathscr{E}$ *occurs with* $\alpha = \frac{8}{n^{1/6}}$ *with probability at least* $1 - \frac{2}{n}$*.*

Proof. The values $w_G(B_i, B_j)$ are distributed according to $\text{Binomial}(N_{ij}, p_{ij})$, where $N_{ij} = |B_i||B_j|$ and $p_{ij} = f(LCA_P(B_i, B_j))$. By Hoeffding's bound, we have that

$$\Pr[|w_G(B_i, B_j) - p_{ij}N_{ij}| \geq 2\log n\sqrt{N_{ij}}] \leq \frac{1}{n^3}.$$

Furthermore, we have that $\Pr[|\mathscr{L}_{ij}| \geq \frac{6\log n}{\varepsilon}] \leq \frac{1}{n^3}$. Plugging in $sim(B_i, B_j) = \frac{w_G(B_i,B_j)+\mathscr{L}_{ij}}{N_{ij}}$, we obtain

$$\Pr\left[|sim(B_i, B_j) - p_{ij}| \geq \frac{2\log n}{\sqrt{N_{ij}}} + \frac{6\log n}{\varepsilon N_{ij}}\right] \leq \frac{2}{n^3}.$$

Because $N_{ij} \geq n^{4/3}$ and $\varepsilon \geq \frac{1}{n^{1/2}}$, we have $\frac{2\log n}{\sqrt{N_{ij}}} + \frac{6\log n}{\varepsilon N_{ij}} \leq \frac{8\log n}{n^{2/3}} \leq \frac{8}{n^{1/6}} p_{ij}$. Thus, we obtain

$\Pr[|sim(B_i,B_j) - p_{ij}| \geq \alpha p_{ij}] \leq \frac{2}{n^3}$, with $\alpha = \frac{8}{n^{1/6}}$. Taking a union bound over all $\binom{k}{2} \leq n^2$ choices of i, j, we obtain our result. $\square$

Finally, we show that when $\mathscr{E}$ occurs, then DPHCBlocks finds an approximate ground-truth tree. A similar result was proved in [54], though our lemma statement is sufficiently different that we include a proof here.

Lemma 9. *Assume that event $\mathscr{E}$ occurs. Then, the tuple (B,T,f') returned by Algorithm 15 is a $(1+\alpha)$-approximate ground-truth tree for (B,P,f).*

Proof. We want to show that for all $B_i, B_j \in V$, we have

$$(1-\alpha)f(LCA_P(B_i,B_j)) \leq f'(LCA_{P'}(B_i,B_j)) \leq (1+\alpha)f(LCA_P(B_i,B_j)).$$

Let $I = LCA_T(B_i,B_j)$ be the internal node in which B_i,B_j are merged, and let C_i,C_j be the children of I such that $B_i \subseteq C_i$ and $B_j \subseteq C_j$. We have that

$$f'(LCA_{P'}(B_i,B_j)) = sim(C_i,C_j) = \max_{B \in C_i, B' \in C_j} sim(B,B').$$

Thus, it holds that $sim(B_i,B_j) \leq f'(LCA_{P'}(B_i,B_j))$. As event $\mathscr{E}$ holds, we have that $sim(B_i,B_j) \geq (1-\alpha)f(LCA_P(B_i,B_j))$.

To finish, we show that $sim(C_i,C_j) \leq (1+\alpha)f(LCA_P(B_i,B_j)$. Let $J = LCA_P(B_i,B_j)$ be the internal node in which B_i,B_j are merged in P, and let D_i,D_j be the children of J such that $B_i \subseteq D_i$ and $B_j \subseteq D_j$. We consider the following two cases.

Case 1:

$C_i \subseteq D_i$ and $C_j \subseteq D_j$. Then, we have

$$sim(C_i,C_j) \leq \max_{B \in D_i, B' \in D_j} sim(B,B') \leq (1+\alpha) \max_{B \in D_i, B' \in D_j} f(LCA_P(B,B')).$$

As D_i, D_j are nodes of the ground-truth tree, it holds that $f(LCA_P(B,B'))$ is the same for any choice of $B \in D_i, B' \in D_j$. In particular, this is true for $f(LCA_P(B_i,B_j))$.

Case 2:

There exists B_ℓ such that $B_\ell \subseteq C_i$ and $B_\ell \nsubseteq D_i$ (or the same holds for C_i, D_i replaced by C_j, D_j). WLOG, suppose the former case holds. Then, there exists a child N of I whose children are N_L, N_R, such that $N_L \subseteq D_i$ and $N_R \cap D_i = \emptyset$. It then follows that

$$sim(N_L, N_R) \leq (1+\alpha) \max_{B \in N_L, B' \in N_R} f(LCA_P(B,B')) \leq (1+\alpha) f(LCA_P(B_i,B_j)),$$

where the second inequality holds because f is decreasing as we ascend P. However, we also have that $sim(N_L,N_R) \geq sim(C_i,C_j)$, as sim also obeys this property (if the last inequality did not hold, then N_L, N_R would not have been merged). This finishes the last case. $\square$

The proof follows by applying Lemma 8 and then Lemma 9.

E.4.2 Proof of Theorem 14

Overview

When running $\mathsf{DPCommunity}$, fix Y, Z_1, Z_2, and let $(\hat{A}_1, \hat{A}_2)$ and $(\hat{A}'_1, \hat{A}'_2)$ be the splits of $\hat{A}$ and an adjacent database $\hat{A}'$. We will view the matrix $F = P(\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2))$ as a vector, and then show that releasing F plus appropriate Gaussian noise satisfies privacy via the Gaussian mechanism. Our proof will bound the L_2 sensitivity of F, given by

$$\Delta_2(F) = \|P(\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2)) - P(\Pi^{(k)}_{\hat{A}'_1}(\hat{A}'_2))\|_F,$$

in terms of the quantity $\Gamma = \frac{\sigma_1(\hat{A}_2)}{\sigma_k(\hat{A}_1) - \sigma_k(\hat{A}_2)}$. Recall that P is a random $m \times \frac{n}{2}$ projection matrix. To control this sensitivity, we will need the fact that P preserves the distances in A via the Johnson-Lindenstrauss projection theorem:

Theorem 17. *(Johnson-Lindenstrauss projection theorem [108]): Let* $0 \leq \alpha < \frac{1}{2}$ *and* $0 \leq \beta \leq 1$*, and* $m = 8\frac{\ln\frac{2}{\beta}}{\alpha^2}$*. If* $x \in \mathbb{R}^n$ *is a vector and* $P \sim \mathcal{N}(0, \frac{1}{\sqrt{m}})^{m\times n}$ *is a random matrix then with probability* $1-\beta$*, we have*

$$(1-\alpha)\|x\|_2 \leq \|Px\|_2 \leq (1+\alpha)\|x\|_2$$

We use the above theorem to show that the matrix P does not increase the sensitivity $\Delta(F)$ with high probability.

