Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy

Haipei Sun
Qatar Computing Research Institute
Stevens Institute of Technology
hsun15@stevens.edu

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

Issa Khalil
Qatar Computing Research Institute
ikhalil@hbku.edu.qa

Yin Yang
Hamad Bin Khalifa University
yyang@hbku.edu.qa

Zhan Qin
SCST*,AZFT†, Zhejiang University
qinzhan@zju.edu.cn

Hui (Wendy) Wang
Stevens Institute of Technology
hwang4@stevens.edu

Ting Yu
Qatar Computing Research Institute
tyu@hbku.edu.qa

ABSTRACT

Many real-world social networks are decentralized in nature, and the only way to analyze such a network is to collect local views of the social graph from individual participants. Since local views may contain sensitive information, it is often desirable to apply *differential privacy* in the data collection process, which provides strong and rigorous privacy guarantees. In many practical situations, the local view of a participant contains not only her own connections, but also those of her neighbors, which are private and sensitive for the neighbors, but not directly so for the participant herself. We call such information beyond direct connections an *extended local view* (*ELV*), and study two fundamental problems related to ELVs: first, how do we correctly enforce differential privacy for all participants in the presence of ELVs? Second, how can the data collector utilize ELVs to obtain accurate estimates of global graph properties?

This paper points out that when collecting ELVs, it is insufficient to apply a straightforward adaptation of *local differential privacy* (*LDP*), a commonly used scheme in practice, to protect the privacy of all network participants. The main problem is that an adversarial data collector can accumulate private information on a specific victim from multiple neighbors of the victim; even though the data collected from each neighbor is perturbed under LDP, their aggregate can still violate the victim's privacy. To prevent this attack, we formulate a novel *decentralized differential privacy* (*DDP*) scheme, which requires that each participant consider not only her own privacy, but also that of her neighbors involved in her ELV.

The stringent privacy requirement of DDP, however, makes it challenging to design an effective mechanism for data collection. Towards this goal, we design a novel multi-phase framework under *DDP* that enables an analyst to accurately estimate subgraph counts, an important property of social graphs. The main idea is that instead of collecting subgraph counts directly, which would require excessively noise, the analyst first asks individuals about their respective minimum noise scale, which is private information since it depends on the local graph structure, and, thus, must be performed under DDP. For some types of subgraphs, this process is applied *recursively*, i.e., the analyst asks about the necessary noise to be injected into the private information on the minimum local noise scale required to protect subgraph counts under DDP. As case studies, we instantiate the proposed framework for three common subgraph patterns: triangles, three-hop paths, and k-cliques. Extensive experiments using real data demonstrate that the proposed scheme leads to accurate estimates of global subgraph counts, whereas baseline solutions fail to obtain meaningful result utility.

CCS CONCEPTS

• **Security and privacy → Data anonymization and sanitization**.

KEYWORDS

decentralized differential privacy; subgraph statistics; social networks

ACM Reference Format:
Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui (Wendy) Wang, and Ting Yu. 2019. Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3319535.3354253

1 INTRODUCTION

In this paper, we consider the problem of analyzing a *decentralized* social network, in which the analyst cannot directly obtain information on the global structure of the network. Instead, the analyst needs to communicate with individual participants of the network, each of which has a limited local view of the whole social graph. Then, the analyst combines information from different participants

*School of Cyber Science and Technology
†Alibaba-Zhejiang University Joint Institute of Frontier Technologies

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '19, November 11–15, 2019, London, United Kingdom
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3354253

to estimate the global network properties. The setting of decentralized social networks could arise due to a variety of reasons. First, many social networks are inherently decentralized as there is no central organizer. For instance, the contact lists in everybody's mobile phones could be pieced together to form a giant contacts social network, though no single entity is aware of the whole network structure. Second, even when there is a centralized entity that possesses the knowledge of the entire network, that entity may choose not to share it with the analyst, out of business, legal or other considerations. For example, an organization operating an anonymous messaging app could see who messages whom (though not necessarily message contents). However, it would be difficult to share such a communication network with outsider analysts.

Collecting information of users' local views has a clear privacy implication, as they often reflect sensitive social interactions among individuals. Any analysis of decentralized social networks thus has to ensure rigorous protection of privacy. In this paper we consider private data collection under the *differential privacy* scheme [11], in which each individual injects random noise into her private information, and only releases the perturbed version to the data collector; the exact private information is never revealed. The scale of random noise is calibrated according to a pre-defined *privacy budget* [11]. A lower privacy budget leads to stronger perturbation and lower accuracy of the analysis, and vice versa.

In many social networks, a participant is aware of not only her own connections, but also a broader subgraph in her local neighborhood. We call such a subgraph an *extended local view* (*ELV*). For instance, with the default setting of Facebook (facebook.com), a user allows each of her friends to see all her connections. In the offline world, we also commonly accumulate knowledge on the relationships between our friends, e.g., when we attend a social event together. Hence, *the ELV of a social network participant often contains multi-hop neighbors and their connections.* Accordingly, the participant could reveal private information about her neighbors' connections to the data collector. To our knowledge, this is the first study that considers this problem: that an individual must protect not only her own privacy, but also the privacy of her neighbors.

The presence of ELVs poses new challenges for privacy protection. As pointed out in Section 2, a straightforward application of *local differential privacy* (*LDP*) [13], a popular scheme used in several major software systems such as Google Chrome [13] and Apple iOS / macOS [38], fails to provide sufficient privacy protection in this case. The deficiency comes from the fact that in LDP, each individual has her privacy budget locally, which covers her entire ELV regardless of what specific information from the ELV is collected. Therefore, an adversarial data collector with a target victim in mind can gather multiple reports of the same private information (e.g., whether the victim has a politically sensitive connection) from multiple individuals in the victim's neighborhood, and combine them to infer the sensitive connection with high confidence.

We address the above problem with a new privacy preservation scheme called *decentralized differential privacy* (*DDP*). As explained in Section 2, all participants of the social network *share* the same privacy budget, which covers the entire social graph; each individual, when reporting information about her ELV to the data collector, must ensure that the released information is sufficiently perturbed to protect all graph participants, i.e., the data collector cannot infer the presence or absence of any edge in the graph from all collected reports. Under DDP, however, it is rather challenging to design an effective mechanism to obtain high result utility of an analysis, since the privacy definition is over the global graph, whereas data come from individual local views.

Towards the goal of accurate social graph analysis under DDP, we propose a multi-phase framework for subgraph counting, a fundamental type of graph analyses, under (ϵ, δ)-decentralized differential privacy (defined in Section 2), where ϵ represents the total privacy budget for all nodes in the social graph, and δ controls the probability that every node's privacy is preserved. The main idea is that instead of collecting information (i.e., local subgraph counts) directly, which would require excessive noise to cover worst-case scenarios, the analyst first asks each node in the network (corresponding to an individual) about the minimum amount of noise necessary to protect the node's local subgraph count under DDP. In a subsequent phase, the analyst determines the minimum noise scale for the whole network, and collects subgraph counts accordingly. Since the noise scale now reflects the true social graph structure rather than pathological, worst-case scenarios, the end result is often significantly more accurate than directly collecting subgraph counts from nodes.

Note that in the first phase, i.e., minimum noise scale computation, the noise scale reported by a node depends on the structure of its ELV, which is private information. Therefore, the noise scale itself must be perturbed to satisfy DDP. In the process of DDP-compliant collection of noise scale, we can *recursively* apply the above framework. In particular, the analyst first asks each node to report the (second-level) minimum noise scale necessary to perturb the (first-level) noise scale for subgraph counts. Then, the analyst aggregates the second-level noise scale information to obtain a tight bound on the noise for subgraph counts. The second-level noise scale, in turn, depends on the nodes' ELV structures, and needs to be collected under DDP.

We instantiate this framework with several different types of subgraph patterns, including triangles, three-hop paths, and k-cliques, each with its own specific optimizations. Extensive experiments, using multiple real datasets, confirm that the proposed methods obtain significantly higher result utility compared to baseline solutions as well as existing ones.

In summary, we make the following contributions in the paper:

- We propose decentralized differential privacy, a new privacy protection scheme for graph analysis that correctly enforces differential privacy for all social network participants, in the presence of extended local views.
- We design a novel multi-phase, recursive framework that utilizes local graph structures to accurately estimate global subgraph counts in a decentralized graph, under the (ϵ, δ)-DDP requirement.
- We instantiate the proposed multi-phase framework on common subgraph patterns such as triangles, three-hop paths and k-cliques, and develop pattern-specific optimization for each case.
- We conduct comprehensive experiments over several real social graphs. The results show that the proposed technique consistently outperforms baseline and existing solutions in terms of result accuracy, by large margins.

2 BACKGROUND AND DECENTRALIZED DIFFERENTIAL PRIVACY

2.1 Differential Privacy

Since first proposed by Dwork et al. [11], differential privacy quickly becomes a de facto standard privacy definition in sensitive data analysis and publishing. Differential privacy was originally designed for the centralized setting, where a database of private user information is managed by a trust party, who answers queries about the database while preserving each user's privacy. Formally, we have the following definition:

Definition 2.1 (Differential privacy). A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy, if for any pair of neighboring datasets D and D' that differ by one record, and any set of possible outputs $\mathcal{S} \subseteq range(\mathcal{M})$, we have

$$\Pr(\mathcal{M}(D) \in \mathcal{S}) \leq \Pr(\mathcal{M}(D') \in \mathcal{S}) \cdot e^{\epsilon} + \delta.$$

When $\delta = 0$, $\mathcal{M}$ satisfies ϵ-differential privacy.

Essentially, differential privacy ensures that from the output of $\mathcal{M}$, one cannot distinguish whether the input is D or D' with high confidence. ϵ is often called the *privacy budget*, as it controls the strength of privacy protection offered by differential privacy.

One common technique to achieve differential privacy is the *Laplace mechanism* [11], which adds noise following the Laplace distribution to obfuscate the true outcome of a query. Specifically, let $f : \mathcal{D} \rightarrow \mathbb{R}^d$ be a function, where $\mathcal{D}$ is a set of datasets and d is a positive integer. The *sensitivity* of f, denoted Δf, is given by $\Delta f = max \parallel f(D) - f(D') \parallel_1$, over all pairs of neighboring datasets D and D'. It has been shown that $\mathcal{M}(D) = f(D) + Y$, where $Y \sim Lap(\frac{\Delta f}{\epsilon})$, satisfies ϵ-differential privacy [11]. We also call $\lambda = \frac{\Delta f}{\epsilon}$ the scale of the Laplace noise.

Differential privacy is composable: given t randomized mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_t$ that satisfy $(\epsilon_1, \delta_1), \ldots, (\epsilon_t, \delta_t)$-differential privacy respectively, the sequential composition of $\mathcal{M}_i(D)$ satisfies $(\sum_{i=1}^{t} \epsilon_i, \sum_{i=1}^{t} \delta_i)$-differential privacy.

2.2 Decentralized Differential Privacy

Let $G = (V, E)$ be a social graph, where V is the set of participants and E is the set of edges. For simplicity, we assume G is undirected. A data analyst, who have no access to the whole graph G, aims to estimate global statistical properties of G, e.g., the number of occurrences of a given subgraph such as triangles or cliques. To do so, the analyst collects information from each participant, i.e., nodes in V. Each node $v \in V$ has an *extended local view* (*ELV*) of G, denoted G_v, which corresponds to a subgraph of G surrounding v.

Since each node v clearly knows all its direct connections, its ELV G_v always contains (i) all edges involving v and (ii) all one-hop neighbors of v, each of which (say, node u) satisfies that there exists an edge $(u, v) \in E$. In this paper, we focus on a common type of ELV that also includes two-hop neighbors[1], as defined in the following.

Definition 2.2 (Two-Hop Extended Local View). Given a node $v \in V$, its two-hop extended local view (ELV) G_v consists of:

- v's one-hop neighbors: $\{u \mid u \in V \wedge (u, v) \in E\}$.
- Edges involving v: $\{e = (v, u) \mid e \in E\}$.
- v's two-hop neighbors: $\{w \mid \exists u \in V, (u, v) \in E \wedge (u, w) \in E\}$.
- Edges involving v's one-hop neighbors: $\{e = (u, w) \mid e \in E \wedge (u, v) \in E\}$

[1]We leave ELVs beyond two-hop neighborhoods for future work as they are less common in practice.

