Privacy-preserving Triangle Counting in Directed Graphs

Ziyao Wei[1], Qing Liu[1], Zhikun Zhang[1], Yunjun Gao[1], Jianliang Xu[2]
[1]Zhejiang University, [2]Hong Kong Baptist University
{wei_zy,qingliucs,zhikun,gaoyj}@zju.edu.cn,xujl@comp.hkbu.edu.hk

ABSTRACT

In directed graphs, the relationship between users is asymmetric, resulting in two different types of triangles, i.e., *cycle* and *flow* triangles. This paper studies the problem of privacy preserving triangle counting in directed graphs. Based on different applications, we consider two scenarios, i.e., trusted and untrusted servers. In the literature, privacy preserving triangle counting in undirected graphs has been widely studied. However, directly applying these algorithms to address our problem suffers from many issues. Specifically, for the trusted server, the differentially private triangle counting algorithms, which are designed for undirected graphs, have suboptimal performance under the directed graphs. Hence, we propose a new centralized differential private releasing algorithm to add the Laplacian noise to the exact numbers via analyzing global sensitivity. In addition, for the untrusted server, the existing techniques can not be used to count cycle and flow triangles with differential privacy due to that the local view of each user in directed graphs is limited to out-neighbors rather than all neighbors. Therefore, we design a novel local differential private releasing algorithm to provide local unbiased estimation, which implies that after the aggregation of all the local estimations on the central server side, an unbiased estimation for the numbers of cycle and flow triangles is deduced. Moreover, the Laplacian noise is added to the local estimation on each user side through analyzing global sensitivity. Empirical experiments on six real-world graphs demonstrate that our proposed algorithms achieve high efficiency and utility.

PVLDB Reference Format:
Ziyao Wei[1], Qing Liu[1], Zhikun Zhang[1], Yunjun Gao[1], Jianliang Xu[2]. Privacy-preserving Triangle Counting in Directed Graphs. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/ZJU-DAILY/PrivTC.

1 INTRODUCTION

Directed graphs can effectively model asymmetric relationships between entities. For instance, in a social network, if user u follows user v but v does not follow u, a directed edge (u, v) is present, while the edge (v, u) is absent. Real-world directed graphs encompass a variety of domains, including social networks [10, 11, 21, 22, 37, 38, 41, 45], E-commerce networks [31, 39], protein-protein interactions [14, 27], and Very Large Scale Integration (VLSI) circuit [44].

Triangle is the smallest cohesive unit in graphs, and counting triangles for a given graph is a crucial task in graph mining and analysis [1, 12, 13, 15, 23]. In directed graphs, triangles, which are different from those in undirected graphs, include two distinct types, i.e., *cycle triangles* and *flow triangles*, as illustrated in Figure 1. Given a directed graph G, triangle counting aims to count the number of cycle triangles and flow triangles in G, which supports numerous applications, such as community search [22, 37] and clustering coefficient computation [10, 38]. However, disclosing the exact numbers of cycle and flow triangles in directed graphs might leak users' *private relationships*. For example, consider a social network in Figure 2, consisting of four users. Assume that each user's follow list is private and everyone is only aware of their own followees, which are referred to as *out-neighbors* in directed graphs. There are eight cases of v_2's out-neighbors, denoted as G_i $(i = 1, 2, ..., 8)$ in Figure 2. Each case corresponds to a unique number of cycle and flow triangles. If users v_1, v_3, and v_4 collude, and the exact number of cycle triangles and flow triangles is exposed, v_1, v_3, and v_4 can collectively infer the follow list of v_2. For example, when the released numbers of flow and cycle triangles are 1 and 2, v_1, v_3, and v_4 can easily infer that v_2's follow list is $\{v_1\}$, which corresponds to G_2. Therefore, it is essential to preserve the triangles' information in directed graphs. Motivated by it, for the first time, we study the problem of privacy-preserving triangle counting in directed graphs.

Edge Differential Privacy. To preserve the edge/relationship information in graphs, *edge differential privacy* (edge-DP) has been widely adopted [19, 42, 43]. The general idea of edge-DP is to guarantee that the impact of a *single edge* on the final output of a randomized graph analysis algorithm is limited. In this paper, we consider two scenarios when using edge-DP for triangle counting in directed graphs.

- The central server is *trusted* and has access to the whole graph. Its objective is to publish the triangle counting results to untrusted third parties for research purposes. For example, Facebook owns the follower/followee information of all its users and allows third-party researchers to query its social network. In this case, the central server can use *edge centralized differential privacy* (edge-CDP) to perturb the triangle-counting results to preserve the edge/relationship information of its users when answer third-party queries.
- The central server is *untrusted* and does not have access to the whole graph. The objective is to collect the triangle counting information from individual users. For example, AT&T owns a huge number of users but does not know the local contacts of their users. When the central server wants to know the number

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

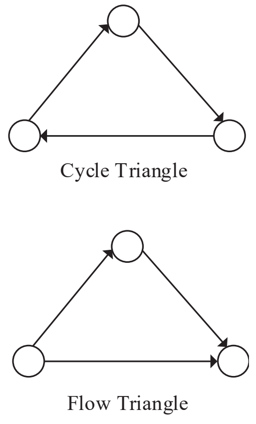
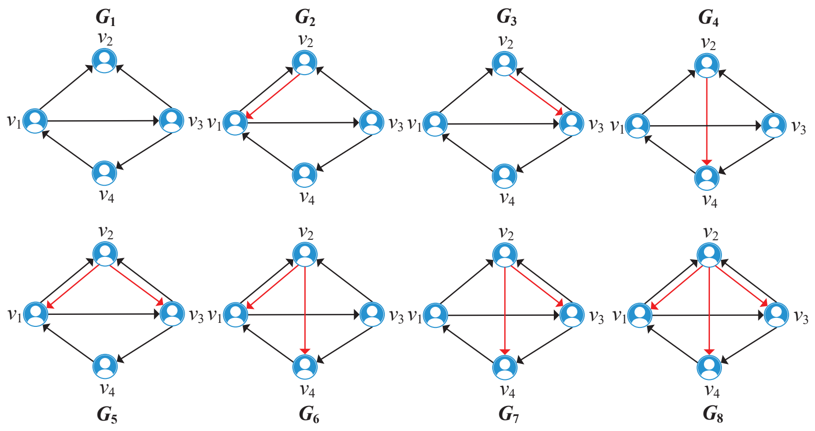


Figure 1: Triangles in Directed Graphs



G	# Cycle Triangles	# Flow Triangles
G_1	1	1
G_2	2	1
G_3	1	2
G_4	2	2
G_5	2	3
G_6	3	3
G_7	2	4
G_8	3	6

Figure 2: Directed Triangle Counting

of triangles of its user networks, it can allow individual users to use *edge local differential privacy* (edge-LDP) to perturb their private contact information and upload it to the server. After receiving the perturbed contacts, the central server can use some aggregation algorithms to extract the triangle counting results.

Centralized Solutions for Trusted Servers. In the literature, the triangle counting problem under edge-CDP for undirected graphs has been well studied [2, 3, 6, 29, 43]. A straightforward way for counting triangles in directed graphs is to adapt the triangle counting algorithms for undirected graphs to separately count cycle and flow triangles. In this way, the total privacy budget should be divided into two parts: one for counting cycle triangles, and the other for counting flow triangles. However, it is hard to select an appropriate ratio of these two privacy budgets to achieve the optimal utility, resulting in the waste of privacy budget. To address this issue, we propose a new centralized approach to count the cycle and flow triangles simultaneously. Specifically, we first employ graph projection technique to obtain a *projected-directed graph* for the given directed graph. Then, the numbers of cycle triangles and flow triangles are counted in the projected-directed graph. Finally, the counting results are adjusted via *Laplacian noise*, whose parameter is determined by the global sensitivity and the total privacy budget. In our proposed approach, the privacy budget does not need to be divided, thereby avoiding the waste of privacy budget. The main challenge of this approach is *how to analyze the global sensitivity of the counting function*. To facilitate the global sensitivity analysis, we construct a *case-directed graph*, based on which we can infer the largest difference of the total numbers of cycle and flow triangles by adding/removing a particular edge.

Local Solutions for Untrusted Servers. Existing studies [9, 16, 17] have proposed numerous algorithms for triangle counting under edge-LDP in undirected graphs. However, in undirected graphs, a vertex is aware of all its neighbors, whereas in directed graphs, a vertex is only aware its *out-neighbors*. The difference in local view makes existing algorithms impossible to solve our problem. It is because the triangle counting algorithms of undirected graphs can not calculate the local estimation, which is computed based on the local view of each user and reported to the central server. Therefore, we design a new local algorithm to count cycle and flow triangles for untrusted servers. Our proposed algorithm consists of three steps. First, we generate the *noisy graph* by using randomized response [40]. Second, each user downloads the noisy graph from the central server and computes the local estimation of the numbers of cycle and flow triangles by adding the *Laplacian noise*. Then, each user reports the local estimation of the numbers of cycle and flow triangles to the central server. Third, the central server aggregates the local estimations of all users. For the local algorithm, the most important issue is *how does each user calculate useful local estimation of the numbers of cycle and flow triangles*. To tackle this issue, we design a novel *local unbiased estimation*, which is calculated by the numbers of local subgraphs. Based on local unbiased estimation, we can obtain an unbiased estimation of the numbers of cycle triangles and flow triangles in input-directed graph on the central server side.

In summary, we make the following contributions.

- To the best of our knowledge, we are the first to study triangle counting problems in directed graphs with differential privacy.
- We introduce the centralized differential private releasing algorithm. To ensure the utility in theoretical aspect, we prove that the output of the algorithm is an unbiased estimation and analyze the variance of the output of the algorithm.
- We introduce the local differential private releasing algorithm. We also prove that the algorithmic output is an unbiased estimation and analyze the upper bound of the variance of the algorithmic output.
- We conduct comprehensive experiments across six real-world datasets to demonstrate the high efficiency and utility of proposed approaches.

Roadmap. We review the related works in Section 2. Section 3 introduces the preliminaries. Section 4 proposes the centralized differential private releasing algorithms for triangle counting in directed graphs. Section 5 proposes the local differential private releasing algorithms for triangle counting in directed graphs. Experiments are presented in Section 6. Finally, we conclude this paper in Section 7.

2 RELATED WORK

Triangle Counting with Differential Privacy. By now, many efforts have been devoted to the study of triangle counting with differential privacy. The existing algorithms can be divided into centralized and non-centralized differential privacy.

For the centralized differential privacy, the straightforward approach for triangle counting is to add the Laplacian noise to the counting function [6]. To reduce the sensitivity, Nissim et al. [29] and Blocki et al. [2] propose the smooth sensitivity and restricted sensitivity, respectively. Avoiding the Laplacian mechanism, Zhang et al. [43] propose a ladder framework and apply their framework for subgraph counting problems including triangle counting. With a relax version of edge differential privacy, Rastogi et al. [33] propose a general algorithm for releasing the count of specified subgraphs including triangles. Karwa et al. [19] further improve the algorithm in [33], and the proposed algorithm satisfies a stronger notation of privacy. For the exponential random graph, Lu and Miklau [26] propose algorithms for estimating the alternating k-triangle [34]. Under node differential privacy, Kasiviswanathan et al. [20] and Ding et al. [4, 5] propose triangle counting algorithms based on linear programming and a projection method, respectively. Besides, Chen and Zhou [3] present a solution to subgraph counting for any subgraphs including triangles, which could satisfy either edge differential privacy or node differential privacy.

For the non-centralized differential privacy, Imola et al. [16] present two algorithms with local differential privacy for triangle counting, one is non-interactive, and the other is interactive. Then, following this work, Imola et al. [17] address the drawback of the interactive triangle counting algorithm of high communication cost. Theoretically, Eden et al. [9] prove the lower bounds of the additive errors of triangle counting algorithms with local differential privacy, including both non-interactive and interactive. Sun et al. [35] propose the decentralized differential privacy and design a framework which can calculate the numbers of subgraphs including triangles. Liu et al. [25] propose the edge relationship differential privacy and a two-phase framework for triangle counting. Imola et al. [18] propose a wedge shuffling technique and applies it to the triangle counting. Liu et al. [24] propose a crypto-assisted differential private triangle counting system, named CARGO.

Note that, existing techniques are proposed for undirtected graphs. In this paper, we consider triangle counting with differential privacy in directed graphs.

Triangle-based Directed Graphs Analysis. The cycle and flow triangles are widely used for directed graph analysis. Takaguchi and Yoshida [36] employ cycle triangle and flow triangle to design cycle truss and flow truss, respectively, which can be used to discover subgraphs with different structures. Then, Liu et al. [22] consider cycle and flow triangles simultaneously and propose D-truss, which requires that each edge should be contained in k_c cycle triangles and k_f flow triangles. D-truss can be effectively used for community search. Fagiolo [10] propose directed clustering coefficient, which inclueds flow (transitive) clustering coefficient and cycle (cyclic) clustering coefficient, to cluster diercted graphs. Trolliet et al. [38] extend the direct clustering coefficients to design an interest clustering coefficient to measure the clustering of directed social graphs with interest links. Moreover, Parente and Colosimo [30] employ cycle and flow triangles to estimate the influence of the network structure on dynamical processes, which is used to model the multiplex brain network. However, these works ignore the privacy issue, which is the focus of this paper.

3 PRELIMINARIES

In this section, we primarily introduce the key concepts of edge-CDP and edge-LDP over directed graphs. We consider a directed graph $G = (V, E)$, where V and E denote the sets of edges and vertices, respectively.

3.1 ϵ-edge-CDP over Directed Graphs

Centralized differential privacy assumes the central server is trustworthy, and aims to protect the user privacy by perturbing the output. It guarantees that for any two neighboring datasets differing by one data record, the outputs of algorithm have indistinguishable distribution. Before formally defining edge-CDP over directed graphs, we give the definition of *neighboring directed graphs*.

Definition 3.1 (Neighboring Directed Graphs). *Given two directed graphs $G = (V, E)$ and $G' = (V', E')$, G and G' are neighbors if (1) $V = V'$, and (2) $|E - E'| = 0, |E' - E| = 1$ or $|E - E'| = 1, |E' - E| = 0$.*

In other words, the neighboring directed graphs are two directed graphs that share the same vertex set but differ by one edge. Based on it, we formally define *ϵ-edge centralized differential privacy* (ϵ-edge-CDP) over directed graphs.

Definition 3.2 (ϵ-edge-CDP over Directed Graphs). *A randomized algorithm $\mathcal{A} : \mathcal{G} \to \mathcal{R}$ satisfies ϵ-edge-CDP if for any pair of neighboring directed graphs, $G, G^{'} \in \mathcal{G}$, and for any subset of possible outputs $O \subseteq \mathcal{R}$, it holds that*

$$\Pr[\mathcal{A}(G) \in O] \le e^{\epsilon}\Pr[\mathcal{A}(G^{'}) \in O].$$

ϵ-edge-CDP requires that for any pair of two neighboring directed graphs G and $G^{'}$, the randomized algorithm $\mathcal{A}$ has similar output distributions. The similarity of $\mathcal{A}(G)$ and $\mathcal{A}(G^{'})$ can be adjusted by the privacy budget ϵ. The smaller ϵ, the more similar of $\mathcal{A}(G)$ and $\mathcal{A}(G^{'})$, leading to stronger privacy protection.

3.2 ϵ-edge-LDP over Directed Graphs

The local differential privacy assumes the central server is untrustworthy. It allows the data owners to perturb their private data locally and upload the perturbed data to the server. Different from the edge-CDP that defines on two neighboring directed graphs, the edge-LDP over directed graphs is based on the *neighboring out-neighbor lists.*

Definition 3.3 (Neighboring Out-neighbor Lists). *For a directed graph $G = (V, E)$ and a vertex $v_i \in V$, let $\lambda_i \in \{0, 1\}^n$ be the list of v_i's out-neighbors. Given another list $\lambda_i' \in \{0, 1\}^n$, λ_i and λ_i' are neighboring out-neighbor lists if $\|\lambda_i - \lambda_i'\|_1 = 1$.*

Exactly, two out-neighbor lists are neighboring if they differ in one bit. Based on Definition 3.3, we formally define the *ϵ-edge local differential privacy* (ϵ-edge-LDP) over directed graphs in the following.