Lemma 10. *Let* $0 \leq \delta < 1$ *and* $m = 64\ln\frac{2n}{\delta}$*. Then, if* $P \sim \mathcal{N}(0, \frac{1}{\sqrt{m}})^{m\times n/2}$ *the following holds with probability at least* $1-\frac{\delta}{4}$*:*

$$\Delta(F) \leq \frac{3}{2}\|\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2) - \Pi^{(k)}_{\hat{A}'_1}(\hat{A}'_2)\|_F.$$

Proof. Let the columns of $\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2)$ be $\{a_1,\ldots,a_{n/4}\}$ and the columns of $\Pi^{(k)}_{\hat{A}'_1}(\hat{A}'_2)$ be $\{a'_1,\ldots,a_{n'/4}\}$. By the union bound, Theorem 17 with $\alpha = \frac{1}{2}$ and $\beta = \frac{\delta}{n}$ applies to all vectors $a_i - a'_i$ with probability at least $1-\frac{\delta}{4}$. Thus, we have

$$\Delta_2(F)^2 = \sum_{i=1}^{n/4}\|P(a_i)-P(a'_i)\|_2^2 \leq (1+\alpha)^2\sum_{i=1}^{n/4}\|a_i-a'_i\|_2^2 = (1+\alpha)^2\|\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2) - \Pi^{(k)}_{\hat{A}'_1}(\hat{A}'_2)\|_F^2.$$

The result follows. □

Finally, we need a bound on the stability of the projection $\Pi^{(k)}_{\hat{A}_1}$ when $\hat{A}_1$ is perturbed. This is the result of the Davis-Kahan Theorem [27].

Theorem 18. *Let* $\hat{A}_1, \hat{A}'_1$ *be matrices where* $d_k = \sigma_k(\hat{A}_1) - \sigma_{k+1}(\hat{A}_1) > 0$*. Then,*

$$\|\Pi^{(k)}_{\hat{A}_1} - \Pi^{(k)}_{\hat{A}'_1}\|_F \leq \frac{\|\hat{A}_1 - \hat{A}'_1\|_F}{d_k}.$$

Furthermore, the above holds replacing $\|\cdot\|_F$ *with* $\|\cdot\|_2$*.*

Having bounded the L_2-sensitivity, we finally use the well-known Gaussian mechanism [73]

Theorem 19. *If $x \in \mathbb{R}^m$ has L_2 sensitivity at most S, then releasing $x+N$, where $N \sim \frac{S}{\varepsilon}\sqrt{2\ln\frac{1.25}{\delta}}\mathcal{N}(0,1)^m$ satisfies (ε,δ)-DP.*

Proof

Let $\hat{A}$ and $\hat{A}'$ be two adjacent inputs, and consider two runs of DPCommunity with fixed Y, Z_1, Z_2, and P; we will show that the outputs satisfy (ε,δ)-DP. Let $\hat{A}_1'$ and $\hat{A}_2'$ be the values of $\hat{A}_1$ and $\hat{A}_2$ when $\hat{A}'$ is used instead of $\hat{A}$. DPCommunity can be viewed as a post-processing of the private release of values $d_k = \sigma_k(\hat{A}_1) - \sigma_{k+1}(\hat{A}_1)$, $\sigma_1(\hat{A}_2)$, and F; thus, we will show that releasing each of these values satisfies privacy.

Using Lindskii's inequality [27], each rank i singular value of $\hat{A}_1, \hat{A}_2$ can only change by 1 when $\hat{A}$ is changed to $\hat{A}'$. Thus, the sensitivity of d_k is 2, of σ_1 is 1, and thus the release of $\tilde{d}_k = d_k + \frac{8}{\varepsilon}\ln\frac{4}{\delta} + Lap(\frac{8}{\varepsilon})$ and $\tilde{\sigma}_1 = \sigma_1 + \frac{4}{\varepsilon}\ln\frac{4}{\delta} + Lap(\frac{4}{\varepsilon})$ both satisfy $(\frac{\varepsilon}{4}, 0)$-DP. Thus, we will show that releasing $\tilde{F}$ satisfies $(\frac{\varepsilon}{2},\delta)$-DP, and privacy will follow by composition.

By Lemma 10 with probability at least $1-\frac{\delta}{4}$, we have

$$\Delta_2(F) \leq \tfrac{3}{2}\|\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2) - \Pi^{(k)}_{\hat{A}_1'}(\hat{A}_2')\|_F$$

We have either $\hat{A}_1 = \hat{A}_1'$ or $\hat{A}_2 = \hat{A}_2'$. We analyze the cases separately.

Case $\hat{A}_1 = \hat{A}_1'$:

Then, $\hat{A}_2$ and $\hat{A}_2'$ differ in one bit, so $\hat{A}_2 = \hat{A}_2' + E$, where E is a matrix that is ± 1 in one entry and 0 everywhere else. Then,

$$\tfrac{3}{2}\|\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2) - \Pi^{(k)}_{\hat{A}_1}(\hat{A}_2')\|_F = \tfrac{3}{2}\|\Pi^{(k)}_{\hat{A}_1}(E)\|_F \leq \tfrac{3}{2}\|E\|_F \leq \tfrac{3}{2},$$

where the inequality holds because projecting vectors onto a subspace cannot increase their magnitude.