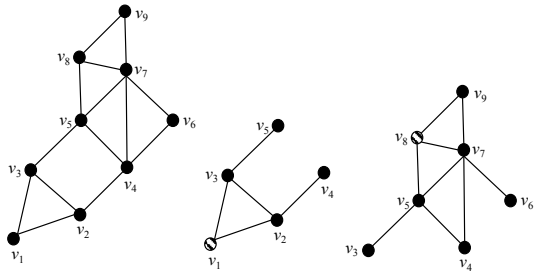


Figure 1: Example of two-hop ELVs

Figure 1 shows an example with 9 nodes $v_1, \ldots, v_9$. The ELV of v_1 contains its one-hop neighbors v_2 and v_3, two-hop neighbors v_4 and v_5, as well as the edges between these nodes. Similarly, the ELV of v_8 consists of nodes v_5, v_7, v_9 (one-hop neighbors), v_3, v_4, v_6 (two-hop neighbors), 3 edges between v_8 and its one-hop neighbors, 2 edges between its one-hop neighbors, e.g., (v_9, v_7)), and 4 edges connecting its one-hop and two-hop neighbors, e.g., (v_5, v_4).

In our setting, since the analyst needs to collect information from all social network participants, we assume that the analyst already knows their identities (i.e., membership in V), and the private information is on the connections between them (i.e., E). In other words, we focus on the *edge privacy* model [20]. This leads to the following definition of neighboring graphs:

Definition 2.3 (Neighboring graphs). Two graphs G and G' are *neighboring graphs* if G and G' only differ in one edge, i.e., G' can be obtained by adding or removing one edge from G.

Why local differential privacy is insufficient. Before presenting our privacy model, we first explain why a straightforward application of *local differential privacy* (*LDP*) [13] fails to provide adequate privacy protection in the presence of ELVs. Specifically, in LDP, each individual has her privacy budget locally, and submits a report based on her local data to the analyst, which is perturbed to satisfy differential privacy. In our setting, the local data of an individual v is her ELV G_v; thus, LDP ensures that the data collector cannot distinguish v's exact data G_v from a neighbor dataset, which in our setting would be a neighbor graph (Definition 2.3) of G_v. Formally, we define (ϵ, δ)-LDP as follows:

Definition 2.4 (Local differential privacy). Given a node $v \in V$ and its ELV G_v, a randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-LDP, iff. for any neighboring graph G'_v of G_v and any set $\mathcal{S}$ of possible outputs of $\mathcal{M}$, we have $\Pr(\mathcal{M}(G_v) \in \mathcal{S}) \leq \Pr(\mathcal{M}(G'_v) \in \mathcal{S}) \cdot e^{\epsilon} + \delta$.

To see why the above privacy definition is insufficient, consider the case where an adversarial data collector with a specific target victim u aims to find whether u is connected to another node v. To do so, the data collector asks u, as well as all of its one-hop neighbors, to report a binary value on whether edge (u, v) exists. After that, the data collector computes the mean value of these binary reports. Although each report satisfies LDP, the mean value

over multiple reports yields an increasingly accurate estimate of the true value as the number of reports grows. With a sufficiently large number of reports, the adversary obtains high confidence on whether edge (u, v) exists. This clearly violates the privacy of u.

Decentralized differential privacy. The key reason that the above definition of LDP fails to provide adequately protection on the social network participants' privacy, is that each participant only considers *her own privacy* when releasing information to the data collector, and the released information *compromises her neighbors' privacy*. To remedy this, we propose a *decentralized differential privacy* (*DDP*) scheme, which ensures that the data collector cannot infer the presence of absence of any edge in the graph from the set of *all* reports collected from all network participants. In particular, we first define the notion of neighboring ELV, as follows.

Definition 2.5 (Neighboring extended local views). Given a graph $G = (V, E)$, a node $v \in V$, its ELV $G_v \subseteq G$, and a neighboring graph G' of G. The neighboring ELV G'_v of G_v with respect to G' is then the ELV of v in G'.

Note that in the above definition, a neighboring ELV G'_v of G_v is *not* the same as a neighbor graph of G_v as Definition 2.3. In particular, two neighboring ELVs may not contain the same set of nodes, and can differ in multiple edges. For instance, in Figure 1, if we remove edge (v_1, v_2) from G, then node v_4 is no longer in the ELV of v_1, since it is now a three-hop neighbor of v_1. Similarly, if we add a new edge (v_1, v_8) to G, then nodes v_7-v_9 enter the ELV of v_1, along with all the edges connecting them. Based on the notion of neighboring ELVs, we define the proposed *decentralized differential privacy* (*DDP*) scheme, as follows.

Definition 2.6 (Decentralized differential privacy). Given a set of nodes $V = v_1, v_2, \ldots v_n$, a set of randomized mechanisms $\{\mathcal{M}_i, 1 \leq i \leq n\}$ collectively satisfy (ϵ, δ)-DDP, iff. for any two neighboring graphs $G = (V, E)$ and $G' = (V, E')$, and any subsets of possible outputs $\{\mathcal{S}_i \subseteq range(\mathcal{M}), 1 \leq i \leq n\}$, we have:

$$\Pr(\mathcal{M}_1(G_1) \in \mathcal{S}_1, \ldots, \mathcal{M}_n(G_n) \in \mathcal{S}_n)$$
$$\leq \Pr(\mathcal{M}_1(G'_1) \in \mathcal{S}_1, \ldots, \mathcal{M}_n(G'_n) \in \mathcal{S}_n) \cdot e^{\epsilon} + \delta,$$

where G_i and G'_i $(1 \leq i \leq n)$ are the neighboring ELVs of v_i with respect to G and G', respectively. In addition, since our *DDP* is under the (ϵ, δ)-DP framework, the composition rule still applies.

Discussion. Similar to the case of LDP (Definition 2.4), in DDP each node v_i applies its own randomized mechanism $\mathcal{M}_i(G_i)$ on its local data, i.e., its ELV G_i. In other words, no knowledge of the global graph G is required when node v_i computes its report $\mathcal{M}_i$ to be submitted to the data collector. On the other hand, unlike LDP where each node independently preserves its own privacy without any consideration for its neighbors, DDP covers the set of all mechanisms applied to all nodes as a whole, and protects all edges in the entire social graph G. Hence, the attack on LDP in which the data collector obtains multiple reports on the same information no longer works under DDP, since the latter by definition guarantees that the data collector cannot infer the presence of absence of an edge from *all* collected reports.

Our problem is different from existing LDP applications where users have independent data, e.g., collecting browser usage [13]. Instead, DDP can be viewed as a generalization of LDP when users' data are dependent, following the general principle of differential privacy in the local setting [10].

Designing effective mechanisms under DDP, however, is also significantly more challenging compared to the case of LDP. The main difficulty is that when generating a perturbed report $\mathcal{M}_i$, each node v_i must consider all possible neighbor graphs of the global graph G; yet, v_i does not know G, except for its own ELV G_i. We address this problem in the next section.

3 A GENERAL FRAMEWORK FOR SUBGRAPH COUNTING UNDER DDP

In this paper, we focus on a fundamental problem in graph analysis: subgraph counting, under the decentralized differential privacy requirement in Definition 2.6. Specifically, let g be a given subgraph pattern, e.g., a triangle, a k-clique, etc., the data analyst aims to estimate the number of occurrences of g in the global graph G, by collecting data from each node in G under (ϵ, δ)-DDP.

Given a node $v_i \in V$ and its two-hop ELV G_i, we define $\gamma_g(v_i)$ as the exact number of occurrences of g in G_i that involve v_i itself. In other words, occurrences of g in G_i that does not contain v_i are not counted in $\gamma_g(v_i)$. In the non-private setting, the data analyst simply collects the exact $\gamma_g(v_i)$ from each v_i, and aggregates them to obtain the total number of occurrences of g in the whole graph G. For example, when g is a triangle, the analyst simply adds up the local triangle counts from every node, and then divides the result by 3, since every triangle is reported 3 times by each of its nodes.

Under the DDP requirement, each node v_i cannot reveal the exact $\gamma_g(v_i)$, as it depends on v_i's private information G_i. Instead, each v_i submits a perturbed report $\mathcal{M}_i(G_i)$ generated through a DDP-compliant mechanism $\mathcal{M}_i$. For instance, one straightforward approach, explained below in Section 3.1, is to let each v_i submit a noisy version $\gamma^*_g(v_i)$ of $\gamma_g(v_i)$, perturbed under DDP. In general, the mechanism $\mathcal{M}_i$ applied at each node v_i can be different, as long as the set of all $\mathcal{M}_i$'s for $1 \leq i \leq n$ satisfy DDP as in Definition 2.6.

In our setting, we assume that the data analyst as well as all participants of the social network strictly follow the proposed protocols. In other words, the adversary is *honest but curious*. The case where the analyst actively breaks the protocol to gain private information, possibly in collusion with some of the network nodes, is out of the scope of this paper, and left as future work.

3.1 A Baseline Approach

We first present a baseline approach, referred to as *Pessimistic Laplace mechanism*, in which the analyst directly collects perturbed subgraph counts from the participants under DDP. As shown soon, this method incurs a prohibitively high amount of noise, due to the fact that in order to satisfy DDP, the method has to consider pathological worst-case scenarios that necessitate heavy perturbations. This highlights the challenge of mechanism design under DDP.

Pessimistic Laplace mechanism follows the standard Laplace mechanism [11]. Specifically, we first extend the notion of *sensitivity* (explained in Section 2.1) to the DDP setting, as follows.

Definition 3.1 (Sensitivity under DDP). Given a set of nodes $V = \{v_i \mid 1 \le i \le n\}$, and a function f, the sensitivity of f is defined as:

$$\Delta(f) = max_{G,G'} \sum_{i=1}^{n} \mid f(G_i) - f(G_i')|,$$

where G and G' are two arbitrary neighboring social graphs with the set of nodes V, and $\{G_i\}$ and $\{G_i'\}$ $(1 \le i \le n)$ are the sets of neighboring ELVs with respect to G and G', respectively.

In Pessimistic Laplace mechanism, given a subgraph pattern g, each participant directly reports a noisy version of its subgraph count $\gamma_g(v_i)$. Formally, $f = \Gamma_g$, where $\Gamma_g(G_i) = \gamma_g(v_i)$. The sensitivity $\Delta(\Gamma_g)$ of Γ_g, however, is prohibitively high, leading to poor result utility.

To explain, consider a simple case where g is a triangle. (Other cases of g are discussed later in Section 4.) We have $\Delta(\Gamma_\triangle) = 3(n-2)$ since (i) in the worst case, an edge (u, v) in an n-node graph can appear in $n-2$ triangles, when both u and v are connected to all other nodes in the entire social graph, and (ii) each triangle is reported three times by its three vertices, respectively. Note that to satisfy DDP, in which privacy is defined over the entire graph G, we must consider all possible cases of G, including the above worst case. According to the following lemmata, adding Laplace noise $Lap\left(\frac{3(n-2)}{\epsilon}\right)$ into $\gamma_\triangle(v_i)$ ensures ϵ-DDP, but leads to a prohibitively high noise variance in the resulting estimated triangle count: $O\left(\frac{n^3}{\epsilon^2}\right)$. Note that the noise variance only depends on the number of n, regardless of the structure of the actual social graph G, since the sensitivity is derived from the worst-case scenario.

Lemma 3.2. *Adding i.i.d. Laplace noise $Lap\left(\frac{3(n-2)}{\epsilon}\right)$ to $\gamma_\triangle(v_i)$ of each v_i satisfies ϵ-DDP* [2].

Lemma 3.3. *Pessimistic Laplace mechanism leads to $O\left(\frac{n^3}{\epsilon^2}\right)$ variance in the estimated total triangle count.*

Discussion. The above description of Pessimistic Laplace ensures ϵ-DPP, which can be viewed as (ϵ, δ)-DDP for the strict case that $\delta = 0$. When $\delta > 0$, we can improve its accuracy as follows. Each node v_i reports its exact subgraph count $\gamma_g(v_i)$ with probability δ, and the perturbed subgraph count (according to the above approach) otherwise. Alternatively, we could follow the Gaussian mechanism [1] instead of the Laplace mechanism, which we call the *Pessimistic Gaussian* mechanism. Neither of these approaches, however, addresses the core issue that we inject noise according to some pathological worst-case scenario, regardless of the actual structure of the social graph G. As our experiments in Section 5 shows, none of these baseline approaches obtain competitive result accuracy.