DEFINITION 3.4 (ϵ-EDGE-LDP). *A randomized algorithm $\mathcal{A} : \{0,1\}^n \rightarrow \mathcal{R}$ satisfies ϵ-edge-LDP if $\forall i \in [n]$, for any pair of out-neighbor lists $\lambda_i, \lambda_i^{'} \in \{0,1\}^n$, and for any subset of possible outputs $O \subseteq \mathcal{R}$, it holds that*

$$\Pr[\mathcal{A}(\lambda_i) \in O] \leq e^{\epsilon}\Pr[\mathcal{A}(\lambda_i^{'}) \in O].$$

3.3 Implementation Mechanism and Properties of Differential Privacy

Laplacian Mechanism. It is the most commonly used mechanism to implement differential privacy. Given a data analysis function f, the Laplacian mechanism adds Laplacian noise to f, thereby achieving differential privacy. The parameter of the Laplacian noise is given by the global sensitivity GS_f of the function f and the privacy budget ϵ.

DEFINITION 3.5 (GLOBAL SENSITIVITY [7]). *Given a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the global sensitivity of f is defined as,*

$$GS_f = \max_{D,D^{'} \in \mathcal{D}: D \sim D^{'}} \|f(D) - f(D^{'})\|_1,$$

where $D \sim D^{'}$ denotes that D and $D^{'}$ are two neighboring datasets.

THEOREM 3.1 (LAPLACIAN MECHANISM [8]). *Given a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the Laplacian mechanism is defined as: $M_L(D, f, \epsilon) = f(D) + (X_1, X_2, \cdots, X_d)$, where X_i i.i.d. follows a Laplacian distribution $\mathrm{Lap}(\frac{GS_f}{\epsilon})$. For a parameter b, the Laplacian distribution has the density function $\mathrm{Lap}(b)(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$. The Laplacian mechanism preserves ϵ-differential privacy.*

Randomized Response. The randomized response [40] is used to implement local differential privacy. Qin et al. [32] prove that applying randomized responses to neighbor lists of each user provides ϵ-edge-LDP.

THEOREM 3.2. *Given a graph G and a neighbor list $\lambda_i \in \{0,1\}^n$ of a vertex, if a randomized algorithm $\mathcal{A}$ outputs a noisy neighbor list $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n) \in \{0,1\}^n$ such that $\forall j \in [n]$, $\alpha_j \neq \lambda_{i,j}$ with probability $p = \frac{1}{e^{\epsilon}+1}$, $\mathcal{A}$ provides ϵ-edge-LDP.*

It is easy to verify that for the out-neighbor list of a vertex in directed graphs, Theorem 3.2 also holds.

Post-Processing and Composition. Differential privacy has useful properties. The post-processing theorem ensures that the output can be processed while maintaining differential privacy. The sequential composition theorem ensures that the sequential composition of differential privacy algorithms also satisfies differential privacy.

THEOREM 3.3 (POST-PROCESSING THEOREM [8]). *Let $\mathcal{A}$ be an algorithm and f be an arbitrary randomized mapping. If $\mathcal{A}$ satisfies ϵ-differential privacy, $f \circ \mathcal{A}$ also satisfies ϵ-differential privacy.*

THEOREM 3.4 (SEQUENTIAL COMPOSITION THEOREM [17]). *Let $\mathcal{A}_1$ be an algorithm satisfying ϵ_1-differential privacy; M be the output of $\mathcal{A}_1$; $\mathcal{A}_2(M)$ be an algorithm satisfying ϵ_2-differential privacy. Then, the sequential composition $(\mathcal{A}_1, \mathcal{A}_2(M))$ satisfies $(\epsilon_1 + \epsilon_2)$-differential privacy.*

4 DIRECTED TRIANGLE COUNTING WITH EDGE-CDP

In this section, we firstly present the problem of triangle counting with edge-CDP in directed graphs. Then, we propose the centralized algorithm to address the problem.

4.1 Problem Statement and Strawman Solution

PROBLEM 1. *Given a directed graph $G = (V, E)$, where all vertices are public and all edges are private, and a privacy budget ϵ, we aim to release the numbers of cycle triangles $f_{c\triangle}(G)$ and flow triangles $f_{f\triangle}(G)$ in G while satisfying ϵ-edge-CDP.*

Strawman Solution. The basic approach to address Problem 1 is to employ the algorithms for undirected graphs to count cycle triangles and flow triangles separately. To this end, we should divide the total privacy budget ϵ into two parts: ϵ_1 and ϵ_2. Then, the noises $\mathrm{Lap}(\frac{GS_{f_{c\triangle}}}{\epsilon_1})$ and $\mathrm{Lap}(\frac{GS_{f_{f\triangle}}}{\epsilon_2})$ are added to the numbers of cycle triangles and flow triangles in the same way as the undirected graphs. Releasing the numbers of cycle triangles and flow triangles satisfy ϵ_1-edge-CDP and ϵ_2-edge-CDP, respectively. Hence, the basic approach satisfies ϵ-edge-CDP according to Theorem 3.4.

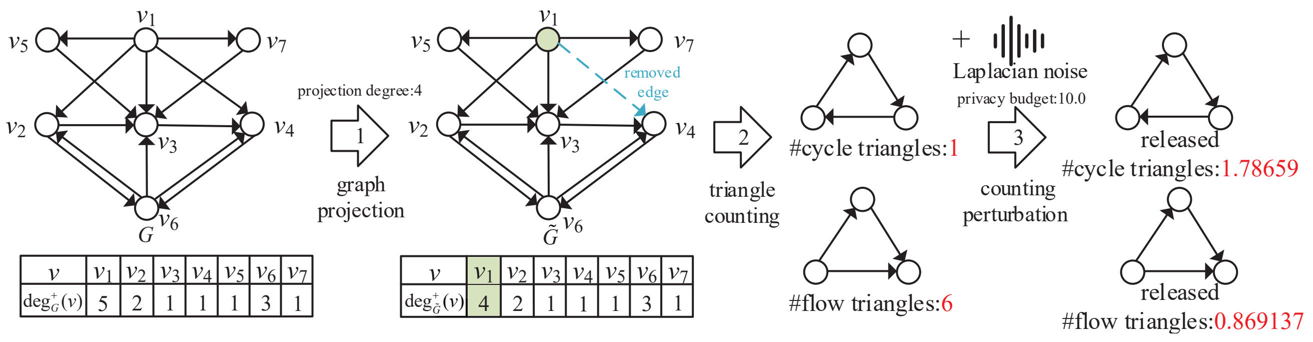
However, the basic approach suffers from the following limitation. For a fixed privacy budget ϵ, it is not easy to decide an optimal privacy budget allocation ratio for ϵ_1 and ϵ_2. The basic approach's utility depends on the privacy budget allocation ratio. If not properly set, the basic approach may fail to achieve optimal utility, leading to underuse of the privacy budget ϵ.

4.2 Overview of Centralized Solution

We design the centralized differential private releasing algorithm based on the principle of improving utility while keeping the same privacy. To address the limitation of the strawman solution that the privacy budget may be wasted, the algorithm releases the numbers of cycle triangles and flow triangles simultaneously. We analyze the global sensitivity of the counting function to implement ϵ-edge-CDP. To further improve the utility of the algorithm, we use graph projection to reduce the global sensitivity, leading to the reduction of the variance of the algorithmic output. As shown in Figure 3, the centralized differential private releasing algorithm contains three phases: (1) graph projection; (2) triangle counting; (3) counting perturbation.

Phase 1: Graph Projection. In the graph projection phase, for each vertex v in the input-directed graph G, assume $\deg_G^+(v)$ is the number of out-neighbors of v. We first make a random permutation of $\{1, 2, 3, \cdots, \deg_G^+(v)\}$. Then we traverse the out-neighbor list of v and at the same time, we traverse the permutation. For any vertex u in the list, if the corresponding number in the permutation is larger than the projection degree $\tilde{d}_{max}$, then u will be removed from the neighbor list.

Phase 2: Triangle Counting. In the triangle counting phase, we compute the exact numbers of cycle triangles and flow triangles in the projected graph $\tilde{G}$ returned from Phase 1. To count cycle triangles, for vertex v in $\tilde{G}$, we traverse the out-neighbor list of v. For any vertex u in v's out-neighbor list, we traverse the out-neighbor list of u. For any vertex w in u's out-neighbor list, if there exists an



v	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\deg^+_G(v)$	5	2	1	1	1	3	1

v	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\deg^+_{\tilde{G}}(v)$	4	2	1	1	1	3	1

Figure 3: A running example for centralized differential private releasing algorithm

```
Algorithm 1: Centralized Algorithm
  Input  : Directed graph $G = (V, E)$; projection degree
           $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$; privacy budget $\epsilon \in \mathbb{R}_{\geq 0}$; the number of all
           the vertices $n$
  Output : The released numbers of cycle triangles $\hat{f}_{c\triangle}(G, \epsilon)$ and
           flow triangles $\hat{f}_{f\triangle}(G, \epsilon)$ in $G$
1 $\tilde{G} \leftarrow$ Graph Projection($G, \tilde{d}_{max}$);
2 $f_\triangle(\tilde{G}) \leftarrow$ Triangle Counting($\tilde{G}$);
3 Compute global sensitivity $GS_{f_\triangle} \leftarrow n + 3\tilde{d}_{max} - 4$;
4 Add the noise $\hat{f}_\triangle(\tilde{G}, \epsilon) \leftarrow f_\triangle(\tilde{G}) + (\mathrm{Lap}(\frac{GS_{f_\triangle}}{\epsilon}), \mathrm{Lap}(\frac{GS_{f_\triangle}}{\epsilon}))$;
5 return $(\hat{f}_{c\triangle}(G, \epsilon), \hat{f}_{f\triangle}(G, \epsilon))$;
```

edge (w, v), then there exists a cycle triangle that contains three vertices v, u, and w. Notice that in this process, each cycle triangle may be counted several times, thus we let $v.id < u.id$ and $v.id < w.id$ to ensure each cycle triangle is counted only once. To count the number of flow triangles, for any vertex v in $\tilde{G}$, we traverse the out-neighbor list of v. For any two vertices u and w, if there exists an edge (u, w), then there exists a flow triangle that contains three vertices v, u, and w. In addition, if there also exists an edge (w, u), then there exists an another flow triangle that contains three vertices v, u, and w.

Phase 3: Counting Perturbation. In the counting perturbation phase, we add i.i.d. Laplacian noise to the counting results returned from Phase 2. The parameter of the Laplacian noise is $\frac{GS_{f_\triangle}}{\epsilon}$. Here, $GS_{f_\triangle}$ is the global sensitivity of counting function $f_\triangle$ and ϵ is the privacy budget. Through adding the noise, the algorithm satisfies ϵ-edge-CDP. We refer the readers to Section 4.3 for more details of Phase 3.

4.3 Sensitivity Analysis

To make the centralized differential private releasing algorithm satisfy ϵ-edge-CDP, the noise should be added to the exact numbers of cycle triangles and flow triangles in the projected-directed graph. To use Laplacian mechanism, unlike undirected graphs, for directed graphs, we need to add two dimensional i.i.d. Laplacian noise. According to Theorem 3.1, the parameter of the Laplacian noise should be determined by the global sensitivity $GS_{f\triangle}$ and the privacy budget ϵ.

Given the counting function $f_\triangle : \mathcal{G} \to \mathbb{Z}^2_{\geq 0}$, we analyze the global sensitivity of $f_\triangle$ according to Definition 3.5. If there is no graph projection phase, then the global sensitivity $GS_{f_\triangle}$ is $4(n-2)$, where n is the number of all vertices in the input-directed graph G. That is because for any edge and another vertex, there exist 4 triangles at most. The aim of graph projection is reducing the global sensitivity, then leading to reducing the variance of algorithmic output. Now, we analyze the global sensitivity of $f_\triangle$ with graph projection.

Theorem 4.1. *Given the counting function $f_\triangle : \mathcal{G} \to \mathbb{Z}^2_{\geq 0}$, where $\mathcal{G} = \{G : \max\{\deg^+(v) : v \in V_G\} \leq \tilde{d}_{max}, |V_G| = n\}$, the global sensitivity of $f_\triangle$ is $n + 3\tilde{d}_{max} - 4$.*

Proof. According to the Definition 3.5, we deduce the global sensitivity $GS_{f_\triangle}$ of $f_\triangle$. The $GS_{f_\triangle}$ can be rewritten as:

$$\begin{aligned} GS_{f_\triangle} &= \max_{G,G' \in \mathcal{G}: G \sim G'} \|f_\triangle(G) - f_\triangle(G')\|_1 \\ &= \max_{G,G' \in \mathcal{G}: G \sim G'} |f_{c\triangle}(G') - f_{c\triangle}(G)| + |f_{f\triangle}(G') - f_{f\triangle}(G)|. \end{aligned}$$

We analyze the special case of two neighboring graphs G and G'. Let the graph shown in Figure 4 be G. Notice that double-direction arrows in Figure 4 represent the existence of two edges, each with two vertices as tail to the other vertex. If we delete the edge (v_1, v_2) and let the graph be G', then we analyze the change of the numbers of cycle triangles and flow triangles, respectively. It is easy to find that the number of cycle triangles will be reduced by $\tilde{d}_{max}$. Similarly, we find that the number of flow triangles will be reduced by $n + 2\tilde{d}_{max} - 4$.

For any directed graph G, if we delete an edge (v_1, v_2) from G, then the change of the number of cycle triangles would never exceed $\tilde{d}_{max}$, because v_2 at most has $\tilde{d}_{max}$ out-neighbors. We also claim that the change of the number of flow triangles would never exceed $n + 2\tilde{d}_{max} - 4$. For any flow triangle that contains (v_1, v_2), we assume the third vertex is v_i, which is different from v_1 and v_2. Assuming $N^+_G(v_1)$ and $N^+_G(v_2)$ represent the sets of the out-neighbors of v_1 and v_2 in G, respectively. Consider 4 cases for v_i:

Case-1: $v_i \in N^+_G(v_1) \cap N^+_G(v_2)$. In this case, there are at most 3 possible flow triangles contains v_1, v_2, and v_i.

Case-2: $v_i \in N^+_G(v_1)$ and $v_i \notin N^+_G(v_2)$. In this case, there are at most 1 possible flow triangles contains v_1, v_2, and v_i.

Case-3: $v_i \in N^+_G(v_2)$ and $v_i \notin N^+_G(v_1)$. In this case, there are at most 1 possible flow triangles contains v_1, v_2, and v_i.

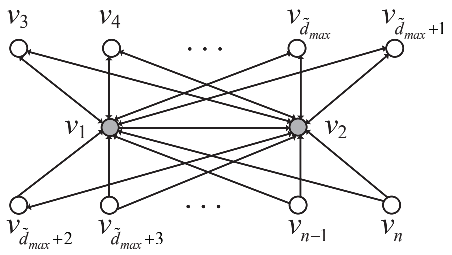


Figure 4: Global Sensitivity Analysis

Case-4: $v_i \notin N^+_G(v_1)$ and $v_i \notin N^+_G(v_2)$. In this case, there are at most 1 possible flow triangles contains v_1, v_2, and v_i.

Due to the constraint of maximum out-degree in directed graph G, the cardinality of $N^+_G(v_1) \cap N^+_G(v_2)$ is at most $\tilde{d}_{max} - 1$. Thus, the number of flow triangles that contain (v_1, v_2) would not exceed $n + 2\tilde{d}_{max} - 4$. □

As shown in Algorithm 1, the Centralized Algorithm first runs graph projection and triangle counting (Lines 1, 2). Then the algorithm computes the global sensitivity $GS_{f_\triangle}$ of the counting function $f_\triangle$ (Line 3). The $GS_{f_\triangle}$ is $n + 3\tilde{d}_{max} - 4$, where n is the number of all vertices in the input-directed graph (i.e. the same as the projected-directed graph) and d_{max} is the projection degree. Then the algorithm adds the i.i.d. Laplacian noise to the exact number of cycle triangles $f_{c\triangle}(\tilde{G})$ and flow triangles $f_{f\triangle}(\tilde{G})$ in $\tilde{G}$ (Line 4). Finally, the algorithm returns the released number of cycle triangles $\hat{f}_{c\triangle}(G, \epsilon)$ and flow triangles $\hat{f}_{f\triangle}(G, \epsilon)$ in G (Line 5).