Case $\hat{A}_2 = \hat{A}_2'$:

Then, $\hat{A}_1$ and $\hat{A}_1'$ differ in one bit, so $\|\hat{A}_1 - \hat{A}_1'\|_F \leq 1$. We have

$$\|\Pi_{\hat{A}_1}^{(k)}(\hat{A}_2) - \Pi_{\hat{A}_1'}^{(k)}(\hat{A}_2')\|_F \leq 2k\|(\Pi_{\hat{A}_1}^{(k)} - \Pi_{\hat{A}_1'}^{(k)})(\hat{A}_2)\|_2 \leq 2k\|\Pi_{\hat{A}_1}^{(k)} - \Pi_{\hat{A}_1'}^{(k)}\|_2\|\hat{A}_2\|_2,$$

where the first inequality holds because each term has rank at most k, so the entire quantity has rank at most $2k$, and the second holds by sub-multiplicativity of $\|\cdot\|_2$. By Theorem 18, we have $\|\Pi_{\hat{A}_1}^{(k)} - \Pi_{\hat{A}_1'}^{(k)}\|_2 \leq \frac{1}{d_k}$. Thus, we have

$$\frac{3}{2}\|\Pi_{\hat{A}_1}^{(k)}(\hat{A}_2) - \Pi_{\hat{A}_1'}^{(k)}(\hat{A}_2')\|_F \leq \frac{3k\|\hat{A}_2\|_2}{d_k} = 3k\Gamma.$$

By concentration of Laplace variables, we have $\tilde{d}_k \leq d_k$ and $\tilde{\sigma}_1 \geq \sigma_1$, so $\Gamma \leq \frac{\tilde{\sigma}_1}{\tilde{d}_k} = \tilde{\Gamma}$ with probability at least $1 - \frac{\delta}{2}$. Thus, the sensitivity $\Delta(F)$ is at most $3k\tilde{\Gamma}$, and $(\frac{\varepsilon}{2}, \frac{\delta}{4})$-DP follows via Theorem 19. Factoring in the aformentioned failure probabilities, the entire release of $\tilde{F}$ satisfies $(\frac{\varepsilon}{2}, \delta)$-DP.

E.4.3 Proof of Corollary 15

Overview

Recall that DPCommunity sees a matrix $\hat{A}$ drawn from $\mathrm{HSBM}(B,P,f)$, with expectation matrix A. We define $\tau^2 = \max f(x)$, $s = \min_{i=1}^{k} |B_i|$, and $\Delta = \min_{u \in B_i, v \in B_j, i \neq j} \|A_u - A_v\|_2$. We will show that DPCommunity approximates $\Pi_{\hat{A}_1}^{(k)}(\hat{A}_2)$, which is guaranteed to cluster the original communities via the following result [199]. We let the columns of $\Pi_{\hat{A}_1}^{(k)}(\hat{A}_2)$, which is indexed by the set Z_2, be $\{b_i : i \in Z_2\}$.

Theorem 20. *([199]): There exists a universal constant C such that if $\tau^2 \geq C\frac{\log n}{n}$, $s \geq C \log n$, and $k < n^{1/4}$. $\Delta > C(\tau\sqrt{\frac{n}{s}} + \tau\sqrt{k \log n} + \frac{\tau\sqrt{nk}}{\sigma_k(A)})$, with probability at least $1 - n^{-1}$, then the* *columns $\{b_i : i \in Z_2\}$ in $\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2)$ satisfy:*

$$\|b_i - b_j\|_2 \le \frac{\Delta}{4} \quad \textit{if } \exists u.\ i \in B_u, j \in B_u \textit{ (i.e. } i, j \textit{ are in the same community)}$$

$$\|b_i - b_j\|_2 \ge \Delta \quad \textit{otherwise.}$$

Thus, the clusters in $\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2)$ cluster the original communities assuming Δ is large enough. We will show that $\tilde{F}$ clusters the original communities assuming some condition on Δ. Since DPCommunity returns $\tilde{F} = P(\Pi^{(k)}_{\hat{A}_1}(\hat{A}_2)) + N$, where N is Gaussian noise, our proof involves showing that the distances in $\tilde{F}$ approximate those in $\Pi^{(k)}_{\hat{A}_1}$ using the Johnson-Lindenstrauss lemma and concentration of the Gaussian noise.

We formally restate Theorem 15:

Theorem 21. *Let $\hat{A}$ be drawn from $\mathrm{HSBM}(B,P,f)$. There is a universal constant $C > 2000$ such that if $\tau^2 \ge C\frac{\log n}{n}$, $s \ge C\sqrt{n \log n}$, $k < n^{1/4}$, $\delta < \frac{1}{n}$, $\sigma_k(A) \ge C\max\{\tau\sqrt{n}, \frac{1}{\varepsilon}\ln\frac{4}{\delta}\}$, and*

$$\Delta > C\max\left\{\frac{k(\ln\frac{1}{\delta})^{3/2}}{\varepsilon}\frac{\sigma_1(A)}{\sigma_k(A)}, \tau\sqrt{\frac{n}{s}} + \tau\sqrt{k\log n} + \frac{\tau\sqrt{nk}}{\sigma_k}\right\},$$

then with probability at least $1 - 3n^{-1}$, DPCommunity returns a set of points $\tilde{F} = \{f_i : i \in Z_2\}$ such that

$$\|f_i - f_j\|_2 \le \frac{2\Delta}{5} \quad \textit{if } \exists u.\ i, j \in B_u$$
$$\|f_i - f_j\|_2 \ge \frac{4\Delta}{5} \quad \textit{otherwise,}$$

and thus the clusters in $\tilde{F}$ indicate the communities.

Proof of Theorem 21

Let the columns of $\tilde{F}$ be $\{f_i : i \in Z_2\}$. We have $f_i = P(b_i) + n_i$, where $n_i \sim \frac{3k\tilde{\Gamma}}{\varepsilon}\sqrt{2\ln\frac{5}{\delta}}N(0,1)^m$. By concentration bounds, we have with probability $1 - \frac{1}{n}$ that each n_i

satisfies

$$\|n_i\|_2 \leq \frac{3k\tilde{\Gamma}}{\varepsilon}\sqrt{2\ln\frac{5}{\delta}}\sqrt{2m\ln n} \triangleq K.$$

Next, applying Theorem 17 on the vectors $b_i - b_j$ for $i, j \in Z_2$, we have $0.9\|b_i - b_j\|_2 \leq \|P(b_i) - P(b_j)\|_2 \leq 1.1\|b_i - b_j\|_2$ with probability $1-\delta > 1-\frac{1}{n}$. Thus, if $\exists u.\ i \in B_u, j \in B_u$, then

$$\begin{aligned}
\|f_i - f_j\|_2 &\leq \|P(b_i) - P(b_j)\|_2 + \|n_i\|_2 + \|n_j\|_2 \\
&\leq 1.1\|b_i - b_j\|_2 + 2K \\
&\leq 0.275\Delta + 2K,
\end{aligned}$$

Otherwise, we have

$$\begin{aligned}
\|f_i - f_j\|_2 &\geq \|P(b_i) - P(b_j)\|_2 - \|n_i\|_2 - \|n_j\|_2 \\
&\geq 0.9\|b_i - b_j\|_2 - 2K \\
&\geq 0.9\Delta - 2K.
\end{aligned}$$

Finally, we show that K can be upper bounded by the singular values of the expectation matrix A. This can be done with the following two lemmas which are proven implicitly in [199].