3.2 Proposed Multi-Phase Framework

Local sensitivity. Observe that the main reason that Pessimistic Laplace mechanism incurs a high noise variance is that it injects noise based on the *global* sensitivity of the graph count function Γ_g, which is determined by a worst-case scenario. In real social networks, however, such pathological scenarios are rare. The proposed framework avoids the excessive noise due to the worst case, by employing the concept of *local sensitivity* [30], defined as follows.

Definition 3.4 (Local sensitivity under DDP). Given a global graph $G = (V, E)$ containing nodes $V = \{v_i | 1 \le i \le n\}$, and a function f. The local sensitivity of f is defined as:

$$LS(f) = max_{G,G'} \sum_{i=1}^{n} |f(G_i) - f(G_i')|,$$

where G' is an arbitrary neighboring graph of G, and $\{G_i\}$ and $\{G_i'\}$ $(1 \le i \le n)$ are the sets of ELVs with respect to G and G', respectively.

For instance, in the case of triangle counting, we have $LS_G(\Gamma_\triangle) = \max_{G'} \sum_{i=1}^{n} |\gamma_\triangle(v_i) - \gamma'_\triangle(v_i)|$, where G' is any neighboring graph of G, and $\gamma'_\triangle(v_i)$ denotes the triangle count to be reported by v_i in G'. In other words, $LS_G(\Gamma_\triangle)$ measures the maximum number of triangles affected by adding or removing one edge in G. If $LS_G(\Gamma_\triangle) \ll n-2$, is it sufficient to ask each user to inject Laplace noise $Lap\left(\frac{3LS_G(\Gamma_\triangle)}{\epsilon}\right)$ into her triangle count? Unfortunately, the answer is no: it has been shown in the literature [30] that injecting noise according to local sensitivity fails to satisfy differential privacy. Figure 2 shows an example in the same setting as in Figure 1. Consider the task of triangle counting. The local sensitivity at node v_1 is 1, since adding / removing any edge in the global graph G (shown in Figure 1) can only change the number of triangles in G_1 by at most 1. Observe that the local sensitivity value at v_1 in fact only depends on its ELV G_1, regardless of the structure of G outside G_1. Now, consider a neighbor graph G' of G, which is identical to G except for one addition edge (v_1, v_8). On G', the local sensitivity of triangle counting at v_1 becomes 2, e.g., adding an edge (v_1, v_5) leads to two additional triangles. Therefore, the value of the local sensitivity reveals whether the global is G or G', meaning that using its exact value in the randomized mechanism $\mathcal{M}_i$ would violate differential privacy.

In the traditional, centralized setting of differential privacy, there exist solutions (e.g., [22, 24, 30, 42]) that correctly inject noise based on an adjusted version of local sensitivity. However, as reviewed in Section 6, none of them applies to our setting, since they all rely on knowledge about the global graph G. Our proposed framework directly tackles the root problem that local sensitivity fails to satisfy differential privacy: that the noise scale itself is private information. The idea is simple: we still injects Laplace noise into each node's subgraph count $\gamma_g(v_i)$, but the scale of the noise is not deterministic. Instead, the noise scale is a random variable sampled from a carefully chosen distribution, such that with $1 - \delta$ probability, the injected Laplace noise can conceal the existence or absence of any particular edge in G. This idea is explored in previous work [21, 24] under the centralized differential privacy model (CDP). But its adoption in the DDP setting is highly non-trivial due to the fundamental differences between CDP and DDP.

Two-phase framework. Let g be a subgraph pattern and $\Gamma_g = \{\gamma_g(v_1), \ldots, \gamma_g(v_n)\}$ be the set of subgraph counts that each user is asked to report. Our solution consists of two phases. Phase 1 applies an (ϵ_1, δ_1)-DDP algorithm to collect information about each user, and decides an appropriate noise scale λ. After that, Phase 2 asks each user to report her subgraph count injected with Laplace

[2] Proofs can be found in the Appendix.

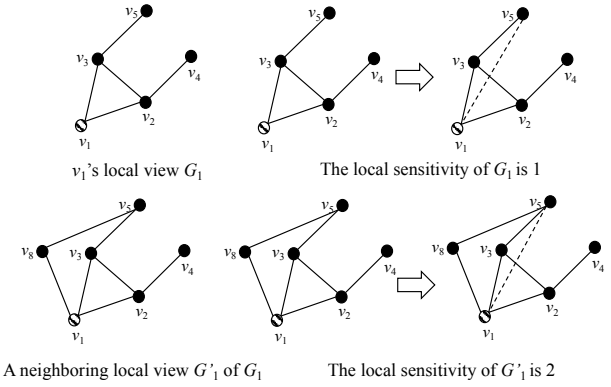


Figure 2: Local view sensitivity

noise $Lap(\lambda)$, and we show that this satisfies ϵ_2-DDP with at least $1-\delta_2$ probability, for some ϵ_2 and δ_2. According to the sequential composition property of differential privacy (explained in Section 2.1), the two-phases as a whole satisfies (ϵ, δ)-DDP where $\epsilon = \epsilon_1 + \epsilon_2$ and $\delta = \delta_1 + \delta_2$. It turns out that Phase 1 of our solution needs to be custom-designed for different types of subgraph patterns g, which we detail in Section 4. In what follows, we present the main requirements for Phase 1 necessary for the proposed framework to satisfy (ϵ, δ)-DDP.

Let λ be the noise scale returned by Phase 1 given a graph G. We require that λ satisfy the following two conditions:

(1) λ is generated using an (ϵ_1, δ_1)-DDP algorithm.
(2) With at least $1-\delta_2$ probability, we have

$$\epsilon_2 \cdot \lambda \geq LS_G\left(\Gamma_g\right). \tag{1}$$

Intuitively, in the above two-phase framework, Phase 1 essentially aims to establish an upper bound on the local sensitivity $LS_G(\Gamma_g)$ to be used in Phase 2. This upper bound is estimated by $\frac{\lambda}{\epsilon_2}$. There is a chance (i.e., with probability δ_2) that Phase 1 can fail, in which case $\frac{\lambda}{\epsilon_2} < LS_G(\Gamma_g)$. Such failures are tolerated by the (ϵ, δ)-differential privacy definition as long as the failure probability satisfies $\delta_2 < \delta$. Meanwhile, in the framework Phase 1 is left as a black box, as long as it satisfies the above conditions.

From two-phase to multi-phase. In the above framework, Phase 1 is a black box that involves an (ϵ_1, δ_1)-DDP mechanism to compute λ. This mechanism can be realized by *recursively* applying the two-phase framework itself, leading to a multi-phase solution. Specifically, we split Phase 1 into two sub-phases: Phase 1.1 and Phase 1.2. Meanwhile, we partition the privacy parameters (ϵ_1, δ_1) allocated to Phase 1 into $(\epsilon_{1,1}, \delta_{1,1})$ and $(\epsilon_{1,2}, \delta_{1,2})$, and assign them to Sub-Phases 1.1 and 1.2, respectively.

As before, in Sub-Phase 1.1, we estimate an appropriate noise scale λ_1 using a black box $(\epsilon_{1,1}, \delta_{1,1})$-DDP mechanism. Then, Sub-Phase 1.2 applies the Laplace mechanism, which, when used with a correct noise scale, satisfies $\epsilon_{1,2}$-differential privacy. Sub-Phase 1.1 has a probability $\delta_{1,2}$ to fail, i.e., its output noise scale is not sufficiently large for Sub-Phase 1.2 to satisfy $\epsilon_{1,2}$-differential privacy.

Essentially, Sub-Phase 1.1 estimates the local sensitivity $LS_G(LS_G(\Gamma_g))$ of local sensitivity $LS_G(\Gamma_g)$ of graph counts. Its output is an upper bound of the true $LS_G(LS_G(\Gamma_g))$ with probability $1-\delta_{1,2}$. Then, Sub-Phase 1.2 uses the estimated $LS_G(LS_G(\Gamma_g))$ to output an estimated $LS_G(\Gamma_g)$, which is an upper bound of its true value with probability $1-\delta_2$. Finally, Phase 2 applies the estimated $LS_G(\Gamma_g)$ to obtain randomized subgraph counts. A concrete instantiation is presented later in Section 4.2.

Correctness of the framework. To formally establish the correctness of the proposed multi-phase framework, it suffices to prove the two-phase case; the multi-phase case can then be proved by induction. For the two-phase framework, we prove that the two requirements for Phase 1 (i.e., it satisfies (ϵ_1, δ_1)-DDP and its output λ satisfies Inequality (1) with probability $1-\delta_2$) ensure that our solution achieves (ϵ, δ)-DDP for $\epsilon = \epsilon_1 + \epsilon_2$ and $\delta = \delta_1 + \delta_2$.

Let (λ, Y) denote the output of Phase 1, and Γ_g^* denote the set of noisy subgraph counts returned by Phase 2. Here, Y represents all additional private information besides λ that is revealed to the data collector during Phase 1. Let $\mathcal{S}_\lambda$ (resp. $\mathcal{S}_\Gamma$) be an arbitrary set of possible outputs from Phase 1 (resp. Phase 2). We will establish the privacy guarantee of our solution by showing that, for any neighboring graphs G and G' and for any Γ_g^*, λ, and Y,

$$\begin{aligned}&\Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}_\lambda \;\middle|\; G\right]\\ &\leq e^\epsilon \cdot \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}_\lambda \;\middle|\; G'\right] + \delta.\end{aligned} \tag{2}$$

Let $\mathcal{S}'_\lambda$ be the subset of $\mathcal{S}_\lambda$ such that

$$\mathcal{S}'_\lambda = \left\{(\lambda, Y) \;\middle|\; (\lambda, Y) \in \mathcal{S}_\lambda \wedge \epsilon_2 \cdot \lambda \geq LS_G\left(\Gamma_g\right)\right\}.$$

We have

$$\begin{aligned}&\Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}_\lambda \;\middle|\; G\right]\\ &= \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G\right] + \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}_\lambda \setminus \mathcal{S}'_\lambda \;\middle|\; G\right]\\ &\leq \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G\right] + \delta_2,\end{aligned} \tag{3}$$

since Phase 1 ensures Eq. (1) with at least $1-\delta_2$ probability. Then, to prove Eq. (2), it suffices to show that

$$\begin{aligned}&\Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G\right]\\ &\leq e^\epsilon \cdot \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G'\right] + \delta_1.\end{aligned} \tag{4}$$

Since (λ, Y) is generated using an (ϵ_1, δ_1)-DDP algorithm, we have

$$\Pr\left[(\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G\right] \leq e^{\epsilon_1} \cdot \Pr\left[(\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G'\right] + \delta_1.$$

Therefore,

$$\begin{aligned}&\Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma, (\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G\right]\\ &= \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma \;\middle|\; (\lambda, Y) \in \mathcal{S}'_\lambda, G\right] \cdot \Pr\left[(\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G\right]\\ &\leq \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma \;\middle|\; (\lambda, Y) \in \mathcal{S}'_\lambda, G\right] \cdot \left(e^{\epsilon_1} \cdot \Pr\left[(\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G'\right] + \delta_1\right)\\ &\leq e^{\epsilon_1} \cdot \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma \;\middle|\; (\lambda, Y) \in \mathcal{S}'_\lambda, G\right] \cdot \Pr\left[(\lambda, Y) \in \mathcal{S}'_\lambda \;\middle|\; G'\right] + \delta_1\end{aligned} \tag{5}$$

We will show that for any $x \geq LS_G(\Gamma_g)/\epsilon_2$, any y, and any set Υ of noisy subgraph counts,

$$\Pr\left[\Gamma_g^* = \Upsilon \;\middle|\; \lambda = x, Y = y, G\right] \leq e^{\epsilon_2} \cdot \Pr\left[\Gamma_g^* = \Upsilon \;\middle|\; \lambda = x, Y = y, G'\right], \tag{6}$$

which would lead to

$$\Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma \;\middle|\; (\lambda, Y) \in \mathcal{S}'_\lambda, G\right] \leq e^{\epsilon_2} \cdot \Pr\left[\Gamma_g^* \in \mathcal{S}_\Gamma \;\middle|\; (\lambda, Y) \in \mathcal{S}'_\lambda, G'\right].$$

This, when combined with Eq.(3), (4), and (5), shows that our two-phase approach ensures (ϵ, δ)-DDP.