Example 4.1. *In the example of Figure 3, the central server can access the whole graph G. The directed graph G contains* 7 *nodes* $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ *and the out-degrees of these vertices in order are* $(5, 2, 1, 1, 1, 3, 1)$. *Firstly, the graph projection process runs on the graph G. The projection degree is set to* 4, *so that after the graph projection, the out-neighbors of* v_1 *is set to* $\{v_2, v_3, v_5, v_7\}$, *the edge* (v_1, v_4) *is eliminated. Then the algorithm finds the numbers of cycle triangles and flow triangles in a projected graph* $\tilde{G}$, *those are* 1 *and* 6. *Then the counting perturbation process adds the Laplacian noise to the exact numbers. The parameter of the Laplacian noise is related to the global sensitivity of the counting function and the privacy budget. Especially, after the projection, the global sensitivity of the counting function is* $n + 3\tilde{d}_{max} - 4$, *that is* 15 *and the privacy budget is* 10.0, *thus the parameter of Laplacian noise is* 1.5. *After the perturbation, the center server releases the numbers of cycle triangles and flow triangles in G, which are* 1.78659 *and* 0.869137.

4.4 Algorithm Analysis

Privacy Analysis. We discuss the privacy of the centralized differential private releasing algorithm.

Theorem 4.2. *For any input privacy budget* ϵ, *the centralized differential private releasing algorithm satisfies* ϵ*-edge-CDP.*

Proof. The centralized differential private releasing algorithm first computes the exact numbers of cycle triangles and flow triangles in the projected graph. We denote the counting function as $f_\triangle : \mathcal{G} \to \mathbb{Z}^2_{\geq 0}$. Then, the algorithm adds the Laplacian noise to the function $f_\triangle$ to implement edge-CDP. The parameter of the Laplacian noise is $\frac{n+3\tilde{d}_{max}-4}{\epsilon}$, where $n+3\tilde{d}_{max}-4$ is the global sensitivity of $f_\triangle$ and ϵ is the privacy budget. According to Theorem 3.1, the centralized differential private releasing algorithm satisfies the ϵ-edge-CDP. □

Utility Analysis. We next show the utility of centralized differential private releasing algorithm from two aspects. Firstly, we prove that the output is the unbiased estimation for the exact numbers of cycle and flow triangles. Next, we analyze the variance of the output. Notice that the output is the unbiased estimations, thus the variance of the output could reflect the magnitude of the error between the output and the exact result according to bias-variance decompositon [28].

Theorem 4.3. *Let* $\hat{f}_\triangle(G, \epsilon) = (\hat{f}_{c\triangle}(G, \epsilon), \hat{f}_{f\triangle}(G, \epsilon))$ *be the output of centralized differential private releasing algorithm. Then, for all* $\epsilon \in \mathbb{R}_{\geq 0}$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, *and G such that any vertex in G has at most* $\tilde{d}_{max}$ *out-neighbors,*

$$\mathbb{E}[(\hat{f}_{c\triangle}(G, \epsilon), \hat{f}_{f\triangle}(G, \epsilon))] = (f_{c\triangle}(G), f_{f\triangle}(G)),$$

$$\mathbb{V}[\hat{f}_{c\triangle}(G, \epsilon)] = \frac{2}{\epsilon^2}(n^2 + 6\tilde{d}_{max}n - 8n + 9\tilde{d}^2_{max} - 24\tilde{d}_{max} + 16),$$

$$\mathbb{V}[\hat{f}_{f\triangle}(G, \epsilon)] = \frac{2}{\epsilon^2}(n^2 + 6\tilde{d}_{max}n - 8n + 9\tilde{d}^2_{max} - 24\tilde{d}_{max} + 16).$$

Complexity Analysis. Let n be the number of all vertices, d_{max} is the largest number of out-neighbors of each vertex and $\tilde{d}_{max}$ is the projection degree. The centralized differential private releasing algorithm contains three phases. The first phase is graph projection and takes $O(nd_{max})$ time. The second phase is triangle counting and takes $O(n\tilde{d}^2_{max})$ time. The third phase is counting perturbation and takes $O(1)$ time. The overall complexity is $O(n\tilde{d}^2_{max} + nd_{max})$.

5 DIRECTED TRIANGLE COUNTING WITH EDGE-LDP

In this section, we firstly present the problem of triangle counting with edge-LDP in directed graphs. Then, we propose the local algorithm to address the problem.

5.1 Problem Statement and Strawman Solution

Problem 2. *Given a directed graph* $G = (V, E)$ *represented as out-neighbor lists* $\lambda_1, \lambda_2, \cdots, \lambda_n$, *where user* $v_i (i = 1, 2, \cdots, n)$ *holds* λ_i, *and a privacy budget* ϵ, *we aim to release the numbers of cycle triangles* $f_{c\triangle}(G)$ *and flow triangles* $f_{f\triangle}(G)$ *in G while satisfying* ϵ*-edge-LDP.*

Strawman Solution. The straightforward approach for Problem 2 is a two-phase method. The algorithm first runs the randomized response on each user side. Each user reports the perturbed out-neighbor list to the central server. The central server aggregates all the neighbor list to a noisy graph, then counts the numbers of cycle triangles and flow triangles in the noisy graph and releases them. According to Theorem 3.2 and Theorem 3.3, this approach satisfies ϵ-edge-LDP.

The straightforward approach faces two main limitations: (1) The approach will consume huge running time, that is because the noisy graph is usually more denser than the input graph. Thus, for each user, the number of out-neighbors of each user in the noisy graph is more than the input graph and the triangle counting requires to traverse out-neighbors of each user repeatedly; (2) The approach can not achieve a satisfactory utility, since the noisy graph is usually more denser than the input-directed graph.

5.2 Overview of Local Solution

We design the local differential private releasing algorithm. The total privacy budget ϵ is divided into two parts ϵ_1, ϵ_2 for each phase in the algorithm, respectively. To address the problem that the local view of each user is too limited, we generate a noisy graph through randomized response on each user which satisfies ϵ_1-edge-LDP. Each user can download the noisy graph from the central server. The local estimation is generated by each user and reported to the central server. To satisfy the ϵ_2-edge-LDP, we analyze the global sensitivity of the local estimation. Compared with the strawman solution, this method reduces the number of times to traverse the out-neighbors of each user in the noisy graph. In addition, this method can provide an unbiased estimation of the numbers of cycle triangles and flow triangles in input-directed graph, which implies that this method reaches higher utility than the strawman solution. As shown in Figure 5, the local differential private releasing algorithm contains two phases: (1) noisy graph generation; (2) local estimation.

Phase 1: Noisy Graph Generation. In the noisy graph generation phase, for each user v_i in the input-directed graph G, we process the randomized response for v_i's out-neighbor list λ_i. Then each user reports the noisy out-neighbor list $\overline{\lambda}_i$ to the central server. The central server aggregates all the noisy out-neighbor lists to generate the noisy graph $\overline{G}$. By using random response, the noisy graph generation phase satisfies ϵ_1-edge-LDP. The details of Phase 1 are in Section 5.3.

Phase 2: Local Estimation. In the local estimation phase, for each user v_i in the input-directed graph G, we run the out-neighbor list projection to set some element of λ_i from 1 to 0, such that the number of v_i's out-neighbors does not exceed the projection degree $\tilde{d}_{max}$. We design a local unbiased estimator and the estimator first needs to obtain the cardinalities of 6 sets, respectively. Then the estimator generates the local estimation (wc_i, wf_i) through these numbers. To generate these sets, v_i should download the noisy graph from the central server, which is generated from Phase 1. To satisfy ϵ_2-edge-LDP for this phase, the Laplacian mechanism is used. The parameter of the Laplacian noise is $\frac{GS_{(wc_i, wf_i)}}{\epsilon_2}$. Here, $GS_{(wc_i, wf_i)}$ is the global sensitivity of the local estimation (wc_i, wf_i) and ϵ_2 is the privacy budget allocated in this phase. By adding the noise, this phase satisfies ϵ_2-edge-LDP. The details of Phase 2 are in Section 5.4.

5.3 Noisy Graph Generation

Due to the topology of directed graphs, for each user v_i, v_i can only view the out-neighbors. However, triangles, unlike stars, v_i could not ensure that he is in either cycle triangles or flow triangles. Specially, each user in cycle triangles can only view one other user. For flow triangles, one user can not view two other users, another user can only view one other user, and the remaining one user can view two other users. Although there is one user who can view the other two users, he can not be sure that there is an edge between the other two users. Thus, for each user v_i, to provide the useful local estimation to the central server, more information about the whole input-directed graph needs to be obtained. Based on this observation, we run the randomized response process on each user side, then each user released the perturbed out-neighbor list to the central server. On the central server side, the server aggregates all the out-neighbor lists to generate a noisy graph, then each user can download the noisy graph from the central server.

Algorithm 2: Noisy Graph Generation

```
Input  : Directed graph $G = (V, E)$ represented as out-neighbor
         lists $\lambda_1, \lambda_2, \cdots, \lambda_n$; privacy budget for Phase 1 $\epsilon_1 \in \mathbb{R}_{\geq 0}$
Output : Noisy graph $\overline{G} = (V, \overline{E})$
// User
1 $p_1 \leftarrow \frac{1}{e^{\epsilon_1}+1}$;
2 for $i = 1$ to $n$ do
3     $\overline{\lambda}_{i,i} \leftarrow 0$;
4     for $j = 1$ to $n$, $j \neq i$ do
5         $b \leftarrow$ Bernoulli$(p_1)$;
6         $\overline{\lambda}_{i,j} \leftarrow \lambda_{i,j} \oplus b$;
7     Release $\overline{\lambda}_i$ to the central server;
// Server
8 $\overline{G} \leftarrow$ aggregate $\overline{\lambda}_i$, $i \in [n]$;
9 return $\overline{G}$;
```

As shown in Algorithm 2, the Noisy Graph Generation first runs on each user side. The algorithm initializes the parameter p_1 used in the random response to $\frac{1}{e^{\epsilon_1}+1}$ (Line 1). Then on each user v_i side, the algorithm traverses the v_i's out-neighbor list. For each element $\lambda_{i,j}$ in λ_i, the algorithm generates a random number b following the Bernoulli distribution of parameter p_1. Then the algorithm generates the element $\overline{\lambda}_{i,j}$ by $\lambda_{i,j} \oplus b$. Notice that the self-loop of v_i is always set to inexistence, since the numbers of cycle triangles and flow triangles will not be affected by self-loops. After the generation of the noisy out-neighbor list $\overline{\lambda}_i$, v_i reports $\overline{\lambda}_i$ to the central server. (Lines 2-7) On the central server side, the server aggregates all the out-neighbor lists to generate a noisy graph $\overline{G}$ (Line 8).

5.4 Local Estimation

Similar to the centralized differential private releasing algorithm, for each user v_i, before the estimating process, we use out-neighbor list projection to remove some out-neighbors of v_i. This aims to reduce the global sensitivity of the local estimation, leading to the reduction of the variance of the algorithmic output. Besides, the projection could also reduce the times of out-neighbor lists traversal, then reducing the running time. To design the local estimator, we form the local triangles, local stars, and local single edges from the projected local view $\tilde{\lambda}_i$ and the noisy graph $\overline{G} = (V, \overline{E})$. The method is replacing edges invisible to v_i with edges in the noisy graph.

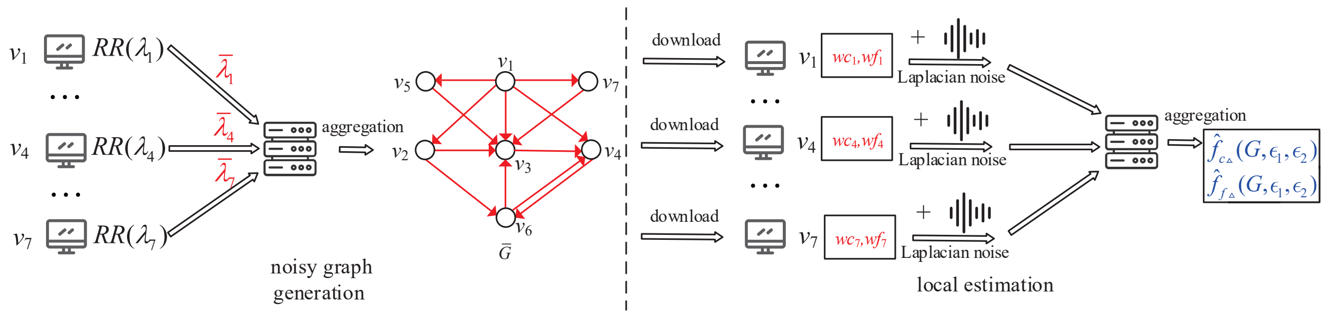


Figure 5: The workflow of local differential private releasing algorithm

We consider local cycle triangles and local subgraphs of cycle triangles,

$$T_i = \{(v_i, v_j, v_k) : \tilde{\lambda}_{i,j} = 1, (v_j, v_k) \in \overline{E}, (v_k, v_i) \in \overline{E}\}, \quad (1)$$

$$T_i^{(1)} = \{(v_i, v_j, v_k) : \tilde{\lambda}_{i,j} = 1, (v_j, v_k) \in \overline{E}, i \neq k\}, \quad (2)$$

$$T_i^{(2)} = \{(v_i, v_j, v_k) : \tilde{\lambda}_{i,j} = 1, (v_k, v_i) \in \overline{E}, j \neq k\}, \quad (3)$$

$$L_i = \{(v_i, v_j, v_k) : \tilde{\lambda}_{i,j} = 1, i \neq k, j \neq k\}. \quad (4)$$

T_i represents the set of local cycle triangles containing v_i; $T_i^{(1)}$ represents the set of local cycle 2-stars starting from v_i; $T_i^{(2)}$ represents the set of local cycle 2-stars whose middle vertex is v_i; L_i represents the set of local single edges starting from v_i.

Similarly, we consider local flow triangles and local subgraphs of flow triangles,

$$R_i = \{(v_i, v_j, v_k) : \tilde{\lambda}_{i,j} = \tilde{\lambda}_{i,k} = 1, (v_j, v_k) \in \overline{E}\}, \quad (5)$$

$$S_i = \{(v_i, v_j, v_k) : \tilde{\lambda}_{i,j} = \tilde{\lambda}_{i,k} = 1, j \neq k\}. \quad (6)$$

R_i represents the set of local flow triangles containing v_i; S_i represents the set of local flow 2-stars whose central vertex is v_i.

Let $t_i, t_i^{(1)}, t_i^{(2)}, l_i, r_i$, and s_i be the cardinalities of $T_i, T_i^{(1)}, T_i^{(2)}$, L_i, R_i, and S_i, respectively. We design the local unbiased estimation $(wc_i, wf_i) = (t_i - p_1 t_i^{(1)} - p_1 t_i^{(2)} + p_1^2 l_i, r_i - p_1 s_i)$. Next, we show the unbiasedness of the estimation after the aggregation of the central server.

Theorem 5.1. *Let $(\frac{1}{3(2p_1-1)^2}\sum_{i=1}^n wc_i, \frac{1}{1-2p_1}\sum_{i=1}^n wf_i)$ be the estimation of the numbers of cycle triangles and flow triangles in input-directed graph G (s.t. the maximum out-degree of vertices in G is not exceed the projection degree $\tilde{d}_{max}$), such that for any user v_i, $wc_i = t_i - p_1 t_i^{(1)} - p_1 t_i^{(2)} + p_1^2 l_i$ and $wf_i = r_i - p_1 s_i$, then*

$$\mathbb{E}[(\frac{1}{3(2p_1-1)^2}\sum_{i=1}^n wc_i, \frac{1}{1-2p_1}\sum_{i=1}^n wf_i)] = (f_{c\triangle}(G), f_{f\triangle}(G)).$$

i.e., $(\frac{1}{3(2p_1-1)^2}\sum_{i=1}^n wc_i, \frac{1}{1-2p_1}\sum_{i=1}^n wf_i)$ is the unbiased estimation for the numbers of cycle triangles and flow triangles in input-directed graph G.

This phase also need to satisfy edge-LDP, otherwise the out-neighbors of v_i may be leaked. The Laplacian mechanism is used to implement edge-LDP. As we allocate the privacy budget ϵ_2, the parameter of the Laplacian noise needs to be set as $\frac{GS_{(wc_i, wf_i)}}{\epsilon_2}$, where $GS_{(wc_i, wf_i)}$ is the global sensitivity of (wc_i, wf_i). Next, we analyze $GS_{(wc_i, wf_i)}$ according to Definition 3.5.