Lemma 11. *Let A be an $m \times n$ (with $m \geq n$) matrix of expectations in $[0,1]$, and let $\hat{A}$ be a randomized rounding of A to $\{0,1\}$. Then, with probability at least $1-\frac{1}{n}$, we have for all $1 \leq i \leq m$, $|\sigma_i(A) - \sigma_i(\hat{A})| \leq 4\tau\sqrt{n} + 4\log n$, where τ^2 is the maximum probability in A.*

Proof. Each $\sigma_{i+1}(A)$ is equal to $\max_{\mathrm{rank}(A_i)=i}\|A - A_i\|_2$. Let $A_i^*, \hat{A}_i^*$ be rank i matrices such that $\sigma_{i+1}(A) = \|A - A_i^*\|_2$ and $\sigma_{i+1}(\hat{A}) = \|\hat{A} - \hat{A}_i^*\|_2$. We have that $\sigma_{i+1}(A) \leq \|A - \hat{A}_i^*\|_2 \leq \|\hat{A} - \hat{A}_i^*\|_2 + \|A - \hat{A}\|_2$.

Thus, it remains to bound $\|A - \hat{A}\|_2$. Let the columns in $A - \hat{A}$ be $a_1, \ldots, a_n$. Using Lemma 7 from [200], we have that with probability at least $1 - \frac{1}{n^3}$, the length of the projection of a_i onto a basis vector e_i is at most $4(\tau + \frac{\log n}{\sqrt{n}})$. Thus, the total length $\|Ae_i\|_2$ is at most $4(\tau + \frac{\log n}{\sqrt{n}}$, and thus $\|A\|_2 \le \sqrt{n}4(\tau + \frac{\log n}{\sqrt{n}})$ establishing that $\sigma_{i+1}(A) \le \sigma_{i+1}(\hat{A}) + 4\sqrt{n}\tau + 4\log n$. Likewise, we can show that $\sigma_{i+1}(A) \ge \sigma_{i+1}(\hat{A}) - 4\sqrt{n}\tau - 4\log n$. $\square$

Lemma 12. *Let A be an expectation matrix of* $\mathrm{HSBM}(B,P,f)$ *with k blocks with minimum block size $s \ge 16\sqrt{n\log n}$, and let C be the submatrix of A with rows Y and columns Z, where $|Y| = \frac{n}{2}$ and $|Z| = \frac{n}{4}$ are chosen randomly from $[n]$ such that $Y \cap Z = \emptyset$. Then, with probability at least $1 - \frac{1}{n}$, for all $1 \le i \le k$, we have*

$$(\tfrac{1}{8} - \tfrac{\sqrt{n\log n}}{s})\sigma_i(A_1) \le \sigma_i(A_1) \le (\tfrac{1}{8} + \tfrac{\sqrt{n\log n}}{s})\sigma_i(A_1)$$

Proof. Observe that the blocks in C are indexed in rows by $B_1 \cap Y, \ldots, B_k \cap Y$ and in columns by $B_1 \cap Z, \ldots, B_k \cap Z$. By Chernoff's bound, with probability at least $1 - \frac{1}{n^2}$, we have for all i that

$$\frac{1}{2} - \frac{\sqrt{n\log n}}{|B_i|} \le \frac{|B_i \cap Y|}{|B_i|} \le \frac{1}{2} + \frac{\sqrt{n\log n}}{|B_i|} \qquad \frac{1}{4} - \frac{\sqrt{n\log n}}{|B_i|} \le \frac{|B_i \cap Z|}{|B_i|} \le \frac{1}{4} + \frac{\sqrt{n\log n}}{|B_i|}.$$

We have $\sigma_k(A) = \min_{\mathrm{rank}(A_{k-1})=k-1} \|A - A_{k-1}\|_F$ and $\sigma_k(C) = \min_{\mathrm{rank}(C_{k-1})=k-1} \|C - C_{k-1}\|_F$; let A^*_{k-1} and C^*_{k-1} be the maximizers of the previous expressions. Let A' denote the matrix C^*_{k-1} with rows and columns duplicated such that each element $(A')_{ij}$ is equal to $(C^*_{k-1})_{xy}$, where x, y are any two points in the same block as i, j, respectively. Accounting for the duplication factors of each block, we have

$$\left(\frac{1}{2} - \frac{\sqrt{n\log n}}{s}\right)\left(\frac{1}{4} - \frac{\sqrt{n\log n}}{s}\right)\|A - A'\|_F \le \|C - C^*_{k-1}\|_F,$$

and thus we see that $(\frac{1}{8} - \frac{\sqrt{n\log n}}{s})\sigma_k(A) \le \sigma_k(A_1)$. By a similar sampling argument, we can show that $(\frac{1}{8} + \frac{\sqrt{n\log n}}{s})\sigma_k(A) \ge \sigma_k(A_1)$. Repeating the argument for $\sqrt{\sigma_i(A)^2 + \cdots \sigma_k(A)^2} = \min_{\mathrm{rank}(A_{i-1})=i-1} \|A - A_{i-1}\|_F$, we obtain the result for all $1 \le i \le k$. $\square$

Let A_1, A_2 be the expectation matrices of $\hat{A}_1, \hat{A}_2$ for fixed Y, Z_2. Using Lemmas 11 and 12, we have that $\sigma_1(\hat{A}_2) \le \sigma_1(A_2) + 4\tau\sqrt{n} + 4\log n \le (\frac{1}{8} + \frac{\sqrt{n\log n}}{s})\sigma_1(A) + 4\tau\sqrt{n} + 4\log n \le$

$\frac{3}{32}\sigma_1(A) + 4\tau\sqrt{n} + 4\log n$. Applying these again, we obtain

$$\begin{aligned}
d_k(\hat{A}_1) &= \sigma_k(\hat{A}_1) - \sigma_{k+1}(\hat{A}_1) \\
&\geq \sigma_k(A_1) - \sigma_{k+1}(A_1) - 8\tau\sqrt{n} - 8\log n \\
&\geq (\frac{1}{8} - \frac{\sqrt{n\log n}}{s})(\sigma_k(A) - \sigma_{k+1}(A)) - 8\tau\sqrt{n} - 8\log n \\
&\geq \frac{1}{16}\sigma_k(A) - 8\tau\sqrt{n} - 8\log n
\end{aligned}$$