Let $\Gamma_g = \{\gamma_g(v_i)\}$ and $\Gamma'_g = \{\gamma'_g(v_i)\}$ be the subgraph counts on G and G' respectively. Since Phase 2 generates Γ_g^* by injecting Laplace noise $Lap(\lambda)$ into each subgraph count, we have

$$\begin{aligned}
&\frac{\Pr\left[\Gamma_g^* = \Upsilon \;\middle|\; \lambda = x, Y = y, G\right]}{\Pr\left[\Gamma_g^* = \Upsilon \;\middle|\; \lambda = x, Y = y, G'\right]} \\
&= \frac{\frac{1}{2x}\exp\left(-\frac{1}{x}\sum_{i=1}^{n}\left|\gamma_g^*(v_i) - \gamma_g(v_i)\right|\right)}{\frac{1}{2x}\exp\left(-\frac{1}{x}\sum_{i=1}^{n}\left|\gamma_g^*(v_i) - \gamma'_g(v_i)\right|\right)} \\
&\leq \exp\left(\frac{1}{x}\sum_{i=1}^{n}\left|\gamma_g(v_i) - \gamma'_g(v_i)\right|\right) \leq \exp\left(\frac{1}{x} LS_G(\Gamma_g)\right) \leq e^{\epsilon_2},
\end{aligned}$$

where the last inequality is due to $x \geq LS_G(\Gamma_g)$. Therefore, Eq. (6) is proved. Thus, we arrive at the following theorem:

Theorem 3.5. *The proposed two-phase framework ensures (ϵ, δ)-DDP whenever $\epsilon_1 + \epsilon_2 \leq \epsilon$ and $\delta_1 + \delta_2 \leq \delta$.*

4 COUNTING DIFFERENT TYPES OF SUBGRAPHS UNDER DDP

In this section, we instantiate the proposed multi-phase framework for subgraph counting under DDP on three types of common subgraphs: triangles, three-hop paths and k-cliques. These instantiations are themselves non-trivial, and involve subgraph-specific optimizations necessary to achieve high result accuracy.

4.1 Triangles

First-cut solution. According to Section 3.2, to achieve (ϵ, δ)-DDP in triangle counting, we need to design an (ϵ_1, δ_1)-DDP algorithm that returns a noise scale λ which satisfies $\epsilon_2 \cdot \lambda \geq LS_G(\Gamma_\triangle)$ with at least $1-\delta_2$ probability. Equivalently, we can compute a differentially private upper bound τ of $LS_G(\Gamma_\triangle)$, and then set $\lambda = \epsilon_2 \cdot \tau$. In the case of triangle counting, our solution sets $\delta_1 = 0$ and $\delta_2 = \delta$.

First of all, we introduce a method for computing a probabilistic upper bound of any value α, when given a noisy version of α injected with Laplace noise:

Lemma 4.1. *Let x be any real value, and $x^* = x + Lap(\alpha)$ for some $\alpha > 0$. Then, with $1 - \delta$ probability,*

$$x^* + \alpha \cdot \log\left(\frac{1}{2\delta}\right) \geq x.$$

By Lemma 4.1, if we are to derive τ, we may first inject Laplace noise $Lap(\lambda_c)$ into $LS_G(\Gamma_\triangle)$, and then sets τ to the noisy value plus $\lambda_c \cdot \log\left(\frac{1}{2\delta}\right)$. This approach, however, only works if (i) $LS_G(\Gamma_\triangle) + Lap(\lambda_c)$ can be computed in the decentralized setting, and (ii) $Lap(\lambda_c)$ is sufficient to ensure (ϵ_1, δ_1)-DDP for $LS_G(\Gamma_\triangle)$. In relation to this, we first note that $LS_G(\Gamma_\triangle)$ equals the maximum number of common neighbors shared by two users in G, i.e.,

$$LS_G(\Gamma_\triangle) = \max_{v_i, v_j \in G, i \neq j} 3 \cdot |N(v_i) \cap N(v_j)|, \tag{7}$$

where $N(v)$ denotes the set of neighbors of user v. This is because (i) adding or removing one edge $\langle v_i, v_j \rangle$ only affects those triangles that contain both v_i and v_j as vertices, (ii) the number of such triangles equals $|N(v_i) \cap N(v_j)|$, and (iii) each of these triangles is reported by three users.

Based on Eq. (7), we may compute a probabilistic upper bound of $LS_G(\Gamma_\triangle)$ in the decentralized setting as follows. First, for each user v_i, we ask her to compute the maximum number $c(v_i)$ of common neighbors that she shares with any other user in her local view, i.e.,

$$c(v_i) = \max_{v_j \in G_i \wedge j \neq i} |N(v_i) \cap N(v_j)|. \tag{8}$$

Note that for any node $v_k \notin G_i$, we have $|N(v_i) \cap N(v_k)| = 0$. Then, we ask v_i to report a noisy version $c^*(v_i)$ of $c(v_i)$ injected with Laplace noise $Lap(\lambda_c)$. After that, we take

$$c^\top(v_i) = c^*(v_i) + \lambda_c \cdot \log\left(\frac{1}{2\delta}\right)$$

as a probabilistic upper bound of $c(v_i)$, and we set

$$\tau = \max_{v_i \in G} c^\top(v_i)$$

as a probabilistic upper bound of $LS_G(\Gamma_\triangle)$.

Unfortunately, the above approach requires $\lambda_c = n/\epsilon_1$ to ensure that τ satisfies ϵ_1-DDP. To explain, observe that adding or removing one edge in G could change each $c(v_i)$ by 1 in the worst case. Therefore, the sensitivity of $\{c(v_1), \ldots, c(v_n)\}$ equals n, due to which we need $\lambda_c \geq n/\epsilon_1$ to guarantee that $\{c^*(v_1), \ldots, c^*(v_n)\}$ is ϵ_1-differentially private. As such, we have $\tau > \frac{n}{\epsilon_1} \cdot \log\left(\frac{1}{2\delta}\right)$, which leads to a prohibitive of noise in Phase 2 of our solution.

Optimized solution. To address the deficiency of the aforementioned method, we propose to avoid directly collecting $\{c(v_1), \ldots, c(v_n)\}$ (as it has a high sensitivity), but let each user v_i report a probabilistic upper bound $d^\top(v_i)$ of her degree $d(v_i)$, i.e., the number of 1-hop neighbors of v_i. The rationale is that $d(v_i) \geq c(v_i)$ holds for any v_i, and hence, we can use a probabilistic upper bound of $d(v_i)$ in place of $c(v_i)$.

In particular, for some λ_d, δ_d that we clarify shortly, we ask each user v_i to inject Laplace noise $Lap(\lambda_d)$ into her degree $d(v_i)$, and then report the noisy degree $d^*(v_i)$; after that, we take

$$d^\top(v_i) = d^*(v_i) + \lambda_d \cdot \log\left(\frac{1}{2\delta_d}\right) \tag{9}$$

as a probabilistic upper bound of $d(v_i)$. We can then set $\tau = \max_{v_i \in G} d^\top(v_i)$ as a probabilistic upper bound of $LS_G(\Gamma_\triangle)$.

The advantage of this approach is that only a small amount of noise is needed in $\{d^*(v_1), \ldots, d^*(v_i)\}$. In particular, since adding or removing an edge in G only changes the degrees of two nodes, each by 1, the sensitivity of $D = \{d(v_1), \ldots, d(v_i)\}$ equals 2. Hence, injecting Laplace noise $Lap(2/\epsilon)$ into D_g achieves ϵ-DDP. The disadvantage, however, is that $d(v_i)$ could be a rather loose upper bound of $c(v_i)$, due to which setting $\tau = \max_{v_i \in G} d^\top(v_i)$ could still lead to excessive noise in Phase 2. This motivates us to develop a *hybrid* approach that combines both $c^\top(v_i)$ and $d^\top(v_i)$.

In the proposed hybrid approach, *different nodes report different information to the analyst*, i.e., $\mathcal{M}_i$ varies depending on v_i. The main idea is as follows. First, we obtain a probabilistic degree upper bound $d^\top(v_i)$ for every user v_i, and we identify the set S of nodes whose

Algorithm 1: Optimized Two-Phase Approach

Input : Privacy budget for phase 1 ϵ_1, privacy budget for phase 2 ϵ_2, invalidation probability δ, a large number h';
Output : Scale λ;

1 $\lambda_d = \frac{2}{0.5\epsilon_1}$; // Server
2 $\delta' = \frac{\delta}{2h'+2}$; // Server
3 **for** *each* v_i **do**
4 $d^\top(v_i) = d(v_i) + Lap(\lambda_d) + \lambda_d \cdot \log\left(\frac{1}{2\delta'}\right)$; // Client
5 Report $d^\top(v_i)$ to server; // Client
6 Sort $\{v_i\}$ into $\{v_{[1]}, v_{[2]}, \ldots, v_{[n]}\}$ by $d^\top(v_i)$ in descending order; // Server
7 **for** $i = 1$ *to* h' **do** // Server
8 **if** $\frac{i}{0.5\epsilon_1} \cdot \log\left(\frac{1}{2\delta'}\right) \geq d^\top(v_{[i+2]})$ **then** // Server
9 **break**; // Server
10 $h = \lceil i/2 \rceil$; // Server
11 $S = \{v_{[i]} | 2 \leq i \leq h+1\}$; // Server
12 $\lambda_c = \frac{h}{0.5\epsilon_1}$; // Server
13 **for** *each* $v_i \in S$ **do**
14 $c^\top(v_i) = c(v_i) + Lap(\lambda_c) + \lambda_c \cdot \log\left(\frac{1}{2\delta'}\right)$ // Client
15 $c^\dagger(v_i) = \min\{c^\top(v_i), d^\top(v_i)\}$ // Client
16 Report $c^\dagger(v_i)$ to server; // Client
17 $\lambda = 3\max\{\frac{1}{\epsilon_2} d^\top(v_{[h'+2]}), \frac{1}{\epsilon_2} \max_{v_i \in S} c^\dagger(v_i)\}$; // Server
18 **return** λ // Server

degree upper bounds are the largest. Intuitively, for any $v \in S$, using $d^\top(v)$ as an upper bound of $c(v)$ is likely to be ineffective, since $c(v)$ could be much smaller than $d^\top(v)$. Therefore, for each $v \in S$, we derive $c^\top(v)$ as an alternative upper bound of $c(v)$, instead of relying solely on $d^\top(v)$. Note that in this case, the amount of Laplace noise injected into $c^\top(v)$ is $O(|S|)$ instead of $O(n)$, since we do not only request $c^\top(v)$ for any $v \notin S$. Finally, we combine $d^\top(v_1), \ldots, d^\top(v_n)$ and $c^\top(v)$ ($v \in S$) to compute an improved upper bound of $LS_G(\Gamma_\triangle)$. We will explain later how to select the set of nodes S used in this step later

Algorithm 1 shows the pseudo-code of the proposed solution for Phase 1 of the framework. The algorithm involves two rounds of reporting; all nodes participate in the first round, and only a selected few participate in the second round. Specifically, the algorithm starts by splitting budget λ_d and δ' in Lines 1-2. This is done by the server. Here we divide the budget ϵ_1 into two halves for the two-round reporting. We also divide the probability δ into $2h'+2$ parts, where h' is a user-specified number indicating the maximum number of clients to do the second round of reporting. The specific value of h' slightly affects the accuracy of the estimation result, but not the correctness of the algorithm. In our experiments, we found that $h' = 100$ usually leads to good results.

After that, the server sends these parameters to all the clients, i.e., nodes in the social network. Lines 4-5 are executed by each client, which calculate the probabilistic upper bound of the actual degree $d(v_i)$ and report it to the server, i.e., the data collector. Then, in Lines 6-11, the server uses a heuristic to decide $h \leq h'$, the number of clients who do the second-round reporting, and obtains the set S of h nodes. The intuition of the heuristic will be explained shortly.

In Line 12, the server spends another half of ϵ in the second-round reporting. After getting λ_c, as shown in Lines 14-16, all the clients in S calculate $c^\top(v_i)$ as their probabilistic upper bound of common neighbor counts, then get their final upper bound $c^\dagger(v_i)$ in Line 15 and report it to the server.