Theorem 5.2. *For each user v_i, given the local estimation (wc_i, wf_i), the global sensitivity of (wc_i, wf_i) is $2(n-2) + 2\tilde{d}_{max}$.*

As shown in Algorithm 3, the Local Estimation first runs on each user side. On each user v_i side, the algorithm has 2 steps, the first is out-neighbor list projection, and the second is using local unbiased estimator to generate the local estimation. In Step 1, the algorithm first initializes $\tilde{\lambda}_i$ to λ_i (Line 3). Then the algorithm generates a random permutation $\mathcal{P}$ of integers ranged from 1 to $\deg_v^+(G)$ (Line 4). Next, for any element $\tilde{\lambda}_{i,j}$ in $\tilde{\lambda}_i$, if $\tilde{\lambda}_{i,j} = 1$, that is $(v_i, v_j) \in E$, then we determine whether the corresponding integer $\mathcal{P}[j]$ in $\mathcal{P}$ is larger than $\tilde{d}_{max}$, and if so, set the $\tilde{\lambda}_{i,j}$ to 0 (Lines 5-7). In Step 2, the algorithm first computes the cardinalities of 6 sets. By using these numbers, the algorithm generates an local unbiased estimation (wc_i, wf_i). (Lines 9-11) Then, the algorithm computes the global sensitivity $GS_{(wc_i, wf_i)}$ of (wc_i, wf_i), and adds the i.i.d. Laplacian noise to (wc_i, wf_i) whose parameter is determined by $GS_{(wc_i, wf_i)}$ and the privacy budget for Phase 2 ϵ_2 (Lines 12, 13). After adding the noise, the algorithm releases $(\hat{wc}_i, \hat{wf}_i)$ to the central server (Line 14). On the server side, the algorithm aggregates all the estimations reported by each user and obtains the estimation of the numbers of cycle triangles $\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2)$ and flow triangles $\hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2)$ in input directed graph G (Line 15).

As Figure 5 shows, the central server can only view all the vertices in the whole graph G, and edges in G can not be viewed. Assume that G has 7 vertices. For any user v_i , v_i generates noisy out-neighbor list $\overline{\lambda}_i$ via randomized response and reports the noisy out-neighbor list $\overline{\lambda}_i$ to the central server. The central server aggregates $\overline{\lambda}_1, \overline{\lambda}_2, \cdots, \overline{\lambda}_7$ to a noisy graph $\overline{G}$. Then, any user v_i downloads the noisy graph from the central server. Through the noisy graph $\overline{G}$ and projected local view $\tilde{\lambda}_i$, the local estimation (wc_i, wf_i) is generated. After adding the Laplacian noise, the noisy local estimation is sent to the central server. Finally, the central server aggregates all the local estimations to obtain the estimated numbers of cycle triangles $\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2)$ and flow triangles $\hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2)$ in G.

5.5 Algorithm Analysis

Privacy Analysis. We discuss the privacy of the local differential private releasing algorithm. Since the algorithm is multi-phase, we analyze the privacy of each phase and use sequential composition theorem to ensure the privacy of the whole algorithm.

Theorem 5.3. *For any input privacy budgets ϵ_1, ϵ_2, the local differential private releasing algorithm satisfies $(\epsilon_1 + \epsilon_2)$-edge-LDP.*

```
Algorithm 3: Local Estimation
Input  : Directed graph $G = (V, E)$ represented as out-neighbor
         lists $\lambda_1, \lambda_2, \cdots, \lambda_n$; noisy graph $\overline{G} = (V, \overline{E})$; privacy
         budget for Phase 2 $\epsilon_2 \in \mathbb{R}_{\geq 0}$; projection degree
         $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$; the parameter $p_1$
Output : The released numbers of cycle triangles $\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2)$
         and flow triangles $\hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2)$ in $G$
   // User
1  for $i = 1$ to $n$ do
2      Step 1: Out-neighbor list projection;
3      Initialize $\tilde{\lambda}_i \leftarrow \lambda_i$;
4      $\mathcal{P} \leftarrow$ a random permutation of $\{1, 2, 3, \cdots, \deg^+_G(v)\}$;
5      for $j = 1$ to $n$ do
6          if $\tilde{\lambda}_{i,j} == 1$ and $\mathcal{P}[j] > \tilde{d}_{max}$ then
7              $\tilde{\lambda}_{i,j} \leftarrow 0$;
8      Step 2: Local unbiased estimator;
9      $t_i, t_i^{(1)}, t_i^{(2)}, l_i \leftarrow$ the cardinalities of $T_i, T_i^{(1)}, T_i^{(2)}, L_i$;
10     $r_i, s_i \leftarrow$ the cardinalities of $R_i, S_i$;
11     $wc_i, wf_i \leftarrow t_i - p_1 t_i^{(1)} - p_1 t_i^{(2)} + p_1^2 l_i, r_i - p_1 s_i$;
12     Compute global sensitivity $GS_{(wc_i, wf_i)} \leftarrow 2(n-2) + 2\tilde{d}_{max}$;
13     $(\hat{wc}_i, \hat{wf}_i) \leftarrow$
           $(wc_i, wf_i) + (\mathrm{Lap}(\frac{GS_{(wc_i, wf_i)}}{\epsilon_2}), \mathrm{Lap}(\frac{GS_{(wc_i, wf_i)}}{\epsilon_2}))$;
14     Release $(\hat{wc}_i, \hat{wf}_i)$ to the central server;
   // Server
15 $(\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2), \hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2)) \leftarrow$
       $(\frac{1}{3(2p_1-1)^2} \sum_{i=1}^{n} \hat{wc}_i, \frac{1}{1-2p_1} \sum_{i=1}^{n} \hat{wf}_i)$;
16 return $(\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2), \hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2))$;
```

Proof. The local differential private releasing algorithm first uses randomized response to perturb the input-directed graph G into noisy graph $\overline{G}$. According to the Theorem 3.2, the Noisy Graph Generation satisfies ϵ_1-edge-LDP. Then, each user adds the Laplacian noise to the local estimation, which makes Local Estimation satisfy ϵ_2-edge-LDP. By Theorem 3.4, the local differential private releasing algorithm satisfies $(\epsilon_1 + \epsilon_2)$-edge-LDP. □

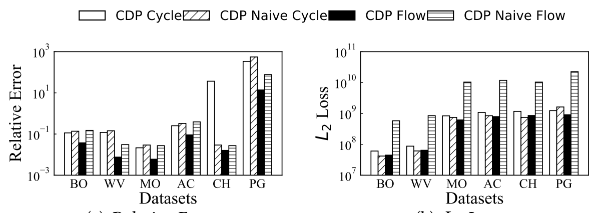
Utility Analysis. We next show that the utility of local differential private releasing algorithm from two aspects. Firstly, we prove that the output is the unbiased estimation for the exact numbers of cycle triangles and flow triangles. Next, we analyze the variance of the output. Similar to the centralized algorithm, notice that the output is the unbiased estimation, thus the variance of the output could reflect the magnitude of the error between the output and the exact result.

Theorem 5.4. *Let $\hat{f}_\triangle(G, \epsilon_1, \epsilon_2) = (\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2), \hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2))$ be the output of local differential private releasing algorithm. Then, for all $\epsilon_1, \epsilon_2 \in \mathbb{R}_{\geq 0}$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, and G such that any vertex in G has at most $\tilde{d}_{max}$ out-neighbors,*

$$\mathbb{E}[(\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2), \hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2))] = (f_{c\triangle}(G), f_{f\triangle}(G)).$$

Table 1: Statistics of the datasets

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$f_{c\triangle}$	$f_{f\triangle}$
Bitcoin OTC (BO)	5.9**K**	35.6**K**	38,581	125,886
Wiki-Vote (WV)	7.1**K**	103.7**K**	43,975	746,557
Math Overflow (MO)	24.8**K**	228**K**	760,663	2,923,310
As-Caida (AC)	26.5**K**	106.8**K**	72,730	218,190
Cit-HepPh (CH)	34.5**K**	421.5**K**	524	1,288,013
P2P-Gnutella (PG)	36.7**K**	88.3**K**	59	1531



(a) Relative Error (b) L_2 Loss

Figure 6: Utility on varying datasets (CDP, CDP Naive)

Theorem 5.5. *Let $\hat{f}_\triangle(G, \epsilon_1, \epsilon_2) = (\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2), \hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2))$ be the output of local differential private releasing algorithm. Then, for all $\epsilon_1, \epsilon_2 \in \mathbb{R}_{\geq 0}$, $\tilde{d}_{max} \in \mathbb{Z}_{\geq 0}$, and G such that any vertex in G has at most $\tilde{d}_{max}$ out-neighbors,*

$$\mathbb{V}[\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2)] \leq O(\frac{e^{4\epsilon_1}}{(1 - e^{\epsilon_1})^4} n^3 (\tilde{d}_{max}^2 + \frac{1}{\epsilon_2^2})),$$

$$\mathbb{V}[\hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2)] \leq O(\frac{e^{\epsilon_1}}{(1 - e^{\epsilon_1})^2} n^2 (\tilde{d}_{max}^2 + \frac{e^{\epsilon_1} n}{\epsilon_2^2})).$$

Complexity Analysis. Let n be the number of all vertices, d_{max} is the largest number of out-neighbors of each vertex and $\tilde{d}_{max}$ is the projection degree. We analyze the complexity of each user and the central server, respectively. We also analysis the communication complexity. Each user first runs the randomized response to perturb the neighbor list and then releases it, which consumes $O(n)$ time and $O(n)$ communication time. Then the central server receives the noisy neighbor lists from all the users, aggregates the noisy graph. This takes $O(n^2)$ time. Then each user receives the noisy graph from the central server, which consume $O(n^2)$ communication time. For any user, the graph projection takes $O(d_{max})$ time and obtaining the local estimation takes $O(n\tilde{d}_{max})$ time, then adding the Laplacian noise and releasing the local estimation both take $O(1)$ time. The central server receives the local estimation and aggregates all the estimations, which takes $O(n)$ time. In summary, each client takes $O(n\tilde{d}_{max})$ time and the central server takes $O(n^2)$ time. In these procedures, the communication between per-user and the central server takes $O(n^2)$ time.

6 EXPERIMENTS

We conduct experiments on a server with Intel(R) Xeon(R) CPU E5-2650 and 128 GB main memory. All experiments are implemented in C++ on the CentOS operating system.

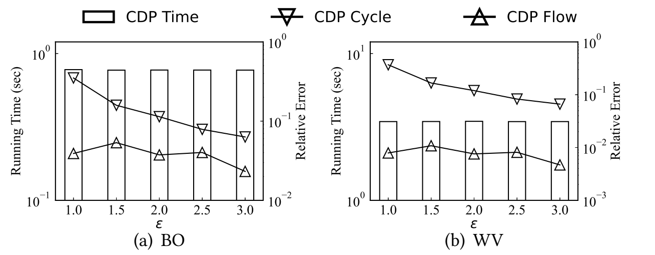


Figure 7: Running time and relative error on varying ϵ (CDP)

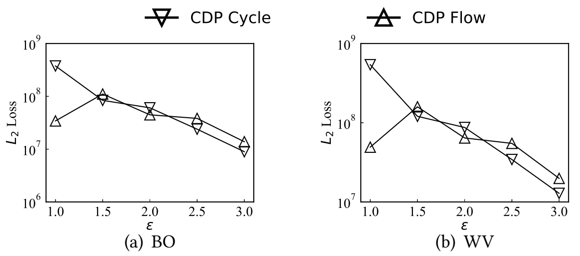


Figure 8: L_2 loss on varying ϵ (CDP)

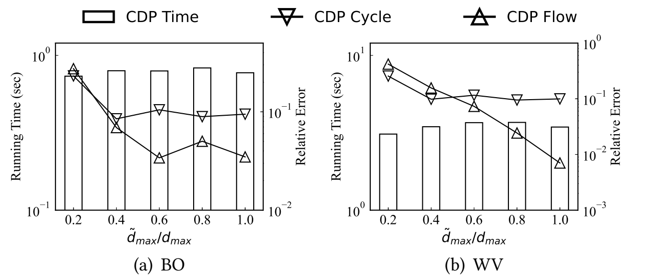


Figure 9: Running time and relative error on varying $\tilde{d}_{max}$ (CDP)

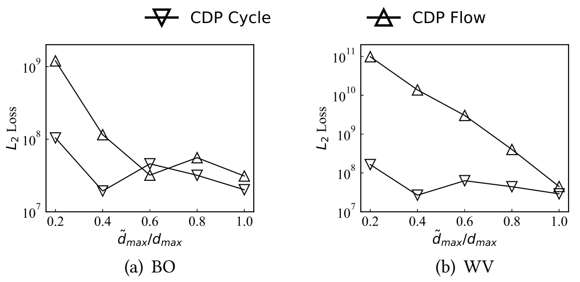


Figure 10: L_2 loss on varying $\tilde{d}_{max}$ (CDP)

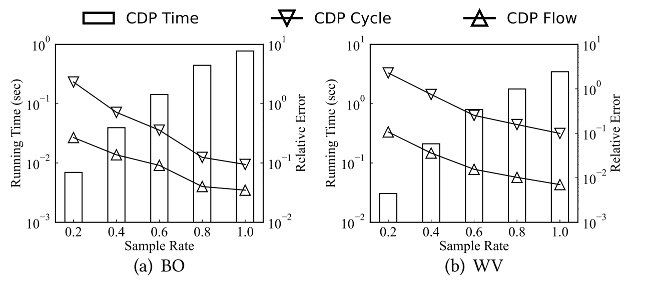


Figure 11: Running time and relative error on varying $|V|$ (CDP)

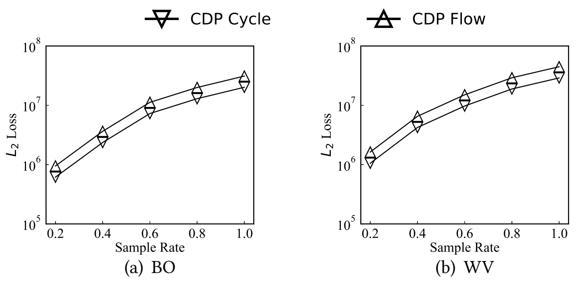


Figure 12: L_2 loss on varying $|V|$ (CDP)

6.1 Experimental Setup

Datasets. We use six real-world directed graphs in our experiments. All datasets can be downloaded from SNAP website.[1] Specifically, Bitcoin OTC is a who-trusts-whom network on the Bitcoin OTC platform; Wiki-Vote is a who-votes-on-whom network on the Wikipedia; Math Overflow is a network of interactions on the website Math Overflow; As-Caida is a full autonomous system network constructed Border Gateway Protocol (BGP) logs; Cit-HepPh is an arxiv high energy physics paper citation network; P2P-Gnutella is a snapshot of Gnutella peer-to-peer file sharing network. Table 1 shows the statistics of these real-world directed graphs.

Algorithms. We compare four algorithms in our experiments.

- **CDP**: The proposed centralized algorithm for triangle counting with edge-CDP in directed graphs.
- **LDP**: The proposed local algorithm for triangle counting with edge-LDP in directed graphs.
- **CDP Naive**: The strawman solution of triangle counting with edge-CDP in directed graphs.
- **LDP Naive**: The strawman solution of triangle counting with edge-LDP in directed graphs.

[1]https://snap.stanford.edu/data/index.html

Parameters and Metrics. The parameters tested in the experiments include privacy budget ϵ, projection degree $\tilde{d}_{max}$, sample rate of vertex cardinality, and privacy budget allocation ratio $\frac{\epsilon_1}{\epsilon}$. The default settings of these 4 parameters are 2.0, d_{max} (i.e., the maximum out-degree of all the vertices in the input-directed graph), 1.0, and 0.5, respectively. For utility evaluating, we measure the relative error and L_2 loss between the algorithmic output and the exact numbers of two type triangles in the input-directed graph. The smaller relative error and L_2 loss mean the larger utility.

6.2 Evaluation of CDP

Overall Results. Firstly, we compare CDP and CDP Naive in utility. The results are shown in Figure 6. For the number of cycle triangles, the relative error and L_2 loss of CDP are close to CDP Naive. But for the number of flow triangles, the relative error and L_2 loss of CDP are almost 50% smaller than CDP Naive.

Then, we evaluate the performance of CDP by varying privacy budget. The results are shown in Figure 7 and Figure 8. We can see that when the privacy budget ϵ is increased, the running time of CDP keeps steady. The reason behind this is that the privacy budget ϵ only affects the addition of the Laplacian noise, thus the running time is insensitive to ϵ.