Finally, we have $\tilde{\Gamma} = \frac{\tilde{\sigma}_1(\hat{A}_2)}{\tilde{d}_k(\hat{A}_1)}$, which with probability at least δ, will satisfy

$$\tilde{\Gamma} \leq \frac{\sigma_1(\hat{A}_2) + \frac{8}{\varepsilon}\ln\frac{4}{\delta}}{d_k(\hat{A}_1) - \frac{16}{\varepsilon}\ln\frac{4}{\delta}} \leq \frac{\frac{3}{32}\sigma_1(A) + 4\tau\sqrt{n} + 4\log n + \frac{8}{\varepsilon}\ln\frac{4}{\delta}}{\frac{1}{16}d_k(A) - 8\tau\sqrt{n} - 8\log n - \frac{16}{\varepsilon}\ln\frac{4}{\delta}}.$$

By our assumption that $\sigma_k(A) \geq 1024\max\{\tau\sqrt{n}, \frac{1}{\varepsilon}\ln\frac{4}{\delta}\}$, we obtain that $\hat{\Gamma} \leq 4\frac{\sigma_1(A)}{\sigma_k(A)}$, This implies that

$$K \leq \frac{12k\sqrt{m\ln\frac{5}{\delta}\ln n}}{\varepsilon}\frac{\sigma_1(A)}{\sigma_k(A)} = \frac{48k\sqrt{2\ln\frac{2n}{\delta}\ln\frac{5}{\delta}\ln n}}{\varepsilon}\frac{\sigma_1(A)}{\sigma_k(A)} \leq \frac{96k(\ln\frac{5}{\delta})^{3/2}}{\varepsilon}\frac{\sigma_1(A)}{\sigma_k(A)},$$

where the last step follows because $\delta < \frac{1}{n}$. From our assumption, we have $2K \leq 0.1\Delta$, and the result follows.

E.4.4 Proof of Corollary 1

In this special case, we can write $A = P \otimes \mathbf{1}_B$, where P is a $k \times k$ matrix with p on the diagonal and q everywhere else, $\mathbf{1}_s$ is a $s \times s$ matrix consisting of all 1s, and $\otimes$ denotes the Kronecker product. It is easy to see that the eigenvalues of P are $\{p + q(k-1), p-q, \ldots, p-q\}$, and the eigenvalues of $\mathbf{1}_s$ are $\{s, 0, \ldots, 0\}$. The eigenvalues of A are the product of the two sets of eigenvalues of P and $\mathbf{1}_s$. Thus, the top k largest eigenvalues are $s(p + q(k-1))$ and then $k-1$ copies of $s(p-q)$.

Thus, the following properties of A hold: (1) $\sigma_1 = s(p + q(k-1))] \leq sk(p+q)$, (2)

$\sigma_k = s(p-q)$, (3) $\tau = \sqrt{p}$, and (4) $\Delta = (p-q)\sqrt{s}$. We are able to apply Theorem 21 when

$$
\begin{aligned}
(p-q)\sqrt{s} &\geq \frac{s(p+q)}{s(p-q)} \frac{Ck(\log \frac{1}{\delta})^{3/2}}{\varepsilon} \\
\frac{(p-q)^2}{p+q} &\geq \frac{C(k \log \frac{1}{\delta})^{3/2}}{\sqrt{n}}.
\end{aligned}
$$

This establishes the result.

Bibliography

[1] 12th Annual Graph Drawing Contest. http://mozart.diei.unipg.it/gdcontest/contest2005/index.html, 2005.

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

[3] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS'19)*, pages 1120–1129, 2019.

[4] Arpit Agarwal, Sanjeev Khanna, Huan Li, and Prathamesh Patil. Sublinear algorithms for hierarchical clustering. *arXiv preprint arXiv:2206.07633*, 2022.

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

[6] Maryam Aliakbarpour, Amartya Shankha Biswas, Themistoklis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.

[7] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA'21)*, pages 522–539, 2021.

[8] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic randomized response techniques, 2003.

[9] G. Andrew, O. Thakkar, H. B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. In *Proc. NeurIPS'21*, pages 1–12, 2021.

[10] Raman Arora and Jalaj Upadhyay. On differentially private graph sparsification and applications. *Advances in neural information processing systems*, 32, 2019.

[11] Udit Arora, Hridoy Sankar Dutta, Brihi Joshi, Aditya Chetan, and Tanmoy Chakraborty. Analyzing and detecting collusive users involved in blackmarket retweeting activities. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–24, 2020.

[12] R. Arratia and L. Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of Mathematical Biology*, 51(1):125–131, 1989.

[13] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *arXiv:1807.00459*, 2018.

[14] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional euclidean spaces. In *International Conference on Machine Learning*, pages 322–331. PMLR, 2017.

[15] Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Proceedings of the 39th Annual Cryptology Conference on Advances in Cryptology (CRYPTO'19)*, pages 638–667, 2019.

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

[17] Rina Foygel Barber and John C. Duchi. Privacy and statistical risk: Formalisms and minimax bounds. *CoRR*, 1412.4451:1–29, 2014.

[18] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17)*, pages 2285—-2293, 2017.

[19] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the 47th annual ACM Symposium on Theory of Computing (STOC'15)*, pages 127–135, 2015.

[20] Mohammadhossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6864–6874. Curran Associates, Inc., 2017.

[21] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology*, CRYPTO 2008, pages 451–468, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] Adrien Benamira, Benjamin Devillers, Etienne Lesot, Ayush K Ray, Manal Saadi, and Fragkiskos D Malliaros. Semi-supervised learning and graph neural networks for fake news detection. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 568–569. IEEE, 2019.

[23] S. K. Bera and A. Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In *Proc. STACS'17*, pages 11:1–11:14, 2017.

[24] S. K. Bera and C. Seshadhri. How the degeneracy helps for triangle counting in graph streams. In *Proc. PODS'20*, pages 457–467, 2020.

[25] S. K. Bera and C. Seshadhri. How to count triangles, without seeing the whole graph. In *Proc. KDD'20*, pages 306–316, 2020.

[26] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *Proceedings of the International Conference on Machine Learning*, pages 634–643, 2019.

[27] Rajendra Bhatia. *Matrix Analysis*, volume 169. Springer Verlag, 1997.

[28] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the International Coference on International Conference on Machine Learning*, pages 1467–1474, 2012.

[29] Vincent Bindschaedler and Reza Shokri. Synthesizing plausible privacy-preserving location traces. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (S&P'16)*, pages 546–563, 2016.