Finally, in Lines 17-18, the server computes the final upper bound of each client, and selects the maximum one. However, it is possible that the client v_i with maximum $c(v_i)$ is not in S, and hence, $\max_{v_i \in S} c^\dagger(v_i)$ is unable to cover the sensitivity. In such case, the client is hiding in $\{v_{[h'+2]}, \ldots, v_{[n]}\}$, and it has to be covered by $d^\top(v_{[h'+2]})$. That is reason that we derive the final λ by getting the maximum value in Line 17. Finally, since every addition/removal of a triangle is always observed by 3 clients, we multiply the result by 3.

Lemma 4.2. *Algorithm 1 satisfies ϵ_1-DDP and, with at least $1-\delta$ probability, returns $\lambda \geq \frac{1}{\epsilon_2} LS_G(\Gamma_\triangle)$.*

As mentioned above, Lines 6-10 in Algorithm 1 are a heuristic to choose h, the size of the set of nodes S who report further their $c^\top$ to deduce a final upper bound λ. Clearly, if h is too small, the final upper bound would be close to the second largest $d^\top$, which is likely to be much larger than the maximum $c(v)$ among all nodes. If h is too large, each node v in S would end up adding too much noise to $c(v)$, again resulting in a much larger final upper bound. To find an appropriate h, we have the following intuition. Suppose that $h = i$. Observe that for each node v in S, besides the Laplace noise, it also needs to add an additional noise $\frac{i}{0.5\epsilon_1} \cdot log\left(\frac{1}{2\delta'}\right)$ to $c(v)$. If this noise is already bigger than $d^\top(v_{[h+2]})$, then any bigger i would not result in smaller final upper bounds. Meanwhile, since $c^\top(v)$ also includes $c(v)$, the i we have now is likely to be more than enough to ensure $c^\top(v) > d^\top$. Therefore, we set $h = \frac{i}{2}$ to derive the final upper bound. In Section 5, we experimentally evaluate the quality of this heuristic in choosing h.

4.2 Three-Hop Paths

Baseline: Pessimistic Laplace mechanism. A three-hop path refers to a set of three edges that form a simple path. Suppose that we let each user v_i report the number $\gamma_\sqcup(v_i)$ of three-hop paths in which she is one of the two nodes in the middle (referred to as the *internal* nodes). In that case, the Pessimistic Laplace mechanism (explained in Section 3.1) lets each v_i inject Laplace noise $Lap(6(n-2)(n-3)/\epsilon)$ into $\gamma_\sqcup(v_i)$ before reporting it. To explain, observe that for any two nodes u and v, there can be at most $6(n-2)(n-3)$ three-hop paths in which $\langle u, v\rangle$ is one of the edges. Accordingly, the sensitivity of $\Gamma_\sqcup = \{\gamma_\sqcup(v_1), \ldots, \gamma_\sqcup(v_n)\}$ equals $6(n-2)(n-3)$, since (i) adding or removing an edge $\langle u, v\rangle$ in G affects at most $3(n-2)(n-3)$ three-hop paths, and (ii) each three-hop path is reported by two users.

Two-phase solution. Next we apply the proposed two-phase framework, for which the key is to develop an (ϵ_1, δ)-differentially private algorithm for computing a probabilistic upper bound of $LS_G(\Gamma_\sqcup)$. Observe that

$$LS_G(\Gamma_\sqcup) \leq \max_{v_i, v_j \in G, i \neq j} 2\Big(d(v_i) \cdot d(v_j) + \sum_{v_\ell \in N(v_i)} (d(v_\ell) - 1) + \sum_{v_\ell \in N(v_j)} (d(v_\ell) - 1)\Big), \quad (10)$$

$$\begin{cases}\text{Phase 1:} & \text{Derive } d^\top(v_i) \text{ with } Lap\left(\frac{2}{\epsilon_{1a}}\right) \text{ and } \psi^\top(v_i) \text{ with } Lap\left(\frac{8(n-2)}{\epsilon_{1b}}\right)\\ \text{Phase 2:} & \text{Derive } \Gamma_\sqcup \text{ based on } d^\top(v_i) \text{ and } \psi^\top(v_i)\end{cases}$$

(a) Two-phase solution.

$$\begin{cases}\text{Phase 1:} & \begin{cases}\text{Sub-Phase 1:} & \text{Derive } d^\top(v_i) \text{ with } Lap\left(\frac{2}{\epsilon_{1a}}\right)\\ \text{Sub-Phase 2:} & \text{Derive } \psi^\top(v_i) \text{ based on } Lap\left(\frac{4\left(d^\top_{(1)}+d^\top_{(2)}\right)}{\epsilon_{1b}}\right)\end{cases}\\ \text{Phase 2:} & \text{Derive } \Gamma_\sqcup \text{ based on } d^\top(v_i) \text{ and } \psi^\top(v_i)\end{cases}$$

(b) Proposed three-phase solution.

Figure 3: Comparison of two-phase and three-phase solutions for counting three-hop paths under DDP.

where $d(v)$ denotes the degree of v and $N(v)$ denotes the set of neighbors of v. This is because there can be (i) at most $d(v_i) \cdot d(v_j)$ three-hop paths in which v_i and v_j are the two internal nodes, (ii) at most $\sum_{v_\ell \in N(v_i)} (d(v_\ell) - 1) + \sum_{v_\ell \in N(v_j)} (d(v_\ell) - 1)$ three-hop paths in which $\langle v_i, v_j \rangle$ is an edge and either v_i or v_j is an end point. Let $\psi(v_i) = \sum_{v_\ell \in N(v_i)} 2(d(v_\ell) - 1)$ and $\Psi = \{\psi(v_1), \ldots, \psi(v_n)\}$. By Eq. (10), if we can derive probabilistic upper bounds $d^\top(v_i)$ and $\psi^\top(v_i)$ for $d(v_i)$ and $\psi(v_i)$, respectively, then we can use $\max_{v_i, v_j \in G, i \neq j} \left(2d^\top(v_i) \cdot d^\top(v_j) + \psi^\top(v_i) + \psi^\top(v_j)\right)$ as an upper bound of $LS_G(\Gamma_\sqcup)$.

Note that $d^\top(v_i)$ can be computed based on Eq. (9). To derive $\psi^\top(v_i)$, we may utilize Lemma 4.1 as follows. First, we let each user v_i inject Laplace noise $Lap\left(\lambda_\psi\right)$ into $\psi(v_i)$, to obtain a noisy value $\psi^*(v_i)$, and then setting

$$\psi^\top(v_i) = \psi^*(v_i) + \lambda_\psi \cdot \log\left(\frac{1}{2\delta_\psi}\right), \tag{11}$$

for some δ_ψ. This approach, however, offers inferior accuracy, as it requires $\lambda_\psi \geq 8(n-2)/\epsilon$ to achieve ϵ-DDP, because the sensitivity of Ψ equals $8(n-2)$. To understand this, observe that when all nodes in G are connected to each other, we have $\psi(v_i) = 2(n-1)(n-2)$ for every user v_i. If we remove the edge $\langle v_1, v_2 \rangle$, then we have $\psi(v_1) = \psi(v_2) = 2(n-2)^2$, and $\psi(v_j) = 2(n-1)(n-2) - 4$ for $j \geq 3$. This worst-case scenario leads to a total change of $8(n-2)$ in Ψ, which explains the sensitivity of Ψ.

Proposed three-phase solution. Next we present the proposed solution, which recursively applies the two-phase framework to the estimation of Ψ. Observe that although Ψ has a large sensitivity, its local sensitivity $LS_G(\Psi)$ can be much smaller:

$$LS_G(\Psi) \leq \max_{v_i, v_j \in G \wedge i \neq j} 4\left(d(v_i) + d(v_j)\right). \tag{12}$$

In particular, when one edge $\langle v_i, v_j \rangle$ is added or removed in G, (i) $\psi(v_i)$ changes by at most $2d(v_j)$, (ii) $\psi(v_j)$ changes by at most $2d(v_i)$, and (iii) $\{\psi_\ell \mid \ell \neq i, j\}$ changes by at most $2d(v_i) + 2d(v_j)$.

The fact that $LS_G(\Psi)$ is relatively small motivates the recurisve application of the two-phase framework on the estimation of Ψ, i.e., we first compute a probabilistic upper bound of $LS_G(\Psi)$, and then inject noise into Ψ accordingly. After that, we derive a probabilistic upper bound $\psi^\top(v_i)$ for each user v_i, and ask them to report their square counts injected with Laplace noise $Lap\left(\frac{max_{v_i \in G} \psi^\top(v_i)}{\epsilon_2}\right)$. Figure 3 illustrates the differences between the proposed three-phase solution and the aforementioned two-phase solution, which suffers from the large sensitivity of Ψ.

We now explain how we derive a probabilistic upper bound $\psi^\top(v_i)$ of each $\psi(v_i)$. First, we compute a probabilistic degree upper bound $d^\top(v_i)$ for each v_i, based on Eq. (9). Let $d^\top_{(1)}$ and $d^\top_{(2)}$ be the largest two degree upper bounds. By Eq. (12), we can take $4\left(d^\top_{(1)} + d^\top_{(2)}\right)$ as a probabilistic upper bound of $LS_G(\Psi)$. After that, we let each user v_i inject Lapalce noise $Lap(\lambda_\psi)$ into $\psi(v_i)$, with $\lambda_\psi = 4\left(d^\top_{(1)} + d^\top_{(2)}\right)/\epsilon_1$, and then report the noisy value $\psi^*(v_i)$. Then, we derive an upper bound $\psi^\top(v_i)$ of each $\psi(v_i)$ based on Eq. (11). Finally, we compute $\max_{v_i \in G} \psi^\top(v_i)$ as a probabilistic upper bound of $LS_G(\Gamma_\sqcup)$.

Algorithm 2: Optimized Three-Phase Approach

Input : Privacy budget for phase 1 ϵ_1, privacy budget for phase 2 ϵ_2, invalidation probability δ, a large number h'

1 $\lambda_d = \frac{2}{0.5\epsilon_1}$; // Server
2 $\delta_1 = \delta$; // Server
3 **for** *each* v_i **do**
4 $d^\top(v_i) = d(v_i) + Lap(\lambda_d) + \lambda_d \cdot \log\left(\frac{1}{2\delta_1}\right)$; // Client
5 Report $d^\top(v_i)$ to server; // Client
6 Sort $\{v_i\}$ into $\{v_{[1]}, v_{[2]}, \ldots, v_{[n]}\}$ by $d^\top(v_i)$ in descending order; // Server
7 $\lambda_\psi = \frac{4\left(d^\top(v_{[1]}) + d^\top(v_{[2]})\right)}{0.5\epsilon_1}$; // Server
8 $\delta_2 = \delta$; // Server
9 **for** *each* v_i **do**
10 $\psi^\top(v_i) = \psi(v_i) + Lap(\lambda_\psi) + \lambda_\psi \cdot \log\left(\frac{1}{2\delta_2}\right)$; // Client
11 Report $\psi^\top(v_i)$ to server; // Client
12 $\lambda = \frac{1}{\epsilon_2} \max_{v_i \in G} \psi^\top(v_i)$; // Server
13 **return** λ // Server

Algorithm 2 shows the pseudo-code of the above method for computing λ. The following lemma establishes the privacy guarantee of the algorithm.

Lemma 4.3. *Algorithm 2 satisfies (ϵ_1, δ_1)-DDP and, with $1 - \delta_2$ probability, returns $\lambda \geq \frac{1}{\epsilon_2} LS_G(\Gamma_\sqcup)$.*

4.3 k-Cliques

Pessimistic Laplace mechanism. A k-clique refers to a set of k nodes that are fully connected to each other. Let $\gamma_{k\mathbb{C}}(v_i)$ be the number of k-cliques that user v_i appears in, and $\Gamma_{k\mathbb{C}} = \{\gamma_{k\mathbb{C}}(v_1), \ldots, \gamma_{k\mathbb{C}}(v_n)\}$. To obtain $\Gamma_{k\mathbb{C}}$ with ϵ-DDP, our based solution, namely Pessimistic Laplace mechanism, lets each user v_i inject Laplace noise $Lap\left(k\binom{n-2}{k-2}/\epsilon\right)$ into $\gamma_{k\mathbb{C}}(v_i)$, since the sensitivity of $\Gamma_{k\mathbb{C}}$ equals $k\binom{n-2}{k-2}$. This is because (i) adding or removing one edge e in G affects only those k-cliques where e is an edge, (ii) there are at most $\binom{n-2}{k-2}$ such k-cliques, and (iii) each k-clique is reported by k users.