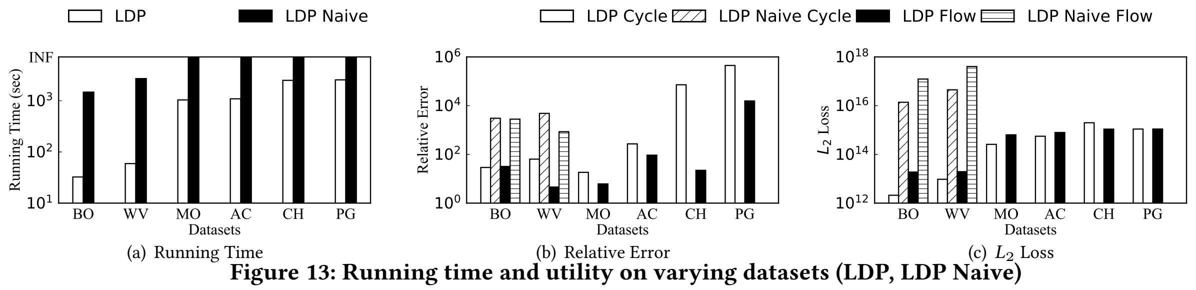


Figure 13: Running time and utility on varying datasets (LDP, LDP Naive)

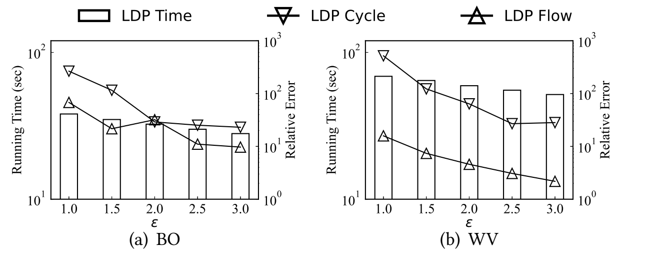


Figure 14: Running time and relative error on varying ϵ (LDP)

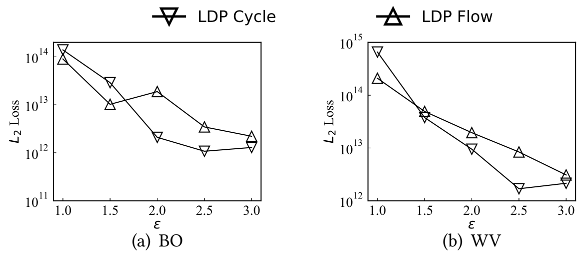


Figure 15: L_2 loss on varying ϵ (LDP)

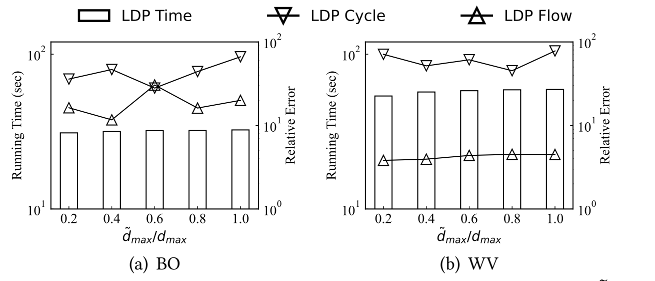


Figure 16: Running time and relative error on varying $\tilde{d}_{max}$ (LDP)

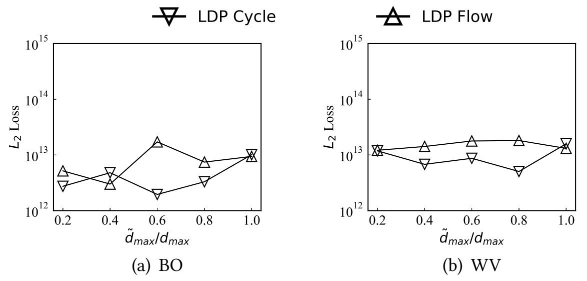


Figure 17: L_2 loss on varying $\tilde{d}_{max}$ (LDP)

For each dataset, when the privacy budget ϵ is increased, both the relative error of CDP is decreased. ϵ affects the scale parameter of the Laplacian noise. When ϵ is increased, the scale parameter is decreased, then leading to the noise is reduced, thus the relative error and L_2 loss are reduced.

Impact of Projection Degree. We next evaluate the impact of projection degree on CDP. The results are shown in Figure 9 and Figure 10. When $\tilde{d}_{max}$ is increased, the running time keeps steady. This is because that for each dataset, the out-degrees of vertices exceeding 95% shall not exceed $0.2d_{max}$. Thus, the graph projection process only affects a small number of vertices.

Then when $\tilde{d}_{max}$ is increased, the relative error and L_2 loss of CDP are first decreased, then keep steady. The reason behind that is when $\tilde{d}_{max}$ is small, after the graph projection process, the difference between the projected-directed graph and input-directed graph is large. The CDP first compute the exact numbers of cycle triangles and flow triangles in the input graph. The large difference between the projected-directed graph and input-directed graph leads to the excessively inaccurate counting results, then the relative error and L_2 loss become large. Then when $\tilde{d}_{max}$ is exceeded a threshold, the difference between the projected-directed graph and input-directed graph becomes small, thus the relative error and L_2 loss keep steady.

Impact of Graph Size. Then we evaluate the effect on CDP of graph size. For every datasets, we sample 5 subgraphs by adjusting the vertex cardinality. The results are shown in Figure 11 and Figure 12. When sample rate is increased, the running time of CDP is increased. The reason is that the size of input-directed graph becomes large. For each dataset, CDP is scale well to graph size.

For utility, when sample rate is increased, the relative error of CDP is decreased and the L_2 loss of CDP is increased. The reason behind that is when the input-directed graph becomes larger, the numbers of cycle and flow triangle usually become larger, thus the relative error becomes smaller and the L_2 loss becomes larger.

6.3 Evaluation of LDP

Overall Results. We compare LDP and LDP Naive in running time and utility. The results are shown in Figure 13. We abort the algorithm if the running time exceeds 2 hours. The running time of LDP is only about 2% of LDP Naive. For utility, the relative error and L_2 loss of LDP are smaller than LDP Naive. Specially, the relative error of LDP is about 1% of LDP Naive and the L_2 loss of LDP is about 0.01% of LDP Naive.

Then, we also evaluate the performance of LDP by varying privacy budget. The results are shown in Figure 14 and Figure 15. We can see that when the privacy budget ϵ is increased, the running time of LDP is decreased. The reason behind this is that for LDP,

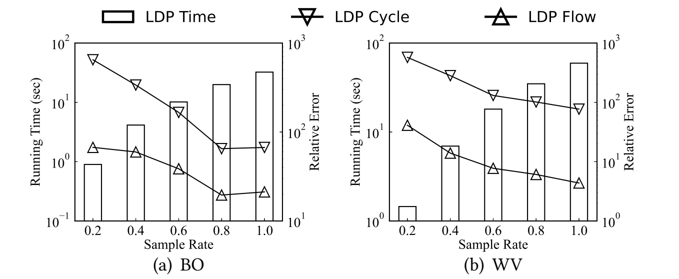


Figure 18: Running time and relative error on varying $|V|$ (LDP)

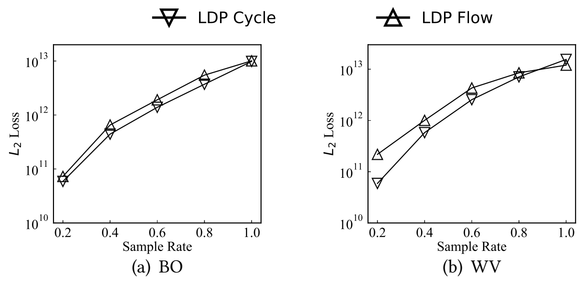


Figure 19: L_2 loss on varying $|V|$ (LDP)

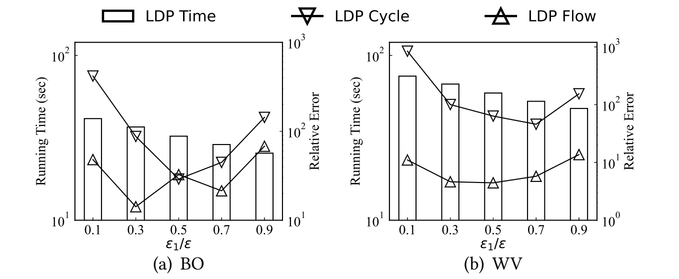


Figure 20: Running time and relative error on varying $\frac{\epsilon_1}{\epsilon}$ (LDP)

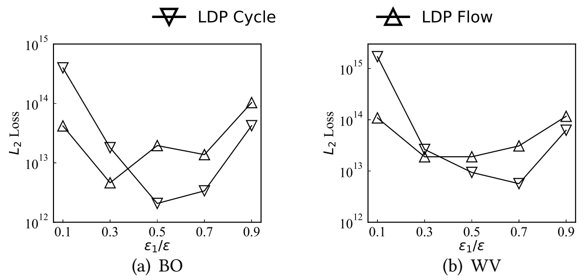


Figure 21: L_2 loss on varying $\frac{\epsilon_1}{\epsilon}$ (LDP)

the randomized response process requires parameter p_1 which relies on ϵ. Specifically, when ϵ is increased, then p_1 is decreased, leading to the number of edges in the noisy graph $|\overline{E}|$ is usually increased, since for all datasets $|E| \ll |V|^2$.

Moreover, for utility, we measure the relative error and L_2 loss of LDP, respectively. When the privacy budget ϵ is increased, the relative error of LDP is decreased. ϵ affects edge changed probability p_1. When ϵ is increased, the edge changed probability is decreased, then the noisy graph is more similar to the input-directed graph, thus the outputs are more close to the exact numbers of cycle triangles and flow triangles in the input-directed graph, leading to the relative error and L_2 loss are all reduced.

Impact of Projection Degree. We next evaluate the impact of projection degree on LDP. The results are shown in Figure 16 and Figure 17. When $\tilde{d}_{max}$ is increased, the running time of LDP also keeps steady. The reason is the same as the reason of the impact of projection degree on CDP.

When $\tilde{d}_{max}$ is increased, then relative error and L_2 loss keep steady for all datasets. The reason is that each user can view not only the local projected out-neighbor list, but also the noisy graph, leading to the LDP is not sensitive to $\tilde{d}_{max}$.

Impact of Graph Size. Then we evaluate the effect on LDP of graph size. The results are shown in Figure 18 and Figure 19. When sample rate is increased, the running time of LDP is increased. The reason is that the size of input-directed graph becomes large. For each dataset, LDP is scale well to graph size.

For utility, when sample rate is increased, the relative error of LDP is decreased and the L_2 loss of LDP is increased. The reason behind that is the same as the reason of the impact of graph size on CDP.

Impact of Privacy Budget Allocation. Finally, for LDP, we evaluate the influence of the privacy allocation ratio on efficiency and utility. The results are shown in Figure 20 and Figure 21. We can see that when privacy budget allocation ratio $\frac{\epsilon_1}{\epsilon}$ is increased, the running time is decreased. The reason behind that is the less privacy budget used in the first phase, the less edges are involved in the noisy graph. Then the running time of the local estimation phase becomes smaller.

For utility, we can see that when $\frac{\epsilon_1}{\epsilon}$ is increased, the relative error and L_2 loss are firstly decreased, and then increased. The phenomenon suggests that when we allocate privacy budgets, we can improve the utility by trying to be as far as possible across different stages.

7 CONCLUSION

In this paper, we have studied privacy-preserving triangle counting problem in directed graphs. To address this problem, we have proposed two algorithms, with centralized and local differential privacy, respectively. The centralized differential private releasing algorithm first runs the graph projection process to obtain the projected graph. Then the algorithm computes the exact numbers of cycle triangles and flow triangles in the projected graph. Next, the algorithm perturbs the counting results by adding the Laplacian noise. For the local differential private releasing algorithm, the noisy graph is generated via randomized response. Each user computes the cardinalities of six sets relying on the noisy graph and projected local view, then computes the local estimation. Next, each user adds the Laplacian noise to the local estimation. Finally, the central server aggregates all the local estimations reported by each user. Our theoretical analysis and empirical evaluations confirm the efficiency and utility of proposed algorithms.

REFERENCES

[1] David A. Bader, Fuhuan Li, Anya Ganeshan, Ahmet Gündogdu, Jason Lew, Oliver Alvarado Rodriguez, and Zhihui Du. 2023. Triangle Counting Through

Cover-Edges. In *HPEC 2023*. IEEE, 1–7.

[2] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2013. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS 2013*. ACM, 87–96.

[3] Shixi Chen and Shuigeng Zhou. 2013. Recursive mechanism: towards node differential privacy and unrestricted joins. In *SIGMOD 2013*. ACM, 653–664.

[4] Xiaofeng Ding, Shujun Sheng, Huajian Zhou, Xiaodong Zhang, Zhifeng Bao, Pan Zhou, and Hai Jin. 2022. Differentially Private Triangle Counting in Large Graphs. *IEEE Trans. Knowl. Data Eng.* 34, 11 (2022), 5278–5292.

[5] Xiaofeng Ding, Xiaodong Zhang, Zhifeng Bao, and Hai Jin. 2018. Privacy-Preserving Triangle Counting in Large Graphs. In *CIKM 2018*. ACM, 1283–1292.

[6] Cynthia Dwork. 2006. Differential Privacy. In *ICALP 2006 (Lecture Notes in Computer Science)*, Vol. 4052. Springer, 1–12.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC 2006 (Lecture Notes in Computer Science)*, Vol. 3876. Springer, 265–284.

[8] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[9] Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. 2023. Triangle Counting with Local Edge Differential Privacy. In *ICALP 2023*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 52:1–52:21.

[10] Giorgio Fagiolo. 2006. Clustering in complex directed networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* 76 2 Pt 2 (2006), 026107.

[11] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. 2019. Effective and Efficient Community Search Over Large Directed Graphs. *IEEE Trans. Knowl. Data Eng.* 31, 11 (2019), 2093–2107.

[12] Xiangyang Gou and Lei Zou. 2021. Sliding Window-based Approximate Triangle Counting over Streaming Graphs with Duplicate Edges. In *SIGMOD 2021*. ACM, 645–657.

[13] Xiangyang Gou and Lei Zou. 2023. Sliding window-based approximate triangle counting with bounded memory usage. *VLDB J.* 32, 5 (2023), 1087–1110.

[14] Masoud Reyhani Hamedani, Jin-Su Ryu, and Sang-Wook Kim. 2023. ELTRA: An Embedding Method based on Learning-to-Rank to Preserve Asymmetric Information in Directed Graphs. In *CIKM 2023*. ACM, 2116–2125.

[15] Lin Hu, Lei Zou, and Yu Liu. 2021. Accelerating Triangle Counting on GPU. In *SIGMOD 2021*. ACM, 736–748.

[16] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2021. Locally Differentially Private Analysis of Graph Statistics. In *USENIX Security 2021*. USENIX Association, 983–1000.

[17] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-Efficient Triangle Counting under Local Differential Privacy. In *USENIX Security 2022*. USENIX Association, 537–554.

[18] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Differentially Private Triangle and 4-Cycle Counting in the Shuffle Model. In *CCS 2022*. ACM, 1505–1519.

[19] Vishesh Karwa, Sofya Raskhodnikova, Adam D. Smith, and Grigory Yaroslavtsev. 2011. Private Analysis of Graph Structure. *Proc. VLDB Endow.* 4, 11 (2011), 1146–1157.

[20] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2013. Analyzing Graphs with Node Differential Privacy. In *TCC 2013 (Lecture Notes in Computer Science)*, Vol. 7785. Springer, 457–476.

[21] Xuankun Liao, Qing Liu, Jiaxin Jiang, Xin Huang, Jianliang Xu, and Byron Choi. 2022. Distributed D-core Decomposition over Large Directed Graphs. *Proc. VLDB Endow.* 15, 8 (2022), 1546–1558.

[22] Qing Liu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. Truss-based Community Search over Large Directed Graphs. In *SIGMOD 2020*. ACM, 2183–2197.

[23] Quanquan C. Liu and C. Seshadhri. 2024. Brief Announcement: Improved Massively Parallel Triangle Counting in O(1) Rounds. In *PODC 2024*. ACM, 519–522.

[24] Shang Liu, Yang Cao, Takao Murakami, Jinfei Liu, and Masatoshi Yoshikawa. 2024. CARGO: Crypto-Assisted Differentially Private Triangle Counting Without Trusted Servers. In *ICDE 2024*. IEEE, 1671–1684.

[25] Yuhan Liu, Suyun Zhao, Yixuan Liu, Dan Zhao, Hong Chen, and Cuiping Li. 2022. Collecting Triangle Counts with Edge Relationship Local Differential Privacy. In *ICDE 2022*. IEEE, 2008–2020.

[26] Wentian Lu and Gerome Miklau. 2014. Exponential random graph estimation under differential privacy. In *KDD 2014*. ACM, 921–930.