[30] Andreas Björklund, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Approximate counting of k-paths: Deterministic and in polynomial space. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP'19)*, pages 24:1–24:15, 2019.

[31] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 410–419. IEEE, 2012.

[32] Mark Bun, Marek Elias, and Janardhan Kulkarni. Differentially private correlation clustering. In *International Conference on Machine Learning*, pages 1136–1146. PMLR, 2021.

[33] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Attestable robustness for secure federated learning. In *arXiv:2107.03311*, 2021.

[34] X. Cao, J. Jia, and N. Z. Gong. Data poisoning attacks to local differential privacy protocols. In *Proc. Usenix Security'21*, pages 947–964, 2021.

[35] R. Chan. The cambridge analytica whistleblower explains how the firm used facebook data to sway elections. https://www.businessinsider.com/cambridge-analytica-whistleblower-christopher-wylie-facebook-data-2019-10, 2019.

[36] T-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *Proceedings of the 20th Annual European Conference on Algorithms*, ESA'12, pages 277–288, Berlin, Heidelberg, 2012. Springer-Verlag.

[37] Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. SIAM, 2017.

[38] Vaggos Chatziafratis, Grigory Yaroslavtsev, Euiwoong Lee, Konstantin Makarychev, Sara Ahmadian, Alessandro Epasto, and Mohammad Mahdian. Bisect and conquer: Hierarchical clustering via max-uncut bisection. In *International Conference on Artificial Intelligence and Statistics*, pages 3121–3132. PMLR, 2020.

[39] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[40] Hongjie Chen, Vincent Cohen-Addad, Tommaso d'Orsi, Alessandro Epasto, Jacob Imola, David Steurer, and Stefan Tiegel. Private estimation algorithms for stochastic block models and mixture models. *arXiv preprint arXiv:2301.04822*, 2023.

[41] Rui Chen, Gergely Acs, and Claude Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS'12)*, pages 638–649, 2012.

[42] Shixi Chen and Shuigeng Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 International Conference on Management of Data (SIGMOD'13)*, pages 653–664, 2013.

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

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

[45] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. In *arXiv:1712.05526*, 2017.

[46] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 883–900, 2021.

[47] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'19)*, pages 375–403, 2019.

[48] Albert Cheu and Maxim Zhilyaev. Differentially private histograms in the shuffle model from fake users. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 440–457, 2022.

[49] S. Chu and J. Cheng. Triangle listing in massive networks and its applications. In *Proc. KDD'11*, pages 672–680, 2020.

[50] Vincent Cohen-Addad, Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Andres Munoz, David Saulpic, Chris Schwiegelshohn, and Sergei Vassilvitskii. Scalable differentially private clustering via hierarchically separated trees. *arXiv preprint arXiv:2206.08646*, 2022.

[51] Vincent Cohen-Addad, Alessandro Epasto, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. Near-optimal private and scalable k-clustering. In *Advances in Neural Information Processing Systems*, 2022.

[52] Vincent Cohen-Addad, Chenglin Fan, Silvio Lattanzi, Slobodan Mitrović, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Near-optimal correlation clustering with privacy. *arXiv preprint arXiv:2203.01440*, 2022.

[53] Vincent Cohen-Addad, Varun Kanade, and Frederik Mallmann-Trenn. Hierarchical clustering beyond the worst-case. *Advances in Neural Information Processing Systems*, 30, 2017.

[54] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, 66(4):1–42, 2019.

[55] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1655–1658, 2018.

[56] E. Cyffers and A. Bellet. Privacy amplification by decentralization. *CoRR*, 2012.05326, 2021.

[57] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127, 2016.

[58] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of data (SIGMOD'16)*, pages 123–138, 2016.

[59] Laxman Dhulipala, David Eisenstat, Jakub Łacki, Vahab Mirronki, and Jessica Shi. Hierarchical agglomerative graph clustering in poly-logarithmic depth. In *Neurips 2022*, 2022.

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

[61] Ibai Diez, Paolo Bonifazi, Iñaki Escudero, Beatriz Mateos, Miguel A Muñoz, Sebastiano Stramaglia, and Jesus M Cortes. A novel brain partition highlights the modular skeleton shared by structure and function. *Scientific reports*, 5:10532, 2015.

[62] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17)*, pages 3574–3583, 2017.

[63] Jingqiu Ding, Tommaso d'Orsi, Rajai Nasser, and David Steurer. Robust recovery for stochastic block models. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 387–394. IEEE, 2022.

[64] X. Ding, S. Sheng, H. Zhou, X. Zhang, Z. Bao, P. Zhou, and H. Jin. Differentially private triangle counting in large graphs. *IEEE Transactions on Knowledge and Data Engineering (Early Access)*, pages 1–14, 2021.

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

[66] 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.

[67] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *Proceedings of the IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS'13)*, pages 429–438, 2013.

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

[69] Hridoy Sankar Dutta and Tanmoy Chakraborty. Blackmarket-driven collusion on online media: a survey. *ACM/IMS Transactions on Data Science (TDS)*, 2(4):1–37, 2022.

[70] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming (ICALP'06)*, pages 1–12, 2006.

[71] Cynthia Dwork. Differential privacy and the us census. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, pages 1–1, 2019.

[72] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

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

[74] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, August 2014.

[75] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2014.

[76] T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. In *Proc. FOCS'15*, pages 614–633, 2015.

[77] Talya Eden, Quanquan C Liu, Sofya Raskhodnikova, and Adam Smith. Triangle counting with local edge differential privacy. *arXiv preprint arXiv:2305.02263*, 2023.

[78] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[79] Marek Eliáš, Michael Kapralov, Janardhan Kulkarni, and Yin Tat Lee. Differentially private release of synthetic graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 560–578. SIAM, 2020.

[80] Alessandro Epasto, Vahab Mirrokni, Bryan Perozzi, Anton Tsitsulin, and Peilin Zhong. Differentially private graph learning via sensitivity-bounded personalized pagerank. In *Neurips*, 2022.

[81] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[82] Ulfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, and Kunal Talwar. Amplification by shuffling: from local to central differential privacy via anonymity. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pages 2468–2479, 2019.

[83] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[84] 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.

[85] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

[86] 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.

[87] Yingjie Fei and Yudong Chen. Achieving the Bayes error rate in synchronization and block models by SDP, robustly. *IEEE Trans. Inform. Theory*, 66(6):3929–3953, 2020.

[88] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *Proceedings of the 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS'21)*, pages 954–964, 2021.

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

[90] Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6):bbab159, 2021.