Dataset	Num. of nodes	Num. of edges	Avg. deg.	Num. of triangles	Num. of three-hop paths	Num. of 4-cliques
Facebook	4,039	88,234	43.69	1,612,010	1,055,326,189	30,004,668
HepPh	12,008	118,489	19.73	3,358,499	3,146,167,903	150,281,372
AstroPh	18,771	198,050	21.10	1,351,441	986,743,120	9,580,415

Table 1: Dataset Properties

Proposed solution. Next we apply the proposed two-phase framework to obtain a more accurate estimate of the k-clique count. For this purpose, we present an ϵ_1-differentially private algorithm for computing a probabilistic upper bound of $LS_G(\Gamma_{k\mathbb{C}})$. First, we have

$$LS_G(\Gamma_{k\mathbb{C}}) = \max_{v_i, v_j \in G, i \neq j} k \cdot \mathbb{C}\left(G_{i\cap j}, k-2\right), \tag{13}$$

where $G_{i\cap j}$ denotes the subgraph of G induced the common neighbors of v_i and v_j, and $\mathbb{C}\left(G_{i\cap j}, k-2\right)$ denotes the number of $(k-2)$-cliques in $G_{i\cap j}$. To explain, observe that if an k-clique is affected by the presence or absence of an edge $\langle v_i, v_j\rangle$, then (i) the k-clique must contain both v_i and v_j, and (ii) apart from v_i and v_j, the remaining $k-2$ nodes in the clique must form a $(k-2)$-clique. There exist only $\mathbb{C}\left(G_{i\cap j}, k-2\right)$ such k-cliques, and each of them is reported by k users. Therefore, Eq. (13) holds.

Let $z(v_i) = \max_{v_j \in G, i\neq j} k\mathbb{C}\left(G_{i\cap j}, k-2\right)$. By Eq. (13), if we can derive a probabilistic upper bound $z^\top(v_i)$ of each $z(v_i)$, then we may use $\max_{v_i\in G} z^\top(v_i)$ as an upper bound of $LS_G(\Gamma_{k\mathbb{C}})$. By Lemma 4.1, we can compute $z^\top(v_i)$ as

$$z^\top(v_i) = z^*(v_i) + \lambda_z \cdot \log\left(\frac{1}{2\delta_z}\right),$$

where λ_z and δ_z are constants and $z^*(v_i)$ is obtained by injecting Laplace noise $Lap(\lambda_z)$ into $z(v_i)$. To ensure ϵ-DDP, however, we need $\lambda_z \geq kn\left(\binom{n-2}{k-2} - \binom{n-3}{k-2}\right)/\epsilon$, since adding or removing one edge in G may change each $z(v_i)$ by up to $k\left(\binom{n-2}{k-2} - \binom{n-3}{k-2}\right)$. This leads to an enormous amount of noise in $z^\top(v_i)$.

To address the above issue, we recursively apply our two-phase framework to derive an alternative upper bound of $LS_G(\Gamma_{k\mathbb{C}})$, in a manner similar to the method illustrated in Figure 3b. First, let $c(v_i)$ be as defined in Eq. (8). Then, we have

$$LS_G(\Gamma_{k\mathbb{C}}) = \max_{v_i, v_j \in G, i\neq j} k \cdot \mathbb{C}\left(G_{i\cap j}, k-2\right) \leq \max_{v_i \in G} k\binom{c(v_i)}{k-2}.$$

Therefore, if we are able to derive an upper bound $\max_{v_i\in G} c^\top(v_i)$ of $\max_{v_i\in G} c(v_i)$, then we can use $\binom{\max_{v_i\in G} c^\top(v_i)}{k-2}$ as an upper bound of $LS_G(\Gamma_{k\mathbb{C}})$. We compute such an upper bound $\max_{v_i\in G} c^\top(v_i)$ using the same method described in Section 4.1. That is, we compute $c^\top(v)$ for a selected set S of users v, as well as a probability degree upper bound $d^\top(v_j)$ for each user v_j. After that, we combine $d^\top(v_1), \ldots, d^\top(v_n)$ and $c^\top(v)$ $(v \in S)$ to derive an upper bound of $\max_{v_i\in G} c(v_i)$.

For brevity, we omit the pseudo-code as it can be easily constructed from the $\lambda_\triangle$ returned by Algorithm 1. That is, after getting $\lambda_\triangle$ from Algorithm 1, let

$$\lambda_{k\mathbb{C}} = k\binom{\frac{\epsilon_2}{3}\lambda_\triangle}{k-2}/\epsilon_2. \tag{14}$$

Since we have

$$\frac{\epsilon_2}{3}\lambda_\triangle = \max\{d^\top(v_{\lfloor h'+2\rfloor}), \max_{v_i\in S} c^\dagger(v_i) \geq \max_{v_i\in G} c(v_i),$$

with probability $1-\delta$, and $\lambda_\triangle$ satisfies ϵ_1-DDP. So $\lambda_{k\mathbb{C}}$ satisfies ϵ_1-DDP and we have $LS_G(\Gamma_{k\mathbb{C}}) \leq k\binom{\frac{\epsilon_2}{3}\lambda_\triangle}{k-2}$ with probability $1-\delta$.

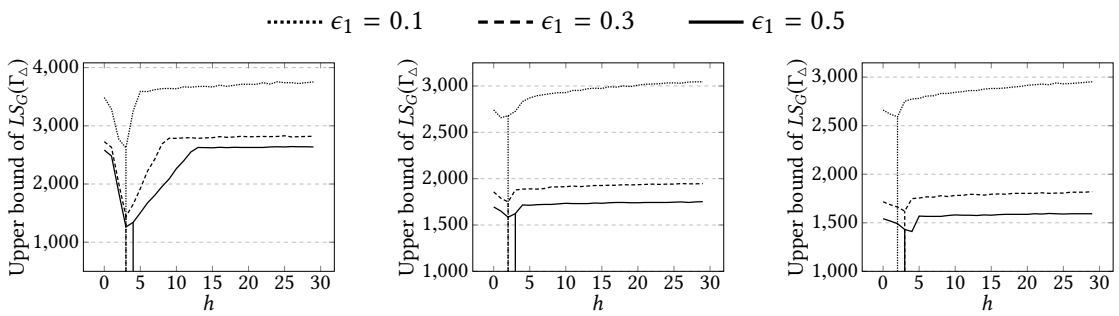
5 EXPERIMENTS

Datasets. We conduct experiments on three real world datasets from *Stanford Large Network Dataset Collection* [26]. The *Facebook dataset* contains "friends list" from Facebook, a large social network. It was collected from surveying participants using the Facebook app. The *HepPh dataset* and *AstroPh dataset* are the co-authorship networks from arXiv, which contains the collaborations between authors who submit their papers to High Energy Physics and Astro Physics, respectively. Table 1 shows the properties of the datasets and their true subgraph pattern counts.

Parameter selection. Since our solutions follow the proposed multi-phase framework, we need to split (ϵ, δ) budget among different phases. Recall from Section 3 that in the two-phase framework, Phase 1 estimates an upper bound of the required noise scale, and Phase 2 reports noisy counts. In order to get more accurate counting results, we assign more privacy budget to Phase 2. Specifically, we set $\epsilon_1 = 0.1\epsilon$ and $\epsilon_2 = 0.9\epsilon$ for Phase 1 and Phase 2, respectively. Regarding the other privacy parameter δ, following the popular guideline in [12], we set δ to $\frac{1}{n}$, where n is the number of nodes in the social network. Inside the proposed solutions described in Section 3, δ is sub-divided, e.g., by the number of entities to be protected for triangle counting.

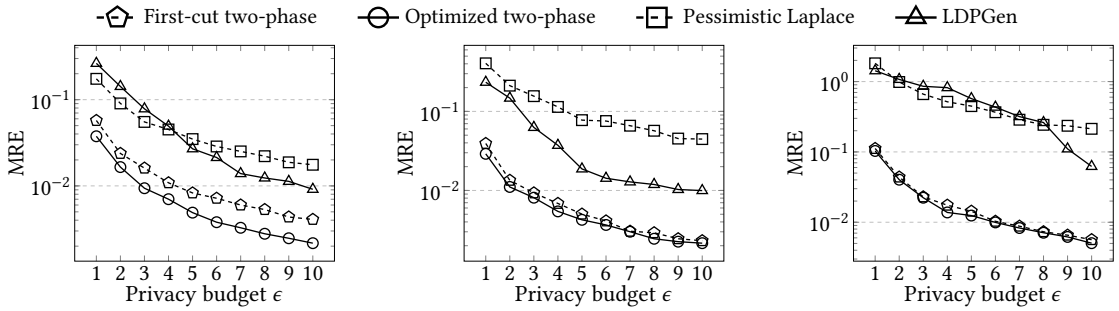
Baselines. We compare the fully optimized versions of proposed solutions for private subgraph counting, denoted by $\mathcal{M}_o$ in the following, against the following baselines: (i) the (ϵ, δ)-DDP version of the baseline method, i.e., Pessimistic Laplace mechanism, described in Section 3, (ii) first-cut versions of the proposed solutions, denoted as $\mathcal{M}_c$ in the following, and (ii) LDPGen [32], which generates synthetic graphs under LDP. The counts of differential private subgraph patterns are computed directly from the synthetic graph. Note that in LDPGen, the data collector only gathers information from the nodes on their direct connections; that is, no ELV is involved in this method, in which case DDP (Definition 2.6) reduces to LDP (Definition 2.4). Besides, we have also run experiments using the Pessimistic Gaussian mechanism, described towards the end of Section 3.1, which injects Gaussian noise into the true counts [1] instead of Laplace noise. From our experiments, we found that Pessimistic Gaussian consistently outputs much noisier results compared to the Pessimistic Laplace mechanism under (ϵ, δ)-DDP. We thus omit the results for Pessimistic Gaussian for brevity. Similarly, we omit Pessimistic Laplace under the stricter ϵ-DDP requirement, which is always worse than the more relaxed (ϵ, δ)-DDP version.

Noise scale selection. Before presenting our main evaluation results, we first demonstrate the effectiveness of the heuristic in Section 4.1 for determining the value of h in Phase 1, which is the



(a) Facebook dataset (b) HepPh dataset (c) AstroPh dataset

Figure 4: Selection of h for triangle counting



(a) Facebook dataset (b) HepPh dataset (c) AstroPh dataset

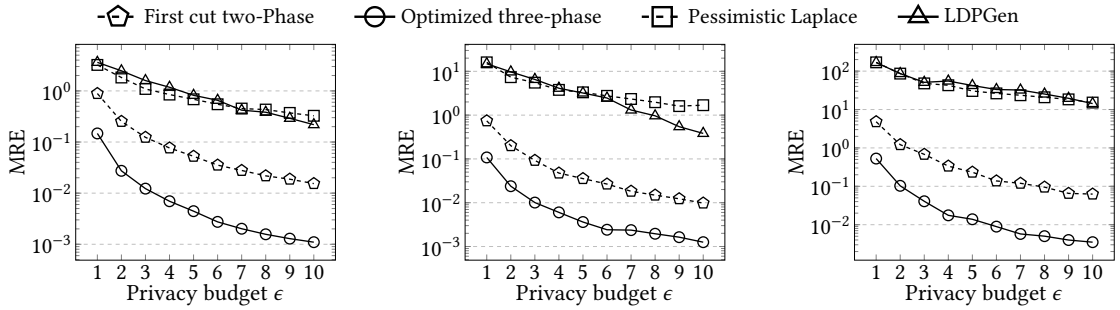
Figure 5: Triangle Counting

number of nodes participating in the second data collection step in the proposed improved solution. Figure 4 depicts the upper bound of $LS_G(\Gamma_\triangle)$, i.e., $\epsilon_1\lambda$ from Phase 1 as a function of h. For each of the datasets, for better presentation, the figure only shows three curves for three values of $\epsilon_1 = \{0.1, 0.3, 0.5\}$, corresponding to the total budget $\epsilon = \{1, 3, 5\}$. The valleys represent the optimal value of h, i.e., with the lowest $\epsilon_1\lambda$, while the vertical lines represent the heuristic value of h as computed in Lines 7-10 in Algorithm 1. The figure confirms that the heuristic values of h are close to the optimal ones. For example, on the Facebook dataset, the optimal and heuristic values are exactly the same for $\epsilon_1 = 0.1$ and $\epsilon_1 = 0.3$, while in the case of $\epsilon_1 = 0.5$, the heuristic deviates by 1 from the optimal value, which results in only 6% increase of the upper bound compared to its optimal value. The figure also shows that the noise scale varies across the datasets, with Facebook being the lowest (e.g., $\epsilon_1\lambda = 1443$ for $\epsilon_1 = 0.3$) and HepPh being the highest (e.g., $\epsilon_1\lambda = 1749$ for $\epsilon_1 = 0.3$). This is simply because the local sensitivity depends on the dataset properties, i.e., number of nodes, node degrees, node degree distribution.