[27] David W. McDonald and Shan He. 2023. Hyperbolic Embedding of Attributed and Directed Networks. *IEEE Trans. Knowl. Data Eng.* 35, 7 (2023), 7003–7015.

[28] Kevin P. Murphy. 2012. *Machine learning - a probabilistic perspective.* MIT Press.

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

[30] Fabrizio Parente and Alfredo Colosimo. 2021. Modelling a multiplex brain network by local transfer entropy. *Scientific Reports* 11, 1 (2021), 15525.

[31] You Peng, Xuemin Lin, Michael Yu, Wenjie Zhang, and Lu Qin. 2023. TDB: Breaking All Hop-Constrained Cycles in Billion-Scale Directed Graphs. In *ICDE 2023*. IEEE, 137–150.

[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 *CCS 2017*. ACM, 425–438.

[33] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. 2009. Relationship privacy: output perturbation for queries with joins. In *PODS 2009*. ACM, 107–116.

[34] Tom A. B. Snijders, Philippa E. Pattison, Garry L. Robins, and Mark S. Handcock. 2006. New Specifications for Exponential Random Graph Models. *Sociological Methodology* 36, 1 (2006), 99–153.

[35] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Wendy Hui Wang, and Ting Yu. 2019. Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy. In *CCS 2019*. ACM, 703–717.

[36] Taro Takaguchi and Yuichi Yoshida. 2016. Cycle and flow trusses in directed networks. *CoRR* abs/1603.03519 (2016). http://arxiv.org/abs/1603.03519

[37] Anxin Tian, Alexander Zhou, Yue Wang, and Lei Chen. 2023. Maximal D-truss Search in Dynamic Directed Graphs. *Proc. VLDB Endow.* 16, 9 (2023), 2199–2211.

[38] Thibaud Trolliet, Nathann Cohen, Frédéric Giroire, Luc Hogie, and Stéphane Pérennes. 2021. Interest clustering coefficient: a new metric for directed networks like Twitter. *J. Complex Networks* 10, 1 (2021).

[39] Srinivas Virinchi and Anoop Saladi. 2023. BLADE: Biased Neighborhood Sampling based Graph Neural Network for Directed Graphs. In *WSDM 2023*. ACM, 42–50.

[40] Stanley L. Warner. 1965. Randomized response: a survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60 309 (1965), 63–6.

[41] Bohua Yang, Dong Wen, Lu Qin, Ying Zhang, Xubo Wang, and Xuemin Lin. 2019. Fully Dynamic Depth-First Search in Directed Graphs. *Proc. VLDB Endow.* 13, 2 (2019), 142–154.

[42] Quan Yuan, Zhikun Zhang, Linkang Du, Min Chen, Peng Cheng, and Mingyang Sun. 2023. PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information. In *USENIX Security 2023*. USENIX Association, 3241–3258.

[43] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2015. Private Release of Graph Statistics using Ladder Functions. In *SIGMOD 2015*. ACM, 731–745.

[44] Qingyun Zhang, Yuming Du, Zhouxing Su, Chu-Min Li, Junzhou Xu, Zhihuai Chen, and Zhipeng Lü. 2024. Threshold-Based Responsive Simulated Annealing for Directed Feedback Vertex Set Problem. In *AAAI 2024*. AAAI Press, 20856–20864.

[45] Xiaotian Zhou, Liwang Zhu, Wei Li, and Zhongzhi Zhang. 2023. A Sublinear Time Algorithm for Opinion Optimization in Directed Social Networks via Edge Recommendation. In *KDD 2023*. ACM, 3593–3602.

APPENDIX

A PROOF OF THEOREM 4.3

The expectation of the Laplacian noise is zero. Thus,

$$\begin{aligned}&\mathbb{E}[(\hat{f}_{c\triangle}(G,\epsilon),\hat{f}_{f\triangle}(G,\epsilon))]\\ =&\mathbb{E}[(f_{c\triangle}(G),f_{f\triangle}(G))+(\mathrm{Lap}(\frac{n+3\tilde{d}_{max}-4}{\epsilon}),\mathrm{Lap}(\frac{n+3\tilde{d}_{max}-4}{\epsilon}))]\\ =&(f_{c\triangle}(G),f_{f\triangle}(G))+(0,0)\\ =&(f_{c\triangle}(G),f_{f\triangle}(G)).\end{aligned}$$

According to the variance of the random variable which is Laplace distributed,

$$\begin{aligned}\mathbb{V}[\hat{f}_{c\triangle}(G,\epsilon)]&=\mathbb{V}[f_{c\triangle}(G)+\mathrm{Lap}(\frac{n+3\tilde{d}_{max}-4}{\epsilon})]\\ &=\mathbb{V}[\mathrm{Lap}(\frac{n+3\tilde{d}_{max}-4}{\epsilon})]\\ &=\frac{2}{\epsilon^2}(n^2+6\tilde{d}_{max}n-8n+9\tilde{d}_{max}^2-24\tilde{d}_{max}+16).\end{aligned}$$

Similarly,

$$\mathbb{V}[\hat{f}_{f\triangle}(G,\epsilon)]=\frac{2}{\epsilon^2}(n^2+6\tilde{d}_{max}n-8n+9\tilde{d}_{max}^2-24\tilde{d}_{max}+16).$$

B PROOF OF THEOREM 5.1

Let $t_*=\sum_{i=1}^n t_i$, $t_*^{(1)}=\sum_{i=1}^n t_i^{(1)}$, $t_*^{(2)}=\sum_{i=1}^n t_i^{(2)}$, and $l_*=\sum_{i=1}^n l_i$. Let $l_*^{(0)}$ be the number of tuples (v_i,v_j,v_k), such that v_i, v_j, and v_k are the vertices in the input graph and $\lambda_{i,j}=1$, and $\lambda_{j,k}=\lambda_{k,i}=0$. Let $l_*^{(1)}$ be the number of tuples (v_i,v_j,v_k), such that $\lambda_{i,j}=\lambda_{j,k}=1$, and $\lambda_{k,i}=0$. Let $l_*^{(2)}$ be the number of tuples (v_i,v_j,v_k), such that $\lambda_{i,j}=\lambda_{k,i}=1$, and $\lambda_{j,k}=0$. Let $l_*^{\triangle}$ be the number of tuples (v_i,v_j,v_k), such that $\lambda_{i,j}=\lambda_{j,k}=\lambda_{k,i}=1$. It is easy to show that $l_*=l_*^{(0)}+l_*^{(1)}+l_*^{(2)}+l_*^{\triangle}$ and $l_*^{\triangle}=3f_{c\triangle}(G)$, where $f_{c\triangle}(G)$ is the exact number of cycle triangles in the input-directed graph G.

Considering a cycle triangle, the triangle is counted $3(1-p_1)^2$ times in the expectation of t_*. Considering a cycle 2-star, the cycle 2-star is counted $p_1(1-p_1)$ times in the expectation of t_*. Considering a single edge, the single edge is counted p_1^2 times in the expectation of t_*, then

$$\mathbb{E}[t_*]=(1-p_1)^2l_*^{\triangle}+p_1^2l_*^{(0)}+p_1(1-p_1)(l_*^{(1)}+l_*^{(2)}).$$

Similarly,

$$\begin{aligned}\mathbb{E}[t_*^{(1)}]&=(1-p_1)l_*^{\triangle}+p_1l_*^{(0)}+(1-p_1)l_*^{(1)}+p_1l_*^{(2)},\\ \mathbb{E}[t_*^{(2)}]&=(1-p_1)l_*^{\triangle}+p_1l_*^{(0)}+(1-p_1)l_*^{(2)}+p_1l_*^{(1)}.\end{aligned}$$

Then, we obtain,

$$\begin{aligned}\mathbb{E}[\sum_{i=1}^n wc_i]&=\mathbb{E}[\sum_{i=1}^n(t_i-p_1t_i^{(1)}-p_1t_i^{(2)}+p_1^2l_i)]\\ &=\mathbb{E}[t_*-p_1t_*^{(1)}-p_1t_*^{(2)}+p_1^2l_*]\\ &=\mathbb{E}[t_*]-p_1\mathbb{E}[t_*^{(1)}]-p_1\mathbb{E}[t_*^{(2)}]+p_1^2\mathbb{E}[l_*]\\ &=3(2p_1-1)^2f_{c\triangle}(G).\end{aligned}$$

Similarly, let $r_*=\sum_{i=1}^n r_i$, $s_*=\sum_{i=1}^n s_i$. Let $s_*^{(0)}$ be the number of tuples (v_i,v_j,v_k), such that $\lambda_{i,j}=\lambda_{i,k}=1$, and $\lambda_{j,k}=0$. Let $s_*^{\triangle}$ be the number of tuples (v_i,v_j,v_k), such that $\lambda_{i,j}=\lambda_{i,k}=\lambda_{j,k}=1$. Then it is easy to show that $s_*=s_*^{(0)}+s_*^{\triangle}$ and $s_*^{\triangle}=f_{f\triangle}(G)$, where $f_{f\triangle}(G)$ is the exact number of flow triangles in the input-directed graph G.

Considering a flow triangle, the triangle is counted $(1-p_1)$ times in the expectation of r_*. Considering a flow 2-star, the triangle is counting p_1 times in the expectation of s_*, then

$$\mathbb{E}[r_*]=(1-p_1)s_*^{\triangle}+p_1s_*^{(0)}.$$

Then, we obtain,

$$\begin{aligned}\mathbb{E}[\sum_{i=1}^n wf_i]&=\mathbb{E}[\sum_{i=1}^n(r_i-p_1s_i)]\\ &=\mathbb{E}[r_*-p_1s_*]\\ &=(1-2p_1)f_{f\triangle}(G).\end{aligned}$$

Thus,

$$\mathbb{E}[(\frac{1}{3(2p_1-1)^2}\sum_{i=1}^n wc_i,\frac{1}{1-2p_1}\sum_{i=1}^n wf_i)]=(f_{c\triangle}(G),f_{f\triangle}(G)).$$

C PROOF OF THEOREM 5.2

Considering two out-neighbor lists λ_i and $\lambda_i^{'}$ of v_i differ in one bit, let d_i and $d_i^{'}$ be the number of 1 elements in λ_i and $\lambda_i^{'}$, respectively. Similarly, let t_i, $t_i^{(1)}$, $t_i^{(2)}$, l_i, r_i, s_i, wc_i, wf_i and $t_i^{'}$, $t_i^{(1)'}$, $t_i^{(2)'}$, $l_i^{'}$, $r_i^{'}$, $s_i^{'}$, wc_i, wf_i correspond to $\tilde{\lambda}_i$ and $\tilde{\lambda}_i^{'}$, respectively. Consider two cases: (1) $d_i<\tilde{d}_{max}$; (2) $d_i\geq\tilde{d}_{max}$.

Case-1: $d_i<\tilde{d}_{max}$.

It is easy to show that $l_i^{'}-l_i=(n-2)$. We claim that $t_i^{'}-t_i\leq l_i^{'}-l_i$, since the t_i is more restrictive on tuples than l_i. Similarly, $t_i^{(1)'}-t_i^{(1)}\leq l_i^{'}-l_i$, $t_i^{(2)'}-t_i^{(2)}\leq l_i^{'}-l_i$.

Then, for the local estimation of the number of cycle triangles,

$$\begin{aligned}|wc_i^{'}-wc_i|&=|t_i^{'}-t_i-p_1(t_i^{(1)'}-t_i^{(1)})-p_1(t_i^{(2)'}-t_i^{(2)})+p_1^2(l_i^{'}-l_i)|\\ &\leq(1-p_1-p_1+p_1^2)|l_i^{'}-l_i|\\ &\leq(1-p_1)^2(n-2)\\ &<(n-2).\end{aligned}$$

It is easy to show that $s_i^{'}-s_i=2\binom{d_i+1}{2}-2\binom{d_i}{2}=2d_i$. Similarly, we claim that $r_i^{'}-r_i\leq s_i^{'}-s_i$.

Then, for the local estimation of the number of flow triangles,

$$\begin{aligned}|wf_i^{'}-wf_i|&=|r_i^{'}-r_i-p_1(s_i^{'}-s_i)|\\ &\leq(1-p_1)|s_i^{'}-s_i|\\ &\leq 2(1-p_1)d_i\\ &<2\tilde{d}_{max}.\end{aligned}$$

Case-2: $d_i\geq\tilde{d}_{max}$.

There are two subcases, namely, **Case-2a** for $d_i^{'}=d_i+1$ and **Case-2b** for $d_i^{'}=d_i-1$.

For **Case-2a**, $d_i^{'}=d_i+1$, then after the graph projection, $\tilde{d}_i^{'}=\tilde{d}_i=\tilde{d}_{max}$. It is easy to show $\tilde{\lambda}_i$ and $\tilde{\lambda}_i^{'}$ differ in 0 or 2 bits. If $\tilde{\lambda}_i$ and $\tilde{\lambda}_i^{'}$ differ in 0 bit, then $|wc_i^{'}-wc_i|=0$ and $|wf_i^{'}-wf_i|=0$.

If $\tilde{\lambda}_i$ and $\tilde{\lambda}_i^{'}$ differ in 2 bits, then $|t_i^{'}-t_i| \le 2(n-2)$, $|t_i^{(1)'}-t_i^{(1)}| \le 2(n-2)$, $|t_i^{(2)'}-t_i^{(2)}| \le 2$, $|l_i^{'}-l_i| = 0$. Then, for the local estimation of the number of cycle triangles,

$$\begin{aligned}|wc_i^{'} - wc_i| &= |t_i^{'} - t_i - p_1(t_i^{(1)'} - t_i^{(1)}) - p_1(t_i^{(2)'} - t_i^{(2)}) + p_1^2(l_i^{'} - l_i)| \\ &\le (1 - p_1 - p_1 + p_1^2) \cdot 2(n-2) \\ &= (1-p_1)^2 \cdot 2(n-2) \\ &< 2(n-2).\end{aligned}$$

It is easy to show that $|r_i^{'}-r_i| \le 2(\tilde{d}_i-2)+1 = 2\tilde{d}_i-3 < 2\tilde{d}_{max}$, $|s_i^{'} - s_i| = 0$. Then, for the local estimation of the number of flow triangles,

$$\begin{aligned}|wf_i^{'} - wf_i| &= |r_i^{'} - r_i - p_1(s_i^{'} - s_i)| \\ &= |r_i^{'} - r_i| \\ &< 2\tilde{d}_{max}.\end{aligned}$$

For **Case-2b**, $d_i^{'} = d_i - 1$. If $d_i > \tilde{d}_{max}$, then after graph projection, $\tilde{d}_i^{'} = \tilde{d}_i = \tilde{d}_{max}$. As the same way in **Case-2a**, we can show that $|wc_i^{'} - wc_i| < 2(n-2)$ and $|wf_i^{'} - wf_i| < 2\tilde{d}_{max}$. If $d_i = \tilde{d}_{max}$, then as the same way in **Case-1**, we can show that $|wc_i^{'} - wc_i| < (n-2)$ and $|wf_i^{'} - wf_i| < 2\tilde{d}_{max}$.

Then, for any user v_i, the global sensitivity of $GS_{(wc_i, wf_i)}$ is $2(n-2) + 2\tilde{d}_{max}$.