[91] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. *Advances in Neural Information Processing Systems*, 33:4040–4054, 2020.

[92] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS'21)*, 2021.

[93] Antonious M. Girgis, Deepesh Data, Suhas Diggavi, Ananda Theertha Suresh, and Peter Kairouz. On the rényi differential privacy of the shuffle model. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS'21)*, pages 2321–2341, 2021.

[94] Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

[95] Olivier Guédon and Roman Vershynin. Community detection in sparse networks via Grothendieck's inequality. *Probab. Theory Related Fields*, 165(3-4):1025–1049, 2016.

[96] 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.

[97] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 705–714, 2010.

[98] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (ICDM'09)*, pages 169–178, 2009.

[99] Aditya Hegde, Helen Möllering, Thomas Schneider, and Hossein Yalame. Sok: Efficient privacy-preserving clustering. *Proceedings on Privacy Enhancing Technologies*, 2021(4):225–248, 2021.

[100] Maria Henriquez. The top data breaches of 2021. https://fortune.com/2021/10/06/data-breach-2021-2020-total-hacks/, 2021.

[101] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Psysical Review E*, 65(2):1–4, 2002.

[102] J. Imola, T. Murakami, and K. Chaudhuri. Locally differentially private analysis of graph statistics. In *Proc. USENIX Security'21*, pages 983–1000, 2021.

[103] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 983–1000, 2021.

[104] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Communication-efficient triangle counting under local differential privacy. In *Proceedings of the 31th USENIX Security Symposium (USENIX Security'22)*, 2022.

[105] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Communication-Efficient Triangle Counting under Local Differential Privacy. *arXiv:2110.06485 [cs]*, January 2022. arXiv: 2110.06485.

[106] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[107] N Jardine and R Sibson. A model for taxonomy. *Mathematical Biosciences*, 2(3-4):465–482, 1968.

[108] William B Johnson. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[109] Z. Jorgensen, T. Yu, and G. Cormode. Publishing attributed social graphs with formal privacy guarantees. In *Proc.SIGMOD'16*, pages 107–122, 2016.

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

[111] 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.

[112] Matthew Joseph, Jieming Mao, and Aaron Roth. Exponential separations in local differential privacy. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 515–527, 2020.

[113] P. Kairouz, H. B. McMahan, and B. Avent *et al.* Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, 2021.

[114] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*, pages 2436–2444, 2016.

[115] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G.L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans,

Josh Gardner, Zachary Garrett, Adria Gascon, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecny, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Ozgur, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramer, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. In *arXiv:1912.04977*, 2019.

[116] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, pages 1376–1385, 2015.

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

[118] J. Kallaugher, A. McGregor, E. Price, and S. Vorotnikova. The complexity of counting cycles in the adjacency list streaming model. In *Proc. PODS'19*, pages 119–133, 2019.

[119] Alexander P. Kartun-Giles and Sunwoo Kim. Counting k-hop paths in the random connection model. *IEEE Transactions on Wireless Communications*, 17(5):3201–3210, 2018.

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

[121] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, and Sofya Raskhodnikova. What can we learn privately? In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS'08)*, pages 531–540, 2008.

[122] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Proceedings of the 10th theory of cryptography conference on Theory of Cryptography (TCC'13)*, pages 457–476, 2013.

[123] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. *CoRR*, abs/2104.06569, 2021.

[124] Isaac Kohen. Four insider threats putting every company at risk. https://www.forbes.com/sites/theyec/2021/10/06/four-insider-threats-putting-every-company-at-risk/, 2021.

[125] A. Kolluri, T. Baluta, and P. Saxena. Private hierarchical clustering in federated networks. In *Proc. CCS'21*, pages 2342–2360, 2021.

[126] Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1–2):161–185, 2012.

[127] Sangeetha Kutty, Richi Nayak, and Lin Chen. A people-to-people matching system using graph mining techniques. *World Wide Web*, 17(3):311–349, 2014.

[128] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

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

[130] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

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

[132] Xiaoguang Li, Neil Zhenqiang Gong, Ninghui Li, Wenhai Sun, and Hui Li. Fine-grained poisoning attacks to local differential privacy protocols for mean and variance estimation, 2022.

[133] Seng Pei Liew, Tsubasa Takahashi, Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Network shuffling: Privacy amplification via random walks. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD'22)*, 2022.

[134] Pedro G. Lind, Marta C. González, and Hans J. Herrmann. Cycles and clustering in bipartite networks. *Physical Review E*, 72(5):1–9, 2005.

[135] Allen Liu and Ankur Moitra. Minimax rates for robust community detection. *CoRR*, abs/2207.11903, 2022.

[136] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. FLAME: Differentially private federated learning in the shuffle model. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*, 2021.

[137] Ashwin Machanavajjhala, Xi He, and Michael Hay. Differential privacy in the wild: A tutorial on current practices & open challenges. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1727–1730, 2017.

[138] M. Manjunath, K. Mehlhorn, K. Panagiotou, and H. Sun. Approximate counting of cycles in streams. In *Proc. ESA'11*, pages 677–688, 2011.

[139] Charles F Mann, David W Matula, and Eli V Olinick. The use of sparsest cuts to reveal the hierarchical community structure of social networks. *Social Networks*, 30(3):223–234, 2008.

[140] Mastodon: Giving social networking back to you. https://joinmastodon.org/, 2020.

[141] Daiki Matsunaga, Toyotaro Suzumura, and Toshihiro Takahashi. Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint arXiv:1909.10660*, 2019.

[142] Julian McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 539–547, Red Hook, NY, USA, 2012. Curran Associates Inc.

[143] A. McGregor and S. Vorotnikova. Triangle and four cycle counting in the data stream model. In *Proc. PODS'20*, pages 445–456, 2020.

[144] Frank McSherry. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 529–537. IEEE, 2001.

[145] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[146] Casey Meehan, Amrita Roy Chowdhury, Kamalika Chaudhuri, and Somesh Jha. Privacy implications of shuffling. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*, pages 1–30, 2022.

[147] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2871–2877, 2015.

[148] Minds: The leading alternative social network. https://wefunder.com/minds, 2021.

[149] Ankur Moitra, William Perry, and Alexander S Wein. How robust are reconstruction thresholds for community detection? In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 828–841, 2016.

[150] Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 814–827, 2016.

[151] Tal Moran and Moni Naor. Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, page 88–108, Berlin, Heidelberg, 2006. Springer-Verlag.