Evaluation metric. We evaluate the accuracy of our approach in counting subgraph patterns (triangle, 3-hop path, and k-clique) on all the aforementioned datasets and compare it with that of the baselines. The accuracy of each method ($\mathcal{M}$) on graph (G) is measured by the mean relative error (*MRE*), that is, $MRE(\mathcal{M}, G) = |\mathcal{M}(G) - f(G)|/f(G)$, where $\mathcal{M}(G)$ is the differential private subgraph pattern count in input graph G, and $f(G)$ is the true subgraph count in G. Each result reported is averaged over 300 repeated runs.

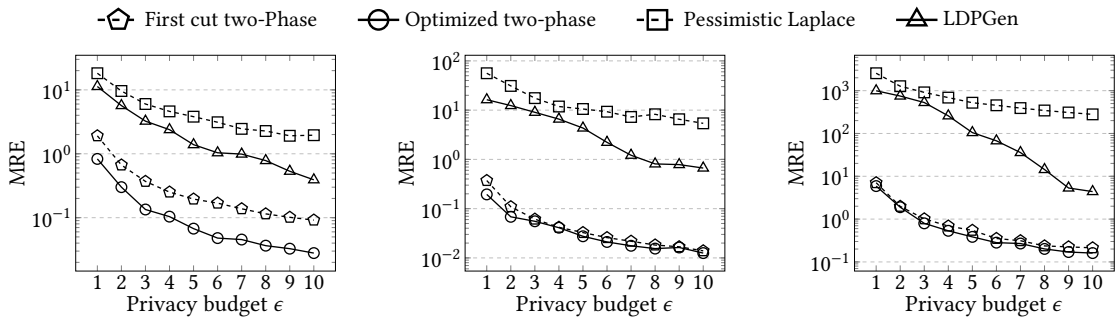
Triangle counts. The *MRE* of triangle counts of all the methods on all the datasets are depicted on Figure 5 while the privacy budget (ϵ) varies from 1 to 10. The results show that our optimized approach achieves good accuracy over all datasets. The proposed solution $\mathcal{M}_o(\triangle)$ clearly outperforms all the other differential private methods in terms of result accuracy. Note that the difference is significant since *MRE* is plotted in log-scale. In the Facebook dataset, for example, when privacy budget is relatively large, e.g., $\epsilon = 5$, its *MRE* always stays below or close to 0.49%. When ϵ decreases, the accuracy drops but it is still smaller than 3.8% even when $\epsilon = 1$. The result of the first-cut solution $\mathcal{M}_c(\triangle)$ is close to $\mathcal{M}_o(\triangle)$ in the case of HepPh and AstroPh datasets, with $\mathcal{M}_o(\triangle)$ strictly better. However, $\mathcal{M}_o(\triangle)$ significantly enhances the accuracy compared to $\mathcal{M}_c(\triangle)$ for the Facebook dataset.

3-hop counts. Figure 6 shows the results for the 3-hop counts. Again, the figure shows that our improved approach achieves good accuracy over all datasets. $\mathcal{M}_o(\sqcup)$ clearly outperforms all the other differential private methods including $\mathcal{M}_c(\sqcup)$, simply because it injects less noise into the true results. The general trend in the results of 3-hop counts is the same as that of triangle counts across datasets and differential private approaches. However, we first note that, even though the number of 3-hop counts is larger than that of triangles, triangle counts are more accurate than those of 3-hop counts. This is because the local sensitivity of triangle counts is much smaller than that of 3-hop counts. Second, we note that the difference between $\mathcal{M}_o(\sqcup)$ and $\mathcal{M}_c(\sqcup)$ is bigger than the difference between $\mathcal{M}_o(\triangle)$ and $\mathcal{M}_c(\triangle)$. This is because $\mathcal{M}_o(\sqcup)$ uses a much tighter bound for the noise scale compared to $\mathcal{M}_c(\sqcup)$, while the noise scale in $\mathcal{M}_o(\triangle)$ is not as much tight compared to $\mathcal{M}_c(\triangle)$.



(a) Facebook dataset (b) HepPh dataset (c) AstroPh dataset

Figure 6: 3-hop-path Counting



(a) Facebook dataset (b) HepPh dataset (c) AstroPh dataset

Figure 7: 4-Clique Counting

Finally, we note that the drop in *MRE* as ϵ increases is faster in the case of 3-hop counts compared to triangle counts for both $\mathcal{M}_o$ and $\mathcal{M}_c$. For example, in the Facebook dataset, the *MRE* of $\mathcal{M}_o(\sqcup)$ drops from 14.7% when $\epsilon = 1$ to 0.44% when $\epsilon = 5$.

4-clique counts. Figure 7 shows the results for the 4-clique counts. k-clique counting for a larger $k > 4$ would be expensive to evaluate due to the high time complexity of counting such cliques. Our proposed approach $\mathcal{M}_o(4\mathbb{C})$ once again achieves good accuracy over all datasets, and consistently outperforms its competitors. The general trend in the results of 4-clique counts is the same as that of triangle counts across datasets and differential private approaches. However, triangle counts are much more accurate than 4-clique counts due to the larger scale of 4-cliques in Eq. (14) compared to triangles. We also note that LDPGen is strictly better than *Pessimistic Laplace* for all privacy budgets on all the datasets.

6 RELATED WORK

Differential privacy [11] has attracted widespread attention from both academia and industry in the last decade. There are two existing models of differential privacy: centralized differential privacy (CDP) and local differential privacy (LDP). In CDP, data from individuals are collected and maintained by a trusted centralized data curator. The trusted curator executes a DP mechanism on the sensitive data and releases outputs, e.g., to untrusted data analysts. In LDP, there is no trusted centralized data curator. Rather, each individual perturbs its own data using a (local) differentially private algorithm. The data analyst collects these perturbed data, and uses it to infer aggregate statistics of the datasets. In a broader sense, this work falls into the category of differential privacy in the local setting [10], and our privacy definition (i.e., DDP) generalizes existing LDP definitions by considering extended local views.

Graph analysis with CDP. Graph analysis under the CDP setting has been studied intensively in the literature. Two different CDP models were defined for graph analysis: *edge differential privacy* and *node differential privacy*. Edge differential privacy considers two graphs as neighbors if they differ in one edge, while node differential privacy considers two graphs as neighbors if one can be obtained from the other by deleting a node and its adjacent edges. Edge differentially private algorithms have focused on releasing various types of graph statistics, including degree distributions [19, 22, 28], cuts [3, 17, 18], degree sequences [19, 22], k-stars and k-triangles [29], and subgraph counts [4, 21, 34, 42]. Triangle counting queries can be answered with edge differential privacy by efficiently computing the smooth sensitivity [30], empirical sensitivity [6], and ladder functions [42].

Earlier works on graph analysis under node differential privacy include [4, 6, 24]. Gehrke et al. [16] defined a generalization of differential privacy, called zero-knowledge privacy, that enforces node differentially privacy for bounded-degree graphs. [5, 33] focus on high-dimension graph data release with node differential privacy. Day et al. [8] investigated graph data publishing under node-differential privacy. Continuous release of graph statistics (e.g., degree distributions and subgraph counts) with node differential privacy has been initiated in [36]. All these CDP solution (for both edge and node differential privacy) require that the data

publisher has the full knowledge of the whole input graph. Therefore, they are not applicable to our setting, where the social graph is decentralized and no party knows the full graph.

Graph analysis with LDP. The LDP notion [23] assumes there is no trusted centralized data curator. Randomized response [41] is one of the simplest LDP techniques. However, directly applying the randomized response method on the local graph information (e.g., neighbor lists) collected from individual users may ruin the property (e.g., sparsity) of the original graph [32]. Gao et al. [15] transform the local graphs into neighbor lists and apply the hierarchical random graph (HRG) approach to add noise on the neighbor lists. Qin et al. [32] design LDPGen, a multi-phase technique that generates representative synthetic decentralized social graphs with local differential privacy. The synthetic graphs can be used for various graph analysis, such as graph modularity, clustering coefficient and assortativity coefficient. To our best knowledge, ours is the first work that considers subgraph counts in decentralized social networks with differential privacy guarantees, and the first to consider the case where each node possesses an extended local view with information beyond direct connections.

LDP for other types of data analysis. Finally, beyond social graph analysis, LDP algorithms for a variety of tasks have been widely investigated recently. Examples include frequency estimation [13, 39], heavy hitters [2, 14, 31], frequent itemsets [40], and marginal tables [7, 43]. The LDP model has been applied to the collection of various data types, including location [9] and positioning data [25], responses from crowdsourcing workers [27, 35, 37], and user data on mobile devices [38].

7 CONCLUSION

Given that more and more data are generated in the context of social networks and the well-spread concern of privacy, the problem of decentralized social network analysis would become increasingly important and relevant in practice. Our work is the one of first efforts toward develop privacy-preserving techniques to address the problem. With the proposed concept of decentralized differential privacy, our framework could be extended in multiple directions. For one, diverse local view models could be further considered. For example, in some social networks, though one cannot see all his two hop neighbors, the connections between her one-hop neighbors would be visible. How to accurately estimate relevant graph properties with such local views under DDP would be interesting future work. Another important direction is to integrate our framework with specific social network applications and carry out more sophisticated graph analysis tasks (e.g., community discovery, social graph recommendation).

ACKNOWLEDGMENTS

This publication was made possible by NPRP grant NPRP10-0208-170408 from the Qatar National Research Fund (a member of Qatar Foundation), and by the National Science Foundation (NSF), USA under grant No. 1350324. The findings herein reflect the work, and are solely the responsibility, of the authors. This work was also supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability, and by the Research Centres Funding Initiative, Provincial Key Research and Development Program of Zhejiang (Grant No. 2019C03133) and Major Scientific Research Project of Zhejiang Lab (Grant No. 2018FD0ZX01).

REFERENCES

[1] Borja Balle and Yu-Xiang Wang. 2018. Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising. In *International Conference on Machine Learning*. 403–412.
[2] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 127–135.
[3] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2012. The johnson-lindenstrauss transform itself preserves differential privacy. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. 410–419.
[4] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 87–96.
[5] Christian Borgs, Jennifer Chayes, and Adam Smith. 2015. Private graphon estimation for sparse graphs. In *Advances in Neural Information Processing Systems*. 1369–1377.
[6] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 653–664.
[7] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*. 131–146.
[8] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*. 123–138.
[9] Rinku Dewri. 2013. Local differential perturbations: Location privacy under approximate knowledge attackers. *IEEE Transactions on Mobile Computing* 12, 12 (2013), 2360–2372.
[10] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Statistical Minimax Rates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. 429–438. https://doi.org/10.1109/FOCS.2013.53
[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.
[12] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (Aug. 2014), 211–407. https://doi.org/10.1561/0400000042
[13] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.
[14] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies* 2016, 3 (2016), 41–61.
[15] Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. 2018. Local Differential Privately Anonymizing Online Social Networks Under HRG-Based Model. *IEEE Transactions on Computational Social Systems* 5, 4 (2018), 1009–1020.
[16] Johannes Gehrke, Edward Lui, and Rafael Pass. 2011. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *Theory of Cryptography Conference*. 432–449.
[17] Anupam Gupta, Aaron Roth, and Jonathan Ullman. 2012. Iterative constructions and private data release. In *Theory of cryptography conference*. 339–356.
[18] Moritz Hardt and Guy N Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. 61–70.
[19] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*. 169–178.
[20] Michael Hay, Chao Li, Gerome Miklau, and David D. Jensen. 2009. Accurate Estimation of the Degree Distribution of Private Networks. In *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*. 169–178. https://doi.org/10.1109/ICDM.2009.11
[21] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.
[22] Vishesh Karwa and Aleksandra B Slavković. 2012. Differentially private graphical degree sequences and synthetic graphs. In *International Conference on Privacy in Statistical Databases*. 273–285.
[23] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[24] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*. Springer, 457–476.