D PROOF OF THEOREM 5.4

According to Theorem 5.1,

$$\begin{aligned}&\mathbb{E}[(\hat{f}_{c\triangle}(G,\epsilon_1,\epsilon_2), \hat{f}_{f\triangle}(G,\epsilon_1,\epsilon_2))] \\ =&\mathbb{E}[(\frac{1}{3(2p_1-1)^2}\sum_{i=1}^{n}\hat{wc}_i, \frac{1}{1-2p_1}\sum_{i=1}^{n}\hat{wf}_i)] \\ =&\mathbb{E}[(\frac{1}{3(2p_1-1)^2}\sum_{i=1}^{n}wc_i, \frac{1}{1-2p_1}\sum_{i=1}^{n}wf_i) \\ =&(f_{c\triangle}(G), f_{f\triangle}(G)).\end{aligned}$$

E PROOF OF THEOREM 5.5

(1) For the released number of cycle triangles,

$$\begin{aligned}\mathbb{V}[\hat{f}_{c\triangle}(G,\epsilon_1,\epsilon_2)] &= \mathbb{V}[\frac{1}{3(2p_1-1)^2}\sum_{i=1}^{n}\hat{wc}_i] \\ &= \frac{1}{9(2p_1-1)^4}\mathbb{V}[\sum_{i=1}^{n}\hat{wc}_i] \\ &= \frac{1}{9(2p_1-1)^4}(\mathbb{V}[\sum_{i=1}^{n}(t_i - p_1t_i^{(1)} - p_1t_i^{(2)} + p_1^2l_i)] \\ &\quad + \mathbb{V}[\sum_{i=1}^{n}\text{Lap}(\frac{2(n-2)+2\tilde{d}_{max}}{\epsilon_2})]).\end{aligned}$$

We consider the term $\mathbb{V}[\sum_{i=1}^{n}(t_i - p_1t_i^{(1)} - p_1t_i^{(2)} + p_1^2l_i)]$.

$$\begin{aligned}&\mathbb{V}[\sum_{i=1}^{n}(t_i - p_1t_i^{(1)} - p_1t_i^{(2)} + p_1^2l_i)] \\ =&\mathbb{V}[\sum_{i=1}^{n}t_i] + p_1^2\mathbb{V}[\sum_{i=1}^{n}t_i^{(1)}] + p_1^2\mathbb{V}[\sum_{i=1}^{n}t_i^{(2)}] - 2p_1\text{Cov}(\sum_{i=1}^{n}t_i, \sum_{i=1}^{n}t_i^{(1)}) \\ &- 2p_1\text{Cov}(\sum_{i=1}^{n}t_i, \sum_{i=1}^{n}t_i^{(2)}) + 2p_1^2\text{Cov}(\sum_{i=1}^{n}t_i^{(1)}, \sum_{i=1}^{n}t_i^{(2)}).\end{aligned}$$

According to Cauchy-Schwarz inequality,

$$\begin{aligned}-\text{Cov}(\sum_{i=1}^{n}t_i, \sum_{i=1}^{n}t_i^{(1)}) &\le \sqrt{\mathbb{V}[\sum_{i=1}^{n}t_i] \cdot \mathbb{V}[\sum_{i=1}^{n}t_i^{(1)}]}, \\ -\text{Cov}(\sum_{i=1}^{n}t_i, \sum_{i=1}^{n}t_i^{(2)}) &\le \sqrt{\mathbb{V}[\sum_{i=1}^{n}t_i] \cdot \mathbb{V}[\sum_{i=1}^{n}t_i^{(2)}]}, \\ \text{Cov}(\sum_{i=1}^{n}t_i^{(1)}, \sum_{i=1}^{n}t_i^{(2)}) &\le \sqrt{\mathbb{V}[\sum_{i=1}^{n}t_i^{(1)}] \cdot \mathbb{V}[\sum_{i=1}^{n}t_i^{(2)}]}.\end{aligned}$$

Now, we bound the term $\mathbb{V}[\sum_{i=1}^{n}t_i]$.

$$\sum_{i=1}^{n}t_i = \sum_{i=1}^{n}\sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n}\sum_{k=1,k\neq i,k\neq j}^{n}\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E}).$$

For any fixed $i, j \in [n]$, let,

$$a_{i,j} = \sum_{k=1,k\neq i,k\neq j}^{n}\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E}),$$

It is easy to show that if i, j are fixed, then for every k, $\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})$ are independent random variables. Then,

$$\begin{aligned}\mathbb{V}[a_{i,j}] &= \mathbb{V}[\sum_{k=1,k\neq i,k\neq j}^{n}\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})] \\ &= \sum_{k=1,k\neq i,k\neq j}^{n}\mathbb{V}[\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})].\end{aligned}$$

For fixed i, j and k, to bound the any term, we consider three cases: **Case-1:** $\lambda_{j,k} = \lambda_{k,i} = 0$; **Case-2:** $\lambda_{j,k} = 1, \lambda_{k,i} = 0$ or $\lambda_{j,k} = 0, \lambda_{k,i} = 1$; **Case-3:** $\lambda_{j,k} = \lambda_{k,i} = 1$. For **Case-1**, $\mathbb{V}[\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})] = p_1^2(1-p_1^2)$. For **Case-2**, $\mathbb{V}[\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})] = p_1(1-p_1)(1-p_1(1-p_1))$. For **Case-3**, $\mathbb{V}[\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})] = (1-p_1)^2(1-(1-p_1)^2)$. Since $0 < p_1 = \frac{1}{e^\epsilon+1} \le \frac{1}{2}$, for all cases, $\mathbb{V}[\mathbb{I}(<v_j,v_k>\in\overline{E}\wedge<v_k,v_i>\in\overline{E})] \le (1-p_1)^2(1-(1-p_1)^2)$.

Thus, for any $i, j \in [n]$, where $i \neq j$, $\mathbb{V}[a_{i,j}] \le (n-2)(1-p_1)^2(1-(1-p_1)^2)$.

According to Cauchy-Schwarz inequality, for any $i, j, i^{'}, j^{'} \in [n]$, where $i \neq j$ or $i^{'} \neq j^{'}$,

$$\begin{aligned}\text{Cov}(a_{i,j}, a_{i^{'},j^{'}}) &\le \sqrt{\mathbb{V}[a_{i,j}] \cdot \mathbb{V}[a_{i^{'},j^{'}}]} \\ &\le (n-2)(1-p_1)^2(1-(1-p_1)^2).\end{aligned}$$

Let $\mathrm{Cov}(a_{i,j}, a_{i,j}) = \mathbb{V}[a_{i,j}]$. Then,

$$\begin{aligned}\mathbb{V}[\sum_{i=1}^{n} t_i] &= \mathbb{V}[\sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} a_{i,j}] \\ &= \sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{i'=1}^{n} \sum_{j'=1,\tilde{\lambda}_{i',j'}=1}^{n} \mathrm{Cov}(a_{i,j}, a_{i',j'}) \\ &\leq n^2 \tilde{d}_{max}^2 (n-2)(1-p_1)^2(1-(1-p_1)^2).\end{aligned}$$

Then, we bound the term $\mathbb{V}[\sum_{i=1}^{n} t_i^{(1)}]$,

$$\sum_{i=1}^{n} t_i^{(1)} = \sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{I}(<v_j, v_k> \in \overline{E}).$$

It is easy to show that for every $j, k \in [n]$ and $j \neq k$, $\mathbb{I}(<v_j, v_k> \in \overline{E})$ are independent random variables. Then,

$$\begin{aligned}\mathbb{V}[t_i^{(1)}] &= \mathbb{V}[\sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{I}(<v_j, v_k> \in \overline{E})] \\ &= \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{V}[\mathbb{I}(<v_j, v_k> \in \overline{E})] \\ &\leq (n-2)\tilde{d}_{max} p_1(1-p_1).\end{aligned}$$

According to Cauchy-Schwarz inequality, for any $i, i' \in [n]$, where $i \neq i'$,

$$\begin{aligned}\mathrm{Cov}(t_i^{(1)}, t_{i'}^{(1)}) &\leq \sqrt{\mathbb{V}[t_i^{(1)}] \cdot \mathbb{V}[t_{i'}^{(1)}]} \\ &\leq (n-2)\tilde{d}_{max} p_1(1-p_1).\end{aligned}$$

Let $\mathrm{Cov}(b_i, b_i) = \mathbb{V}(b_i)$. Then,

$$\begin{aligned}&\mathbb{V}[\sum_{i=1}^{n} t_i^{(1)}] \\ =& \sum_{i=1}^{n} \sum_{i'=1}^{n} \mathrm{Cov}(t_i^{(1)}, t_{i'}^{(1)}) \\ \leq& n^2(n-2)\tilde{d}_{max} p_1(1-p_1).\end{aligned}$$

Similarly, we bound the term $\mathbb{V}[\sum_{i=1}^{n} t_i^{(2)}]$,

$$\sum_{i=1}^{n} t_i^{(2)} = \sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{I}(<v_k, v_i> \in \overline{E}).$$

For any fixed $i, j \in [n]$, let

$$c_{i,j} = \sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{I}(<v_k, v_i> \in \overline{E}).$$

It is easy to show that if i is fixed, then for every k, $\mathbb{I}(<v_k, v_i> \in \overline{E})$ are independent random variables. Then,

$$\begin{aligned}\mathbb{V}[c_{i,j}] &= \mathbb{V}[\sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{I}(<v_k, v_i> \in \overline{E})] \\ &= \sum_{k=1,k\neq i,k\neq j}^{n} \mathbb{V}[\mathbb{I}(<v_k, v_i> \in \overline{E})] \\ &= (n-2)p_1(1-p_1).\end{aligned}$$

According to Cauchy-Schwarz inequality, for any $i, j, i', j' \in [n]$, where $i \neq j$ or $i' \neq j'$,

$$\begin{aligned}\mathrm{Cov}(c_{i,j}, c_{i',j'}) &\leq \sqrt{\mathbb{V}[c_{i,j}] \cdot \mathbb{V}[c_{i',j'}]} \\ &\leq (n-2)p_1(1-p_1).\end{aligned}$$

Let $\mathrm{Cov}(c_{i,j}, c_{i,j}) = \mathbb{V}[c_{i,j}]$. Then,

$$\begin{aligned}\mathbb{V}[\sum_{i=1}^{n} t_i^{(2)}] &= \mathbb{V}[\sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} c_{i,j}] \\ &= \sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{i'=1}^{n} \sum_{j'=1,\tilde{\lambda}_{i',j'}=1}^{n} \mathrm{Cov}(c_{i,j}, c_{i',j'}) \\ &\leq n^2 \tilde{d}_{max}^2 (n-2)p_1(1-p_1).\end{aligned}$$

Thus,

$$-\mathrm{Cov}(\sum_{i=1}^{n} t_i, \sum_{i=1}^{n} t_i^{(1)}) \leq n^2(n-2)\tilde{d}_{max}^{\frac{3}{2}}(1-p_1)^{\frac{3}{2}} p_1 (2-p_1)^{\frac{1}{2}},$$

$$-\mathrm{Cov}(\sum_{i=1}^{n} t_i, \sum_{i=1}^{n} t_i^{(2)}) \leq n^2(n-2)\tilde{d}_{max}^{2}(1-p_1)^{\frac{3}{2}} p_1 (2-p_1)^{\frac{1}{2}},$$

$$\mathrm{Cov}(\sum_{i=1}^{n} t_i^{(1)}, \sum_{i=1}^{n} t_i^{(2)}) \leq n^2(n-2)\tilde{d}_{max}^{\frac{3}{2}} p_1(1-p_1).$$

Then, since $0 < p_1 \leq \frac{1}{2}$,

$$\mathbb{V}[\sum_{i=1}^{n}(t_i - p_1 t_i^{(1)} - p_1 t_i^{(2)} + p_1^2 l_i)] \leq O((1-p_1)^4 n^3 \tilde{d}_{max}^2).$$

Then,

$$\begin{aligned}\mathbb{V}[\hat{f}_{c\triangle}(G, \epsilon_1, \epsilon_2)] &\leq O(\frac{(1-p_1)^4}{(2p_1-1)^4} n^3 \tilde{d}_{max}^2 + \frac{n^3}{(2p_1-1)^4 \epsilon_2^2}) \\ &= O(\frac{e^{4\epsilon_1}}{(1-e^{\epsilon_1})^4} n^3 (\tilde{d}_{max}^2 + \frac{1}{\epsilon_2^2})).\end{aligned}$$

(2) For the released number of flow triangles,

$$\begin{aligned}\mathbb{V}[\hat{f}_{f\triangle}(G, \epsilon_1, \epsilon_2)] &= \mathbb{V}[\frac{1}{1-2p_1} \sum_{i=1}^{n} \hat{wf}_i] \\ &= \frac{1}{(1-2p_1)^2} \mathbb{V}[\sum_{i=1}^{n} \hat{wf}_i] \\ &= \frac{1}{(1-2p_1)^2} (\mathbb{V}[\sum_{i=1}^{n}(r_i - p_1 s_i)] \\ &\quad + \mathbb{V}[\sum_{i=1}^{n} \mathrm{Lap}(\frac{2(n-2) + 2\tilde{d}_{max}}{\epsilon_2})]).\end{aligned}$$

We consider the term $\mathbb{V}[\sum_{i=1}^{n}(r_i - p_1 s_i)]$,

$$\begin{aligned}\mathbb{V}[\sum_{i=1}^{n}(r_i - p_1 s_i)] &= \mathbb{V}[\sum_{i=1}^{n} r_i] \\ &= \mathbb{V}[\sum_{i=1}^{n} \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,\tilde{\lambda}_{i,k}=1,j\neq k}^{n} \mathbb{I}(<v_j, v_k> \in \overline{E})].\end{aligned}$$

It is easy to show that for every $j, k \in [n]$, $j \neq k$, $\mathbb{I}(<v_j, v_k> \in \overline{E})$ are independent random variables. Then,

$$\begin{aligned}\mathbb{V}[r_i] &= \mathbb{V}[\sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,\tilde{\lambda}_{i,k}=1,j\neq k}^{n} \mathbb{I}(<v_j, v_k> \in \overline{E})] \\ &= \sum_{j=1,\tilde{\lambda}_{i,j}=1}^{n} \sum_{k=1,\tilde{\lambda}_{i,k}=1,j\neq k}^{n} \mathbb{V}[\mathbb{I}(<v_j, v_k> \in \overline{E})] \\ &\leq \tilde{d}_{max}(\tilde{d}_{max} - 1)p_1(1 - p_1).\end{aligned}$$

According to Cauchy-Schwarz inequality, for any $i, i^{'} \in [n]$, where $i \neq i^{'}$,

$$\begin{aligned}\mathrm{Cov}(r_i, r_{i'}) &\leq \sqrt{\mathbb{V}[r_i] \cdot \mathbb{V}[r_{i'}]} \\ &\leq \tilde{d}_{max}(\tilde{d}_{max} - 1)p_1(1 - p_1).\end{aligned}$$

Let $\mathrm{Cov}(r_i, r_i) = \mathbb{V}[r_i]$. Then,

$$\begin{aligned}\mathbb{V}[\sum_{i=1}^{n}(r_i - p_1 s_i)] &= \mathbb{V}[\sum_{i=1}^{n} r_i] \\ &= \sum_{i=1}^{n} \sum_{i'=1}^{n} \mathrm{Cov}(r_i, r_{i'}) \\ &\leq n^2 \tilde{d}_{max}(\tilde{d}_{max} - 1)p_1(1 - p_1).\end{aligned}$$

Then,

$$\mathbb{V}[\hat{f}_{f_\triangle}(G, \epsilon_1, \epsilon_2)] \leq O(\frac{e^{\epsilon_1}}{(1 - e^{\epsilon_1})^2} n^2 (\tilde{d}_{max}^2 + \frac{e^{\epsilon_1} n}{\epsilon_2^2})).$$

F MISSING EXPERIMENTAL RESULTS

F.1 Overall Results of CDP

Varying privacy budget, the performance of CDP in terms of running time, relative error, and L_2 loss is shown in Figure 22 and Figure 23.

F.2 Impact of Projection Degree of CDP

Varying projection degree, the performance of CDP in terms of running time, relative error, and L_2 loss is shown in Figure 24 and Figure 25.

F.3 Impact of Graph Size of CDP

Varying graph size, the performance of CDP in terms of running time, relative error, and L_2 loss is shown in Figure 26 and Figure 27.

F.4 Overall Results of LDP

Varying privacy budget, the performance of LDP in terms of running time, relative error, and L_2 loss is shown in Figure 28 and Figure 29.

F.5 Impact of Projection Degree of LDP

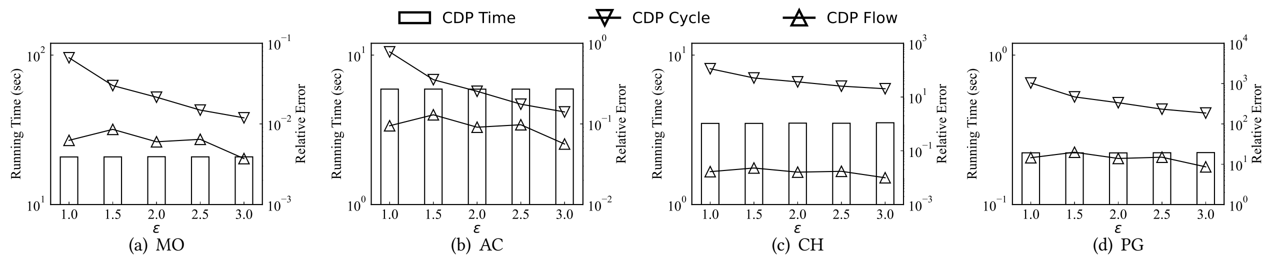
Varying projection degree, the performance of LDP in terms of running time, relative error, and L_2 loss is shown in Figure 30 and Figure 31.