[152] C. Morris. The number of data breaches in 2021 has already surpassed last year's total. https://fortune.com/2021/10/06/data-breach-2021-2020-total-hacks/, 2021.

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

[154] Benjamin Moseley and Joshua Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *Advances in neural information processing systems*, 30, 2017.

[155] Takao Murakami and Yusuke Kawamoto. Utility-optimized local differential privacy mechanisms for distribution estimation. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security'19)*, pages 1877–1894, 2019.

[156] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[157] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

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

[159] H. H. Nguyen, A. Imine, and M. Rusinowitch. Network structure release under differential privacy. *Transactions on Data Privacy*, 9(3):215–214, 2016.

[160] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07)*, pages 75–84, 2007.

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

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

[163] Rafael Pinot. Minimum spanning tree release under differential privacy constraints. *arXiv preprint arXiv:1801.06423*, 2018.

[164] 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, pages 192–203, 2016.

[165] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 425–438, 2017.

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

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

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

[169] Pedro Ribeiro, Pedro Paredes, Miguel E.P. Silva, David Aparício, and Fernando Silva. A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys*, 54(2):28:1–28:36, 2021.

[170] G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph (p^*) models for social networks. *Social Networks*, 29(2):173–191, 2007.

[171] Garry Robins and Malcolm Alexander. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory*, 10:69–94, 2004.

[172] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. Eiffel: Ensuring integrity for federated learning, 2021.

[173] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypte: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 603–619, 2020.

[174] C. Sabater, A. Bellet, and J. Ramon. An accurate, scalable and verifiable protocol for federated differentially private averaging. *CoRR*, 2006.07218, 2021.

[175] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. GAP: Differentially private graph neural networks with aggregation perturbation. *CoRR*, 2203.00949:1–19, 2022.

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

[177] Seyed-Vahid Sanei-Mehri, Yu Zhang, Ahmet Erdem Sariyüce, and Srikanta Tirthapura. FLEET: Butterfly estimation from a bipartite graph stream. In *Proceedings of the 28th ACM international conference on Information & Knowledge Management (CIKM'19)*, pages 1201–1210, 2019.

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

[179] Mohamed Seif, Dung Nguyen, Anil Vullikanti, and Ravi Tandon. Differentially private community detection for stochastic block models. *arXiv preprint arXiv:2202.00636*, 2022.

[180] C. Seshadhri, A. Pinar, and T. G. Kolda. Triadic measures on graphs: The power of wedge sampling. In *Proc. SDM'13*, pages 10–18, 2013.

[181] Haiying Shen, Yuhua Lin, Karan Sapra, and Ze Li. Enhancing collusion resilience in reputation systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(8):2274–2287, 2016.

[182] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proc. CCS'15*, pages 1310–1321, 2015.

[183] Peter HA Sneath and Robert R Sokal. Numerical taxonomy. *Nature*, 193(4818):855–860, 1962.

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

[185] Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[186] 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 *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*, pages 703–717, 2019.

[187] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proc. WWW'11*, pages 607–614, 2011.

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

[189] 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.

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

[191] Tools: Triangle4CycleShuffle. https://github.com/Triangle4CycleShuffle/Triangle4CycleShuffle, 2022.

[192] Tools: TriangleLDP. https://github.com/TriangleLDP/TriangleLDP.

[193] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. DOULION: Counting triangles in massive graphs with a coin. In *Proc. KDD'09*, pages 837–846, 2009.

[194] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Triangle sparsifiers. *Journal of Graph Algorithms and Applications*, 15(6):703–726, 2011.

[195] Michele Tumminello, Fabrizio Lillo, and Rosario N Mantegna. Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, 75(1):40–58, 2010.

[196] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS'14)*, pages 1054–1067, 2014.

[197] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[198] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[199] Van Vu. A simple svd algorithm for finding hidden partitions. *arXiv preprint arXiv:1404.3918*, 2014.

[200] Van H Vu. Spectral norm of random matrices. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 423–430, 2005.

[201] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security'17)*, pages 729–745, 2017.

[202] Tianhao Wang, Bolin Ding, Min Xu, Zhicong Huang, Cheng Hong, Jingren Zhou, Ninghui Li, and Somesh Jha. Improving utility and security of the shuffler-based differential privacy. *Proceedings of the VLDB Endowment*, 13(13):3545–3558, 2020.

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

[204] Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, volume 1558, pages 0090–6778, 2016.

[205] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'13)*, pages 329–340, 2013.

[206] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *PAKDD*, 2013.

[207] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

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

[209] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60 60, no. 309:63–69, 1965.

[210] 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.

[211] B. Wu, K. Yi, and Z. Li. Counting triangles in large graphs by random sampling. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2013–2026, 2016.

[212] Yongji Wu, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Poisoning attacks to local differential privacy protocols for key-value data, 2021.

[213] Siyuan Xia, Beizhen Chang, Karl Knopf, Yihan He, Yuchao Tao, and Xi He. Dpgraph: A benchmark platform for differentially private graph analysis. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2808–2812, 2021.

[214] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 911–920, 2014.

[215] Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. ireduct: Differential privacy with reduced relative errors. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD'11)*, pages 229–240, 2011.

[216] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *ICLR*, 2020.

[217] Min Ye and Alexander Barga. Optimal schemes for discrete distribution estimation under local differential privacy. In *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT'17)*, pages 759—-763, 2017.

[218] Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. Towards locally differentially private generic graph metric estimation. In *Proceedings of the IEEE 36th International Conference on Data Engineering (ICDE'20)*, pages 1922–1925, 2020.

[219] 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.

[220] YouTube: System requirements. https://support.google.com/youtube/answer/78358?hl=en, 2021.

[221] Hailong Zhang, Sufian Latif, Raef Bassily, and Atanas Rountev. Differentially-private control-flow node coverage for software usage analysis. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20)*, pages 1021–1038, 2020.

[222] Hui Zhang, Ashish Goel, Ramesh Govindan, Kahn Mason, and Benjamin Van Roy. Making eigenvector-based reputation systems robust to collusion. In *Algorithms and Models for the Web-Graph: Third International Workshop, WAW 2004, Rome, Italy, October 16, 2004, Proceeedings 3*, pages 92–104. Springer, 2004.

[223] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Private release of graph statistics using ladder functions. In *Proc. SIGMOD'15*, pages 731–745, 2015.

[224] Xu Zheng, Lizong Zhang, Kaiyang Li, and Xi Zeng. Efficient publication of distributed and overlapping graph data under differential privacy. *Tsinghua Science and Technology*, 27(2):235–243, 2021.