[25] Jong Wook Kim, Dae-Ho Kim, and Beakcheol Jang. 2018. Application of local differential privacy to collection of indoor positioning data. *IEEE Access* 6 (2018), 4276–4286.

[26] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[27] Yaliang Li, Chenglin Miao, Lu Su, Jing Gao, Qi Li, Bolin Ding, Zhan Qin, and Kui Ren. 2018. An efficient two-layer mechanism for privacy-preserving truth discovery. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1705–1714.

[28] Bing-Rong Lin and Daniel Kifer. 2013. Information preservation in statistical privacy and bayesian estimation of unattributed histograms. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 677–688.

[29] Wentian Lu and Gerome Miklau. 2014. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 921–930.

[30] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[31] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 192–203.

[32] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 425–438.

[33] Sofya Raskhodnikova and Adam Smith. 2016. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. 495–504.

[34] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. 2009. Relationship privacy: output perturbation for queries with joins. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 107–116.

[35] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. 2018. LoPub: High-Dimensional Crowdsourced Data Publication With Local Differential Privacy. *IEEE Transactions on Information Forensics and Security* 13, 9 (2018), 2151–2166.

[36] Shuang Song, Susan Little, Sanjay Mehta, Staal Vinterbo, and Kamalika Chaudhuri. 2018. Differentially Private Continual Release of Graph Statistics. *arXiv preprint arXiv:1809.02575* (2018).

[37] Haipei Sun, Boxiang Dong, Hui Wendy Wang, Ting Yu, and Zhan Qin. 2018. Truth Inference on Sparse Crowdsourcing Data with Local Differential Privacy. In *2018 IEEE International Conference on Big Data (Big Data)*. 488–497.

[38] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. 2017. Privacy loss in Apple's implementation of differential privacy on MacOS 10.12. *arXiv preprint arXiv:1709.02753* (2017).

[39] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX) Security 17)*. 729–745.

[40] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*. 127–143.

[41] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.

[42] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 731–745.

[43] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 212–229.

APPENDIX

Proof of Lemma 3.2

Proof. Let $\gamma(v_i)^*$ and $\gamma'(v_i)^*$ be the noisy triangle counts of node v_i when its ELV is G_i and G_i', respectively, and s_i be any possible subset of possible noisy triangle count at v_i. Since each node independently perturbs its local triangle count, we have:

$$\begin{aligned}
&\frac{\Pr(\mathcal{M}_1(G_1)\in\mathcal{S}_1,\ldots,\mathcal{M}_n(G_n)\in\mathcal{S}_n)}{\Pr(\mathcal{M}_1(G_1')\in\mathcal{S}_1,\ldots,\mathcal{M}_n(G_n')\in\mathcal{S}_n)}\\
=&\frac{\prod_{i=1}^n\Pr[\gamma(v_i)^*=s_i]}{\prod_{i=1}^n\Pr[\gamma'(v_i)^*=s_i]}=\frac{\prod_{i=1}^n\exp\left(\frac{\epsilon}{\Delta(\Gamma_\triangle)}|s_i-\gamma(v_i)|\right)}{\prod_{i=1}^n\exp\left(\frac{\epsilon}{\Delta(\Gamma_\triangle)}|s_i-\gamma'(v_i)|\right)}\\
=&\prod_{i=1}^n\exp\left(\frac{\epsilon}{\Delta(\Gamma_\triangle)}\left[|s_i-\gamma(v_i)|-|s_i-\gamma'(v_i)|\right]\right)\\
\le&\exp\left(\frac{\epsilon}{\Delta(\Gamma_\triangle)}\sum_{i=1}^n|\gamma(v_i)-\gamma'(v_i)|\right)=\exp(\epsilon),
\end{aligned}$$

where $s_i\in range(\mathcal{M}_i)$ is any possible output of $\mathcal{M}_i$, for $1\le i\le n$. □

Proof of Lemma 3.3

Proof. For each user v_i, the variance is $\sigma_i^2=2\left(\frac{S(\Gamma_\triangle)}{\epsilon}\right)^2$. Thus the total variance is $\sigma^2=\sum_{i=1}^n\sigma_i^2=2n\left(\frac{S(\Gamma_\triangle)}{\epsilon}\right)^2=O\left(\frac{n^3}{\epsilon^2}\right)$. □

Proof of Lemma 4.1

Proof. Without loss of generality, we assume that $\delta\le 0.5$. Then, we have:

$$\begin{aligned}
&\Pr\left[x^*+\alpha\cdot\log\left(\frac{1}{2\delta}\right)\ge x\right]=\Pr\left[Lap(\alpha)+\alpha\cdot\log\left(\frac{1}{2\delta}\right)\ge 0\right]\\
=&\int_{\alpha\cdot\log(2\delta)}^{+\infty}Lap(t,\alpha)\,\mathrm{d}t=1-\int_{-\infty}^{\alpha\cdot\log(2\delta)}Lap(t,\alpha)\,\mathrm{d}t\\
=&1-\left(\frac{1}{2}\exp\left(\frac{\alpha\cdot\log(2\delta)}{\alpha}\right)-0\right)=1-\delta.
\end{aligned}$$

□

Proof of Lemma 4.2

Proof. First, we prove Algorithm 1 satisfies ϵ_1-DDP. Suppose that we have two graph G and G' with one edge difference. And we have $d(v_i)$, $d^\top(v_i)$, $c(v_i)$ and $c^\top(v_i)$ on G, and $d'(v_i)$, $d'^\top(v_i)$, $c'(v_i)$ and $c'^\top(v_i)$ on G'. Note that the change of edge between G and G' only results in the change of degree of maximum two clients v_x and v_y by 1. Thus, for Lines 3-5 we have:

$$\begin{aligned}
&\frac{\prod_{i=1}^n\Pr[d^\top(v_i)=a_i|G]}{\prod_{i=1}^n\Pr[d'^\top(v_i)=a_i|G']}\\
=&\frac{\Pr[d^\top(v_x)=a_x|G]\cdot\Pr[d^\top(v_y)=a_y|G]}{\Pr[d'^\top(v_x)=a_x|G]\cdot\Pr[d'^\top(v_y)=a_y|G]}\\
=&\frac{\exp\left(\frac{|a_x-d(v_x)-\lambda_d\cdot\log(2\delta')|+|a_y-d(v_y)-\lambda_d\cdot\log(2\delta')|}{\lambda_d}\right)}{\exp\left(\frac{|a_x-d'(v_x)-\lambda_d\cdot\log(2\delta')|+|a_y-d'(v_y)-\lambda_d\cdot\log(2\delta')|}{\lambda_d}\right)}\\
\le&\exp\left(\frac{1}{\lambda_d}|d(v_x)-d'(v_x)|+\frac{1}{\lambda_d}|d(v_y)-d'(v_y)|\right)\\
\le&\exp\left(\frac{2}{\lambda_d}\right)=\exp\left(\frac{\epsilon_1}{2}\right).
\end{aligned}$$

For Lines 13-16, since we only let h clients to calculate $c^\top(v_i)$, the change of $\{c^\top(v_i)\}$ will be at most h. Therefore, we have:

$$\begin{aligned}
&\frac{\prod_{v_i\in S}\Pr[c^\top(v_i)=a_i|G]}{\prod_{v_i\in S}\Pr[c'^\top(v_i)=a_i|G']}\\
=&\frac{\prod_{v_i\in S}\exp\left(\frac{1}{\lambda_c}|a_i-c(v_i)-\lambda_c\cdot\log(2\delta')|\right)}{\prod_{v_i\in S}\exp\left(\frac{1}{\lambda_c}|a_i-c'(v_i)-\lambda_c\cdot\log(2\delta')|\right)}\\
\le&\exp\left(\frac{1}{\lambda_c}\sum_{v_i\in S}|d(v_i)-d'(v_i)|\right)\\
\le&\exp\left(\frac{2}{\lambda_c}|S|\right)=\exp\left(\frac{\epsilon_1}{2}\right).
\end{aligned}$$

Note that Lines 13-16 use the result from Lines 3-5, and the final result λ uses the result from Lines 13-16. Due to the composition rule of differential privacy, the algorithm as a whole satisfies ϵ_1-DDP.

Second, we will prove $\lambda \ge \frac{1}{\epsilon_2}LS_G(\Gamma_\triangle)$ with $1-\delta$ probability. The first step that may fail is getting set S. In order to get a valid set S, we need to guarantee that all the $h+2$ clients $\{v_{[1]}, v_{[2]}, \ldots, v_{[h+2]}\}$ satisfy $\forall i \in [1, h+2], d^\top(v_{[i]}) \ge d(v_{(i)})$, where $v_{(i)}$ is the client who has the i-th largest *actual* degree.

Here, we define $\xi_i \in [0,1]$ as the sum of probabilities that another client other than $v_{(i)}$ becomes $v_{[i]}$ and meanwhile $d^\top(v_{[i]}) \ge d(v_{(i)})$. It is easy to infer that for any $i \in [1, h+2]$, we have:

$$\begin{aligned}
\Pr[d^\top(v_{[i]}) \ge d(v_{(i)})] &= \Pr[d^\top(v_{(i)}) \ge d(v_{(i)})] + \xi_i\\
&\ge \Pr[d^\top(v_{(i)}) \ge d(v_{(i)})]\\
&= 1-\delta'.
\end{aligned}$$

Therefore, we have $\Pr[d^\top(v_{[i]}) < d(v_{(i)})] < \delta'$ for $h'+2$ clients.

Then, for all the h clients in S, when calculating $c^\top(v_i)$, from Eq. (4.1) there is also a probability δ' to fail. Since $h \le h'$, we have $h'+2+h \le 2h'+2$ times to fail, every time with a probability $\delta' = \frac{\delta}{2h'+2}$. Due to the *union bound*, the total probability to fail is at most δ.

As we discussed previously, $max_{v_i\in G}c(v_i)$ is covered by either $max_{v_i\in G}\{c^\top(v_i)\}$ or $d^\top(v_{[h'+2]})$. Combined with Eq. (7), we have:

$$\begin{aligned}
\lambda &= 3\max\left\{\frac{1}{\epsilon_2}d^\top(v_{[h'+2]}), \frac{1}{\epsilon_2}\max_{v_i\in S}c^\dagger(v_i)\right\}\\
&\ge 3\max_{v_i\in G}c(v_i) = LS_G(\Gamma_\triangle)
\end{aligned}$$

with probability $1-\delta$. □

Proof of Lemma 4.3

Proof. First, we prove that Algorithm 2 satisfies (ϵ_1, δ_1)-DDP. Similar to our proof in Lemma 4.2, the procedure in Lines 3-5 satisfies $\frac{1}{2}\epsilon_1$-DDP, and with a probability $1-\delta_1$ with $d^\top(v_{[1]}) \ge d(v_{(1)}) \wedge d^\top(v_{[2]}) \ge d(v_{(2)})$. Then, unlike Algorithm 1, the second λ_ψ that we use here requires the output of Lines 3-5. Therefore, Lines 9-11 satisfy $\frac{1}{2}\epsilon_1$-DDP with probability $1-\delta_1$. Overall, the algorithm satisfies (ϵ_1, δ_1)-DDP.

Second, similar to the proof in Lemma 4.2, we have: $\Pr[\max_{v_i\in G}\psi^\top(v_i) \ge \max_{v_i\in G}\psi(v_i)] \ge 1-\delta_2$. This leads to

$$\lambda = \frac{1}{\epsilon_2}\max_{v_i\in G}\psi^\top(v_i) \ge \frac{1}{\epsilon_2}\max_{v_i\in G}\psi(v_i) \ge \frac{1}{\epsilon_2}LS_G(\Gamma_\sqcup)$$

with probability $1-\delta_2$. □