F.6 Impact of Graph Size of LDP

Varying graph size, the performance of LDP in terms of running time, relative error, and L_2 loss is shown in Figure 32 and Figure 33.

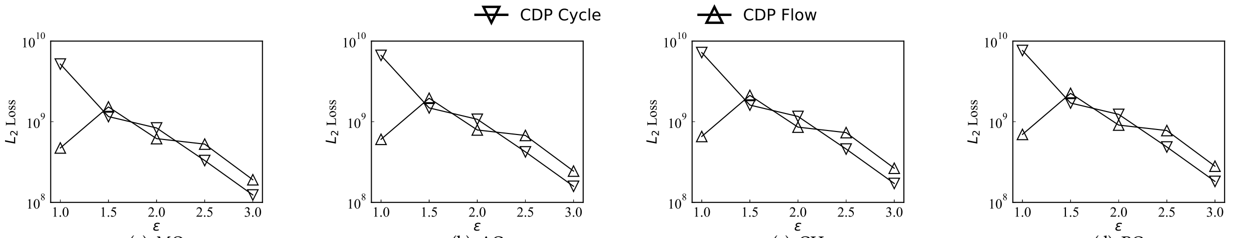
F.7 Impact of Privacy Budget Allocation of LDP

Varying privacy budger allocation ratio, the performance of LDP in terms of running time, relative error, and L_2 loss is shown in Figure 34 and Figure 35.



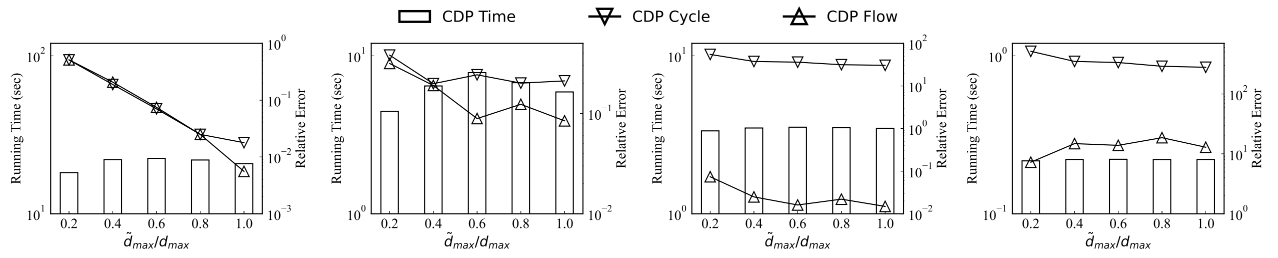
(a) MO (b) AC (c) CH (d) PG

Figure 22: Running time and relative error on varying ϵ (CDP)



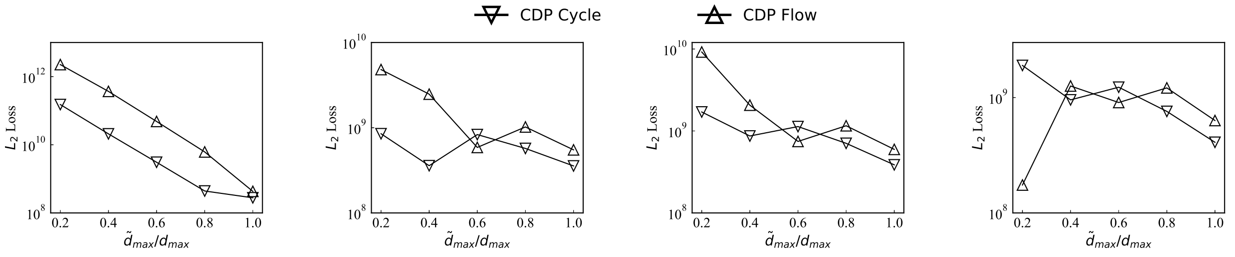
(a) MO (b) AC (c) CH (d) PG

Figure 23: L_2 loss on varying ϵ (CDP)



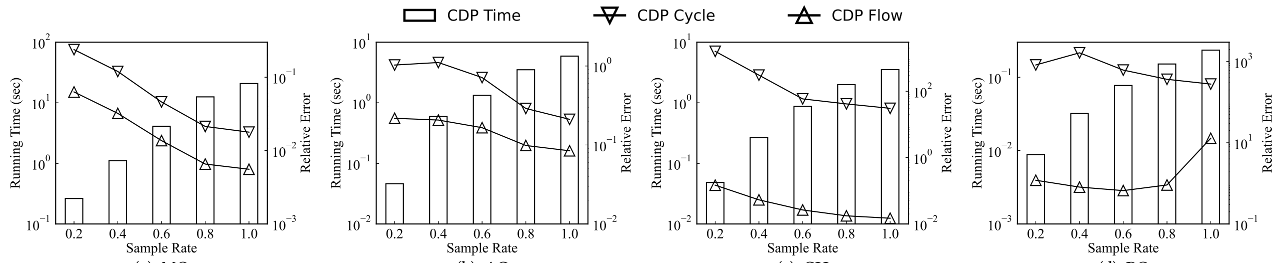
(a) MO (b) AC (c) CH (d) PG

Figure 24: Running time and relative error on varying $\tilde{d}_{max}$ (CDP)



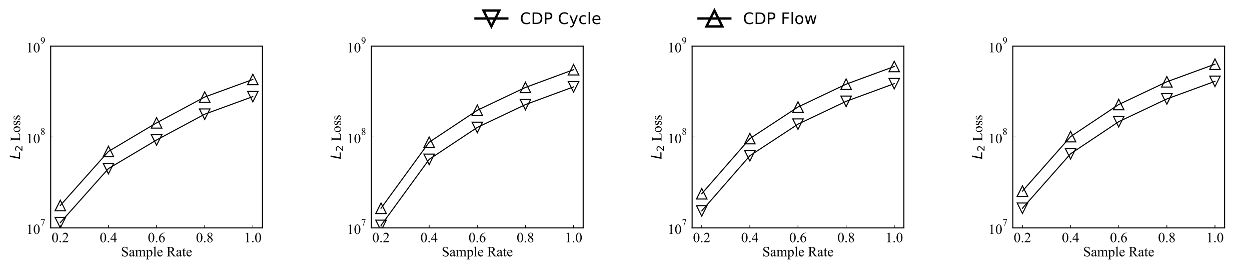
(a) MO (b) AC (c) CH (d) PG

Figure 25: L_2 loss on varying $\tilde{d}_{max}$ (CDP)



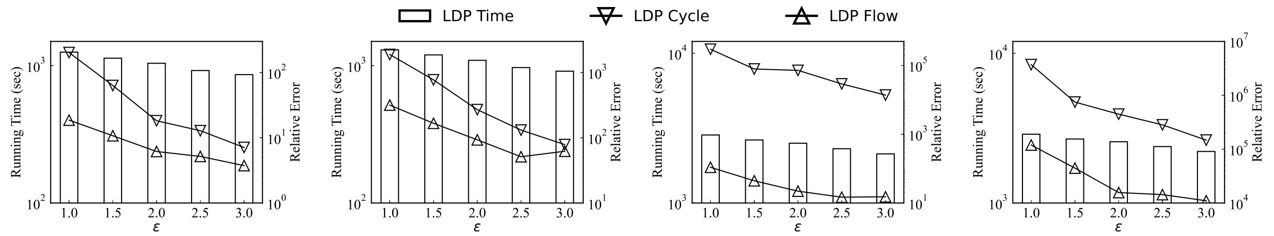
(a) MO (b) AC (c) CH (d) PG

Figure 26: Running time and relative error on varying $|V|$ (CDP)



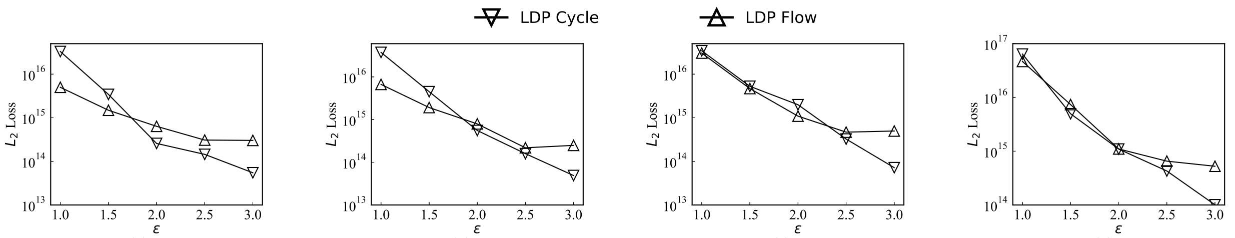
(a) MO (b) AC (c) CH (d) PG

Figure 27: L_2 loss on varying $|V|$ (CDP)



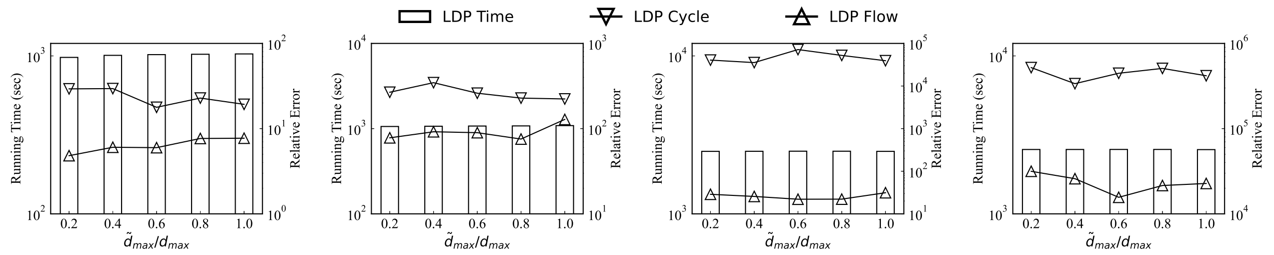
(a) MO (b) AC (c) CH (d) PG

Figure 28: Running time and relative error on varying ϵ (LDP)



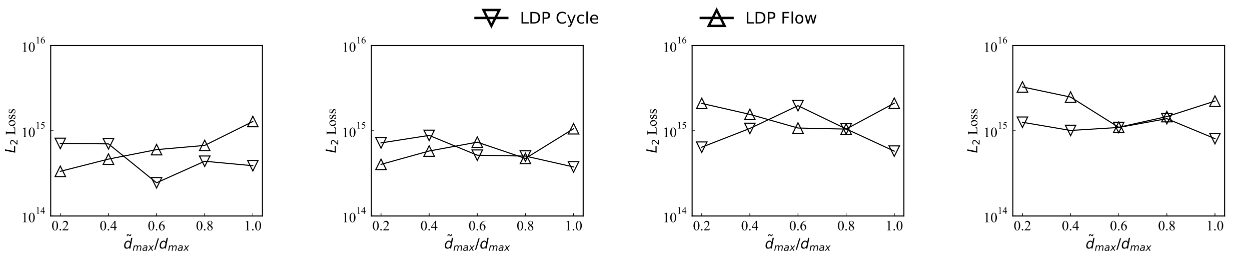
(a) MO (b) AC (c) CH (d) PG

Figure 29: L_2 loss on varying ϵ (LDP)



(a) MO (b) AC (c) CH (d) PG

Figure 30: Running time and relative error on varying $\tilde{d}_{max}$ (LDP)



(a) MO (b) AC (c) CH (d) PG

Figure 31: L_2 loss on varying $\tilde{d}_{max}$ (LDP)

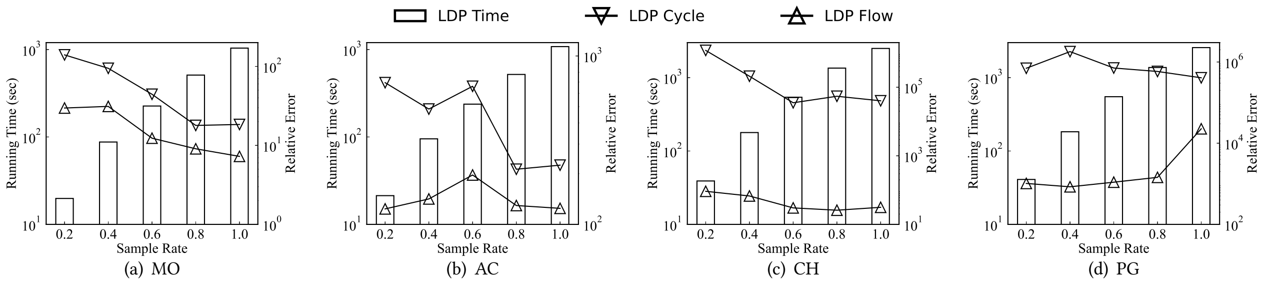


Figure 32: Running time and relative error on varying $|V|$ (LDP)

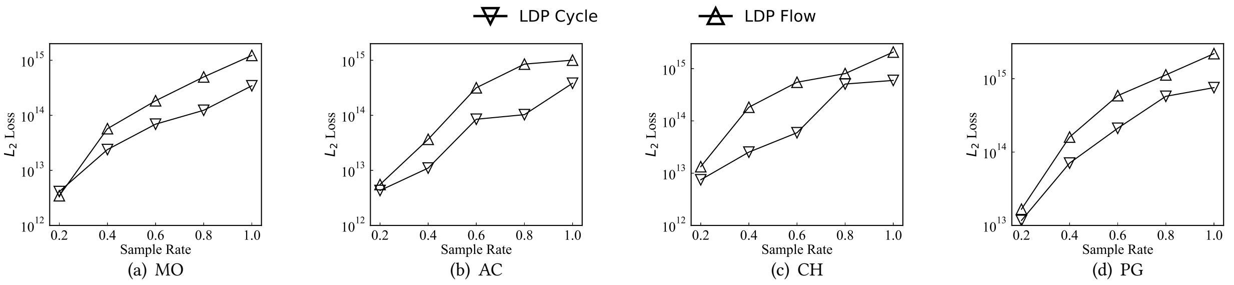


Figure 33: L_2 loss on varying $|V|$ (LDP)

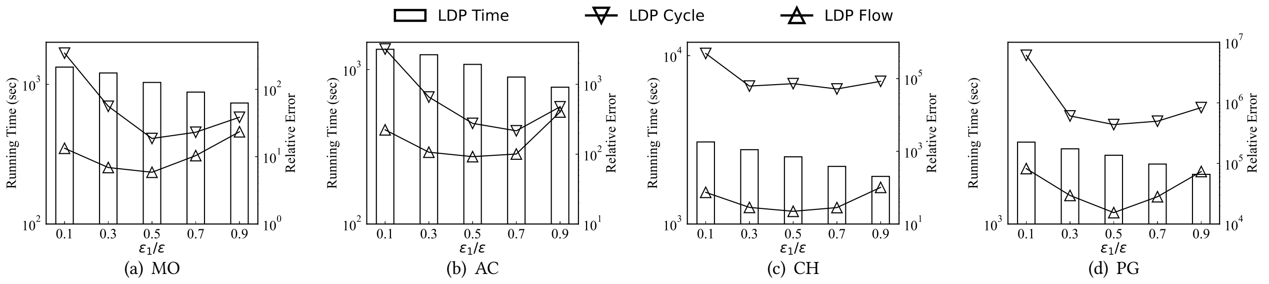


Figure 34: Running time and relative error on varying $\frac{\epsilon_1}{\epsilon}$ (LDP)

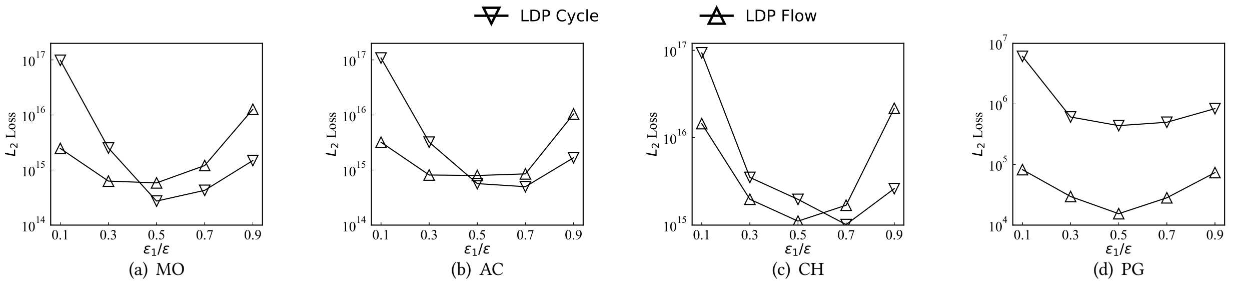


Figure 35: L_2 loss on varying $\frac{\epsilon_1}{\epsilon}$ (LDP)