Locally Differentially Private Degree List Collection in Streaming Directed Graphs

Ziyao Wei, Qing Liu, Zhikun Zhang, Yunjun Gao
College of Computer Science and Technology, Zhejiang University
{wei_zy,qingliucs,zhikun,gaoyj}@zju.edu.cn

ABSTRACT

The degree list is an important characteristic of a graph. Many platforms have to collect the degree for providing some services, such as, user recommendation, financial risk control, and research dataset collection. In streaming directed graphs, the degree list at timestamp t is formally defined as (d_t^+, d_t^-), where d_t^+ (resp. d_t^-) contains out degrees (resp. in degrees) of all the vertices. However, directly collecting degrees from each user may leak the privacy. Most existing work only considers the degree list collection problem in a static scenario of undirected or directed graphs. To bridge this gap, we employ w-event local differential privacy to online collect the degree list of a steaming directed graph. We have proved the parallel composition theorem of event-level local differential privacy and used this theorem to design a collection algorithm. Then, we propose a novel post processing algorithm to adjust the estimated degree list to be digraphic, i.e., there is a directed graph having this degree list. Furthermore, by utilizing the correlation of a streaming directed graph of multiple timestamps, we propose a novel optimization technique to improve the efficiency of the collection algorithm. Experimental results show that proposed algorithms reach high efficiency and utility.

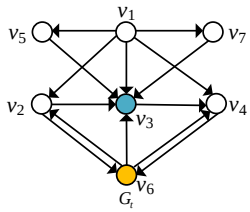
PVLDB Reference Format:
Ziyao Wei, Qing Liu, Zhikun Zhang, Yunjun Gao. Locally Differentially Private Degree List Collection
in Streaming Directed Graphs. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Directed graphs can model asymmetric relationship and has been used in the many fields, such as, web discovery [41, 44], task scheduling [20], and social network analysis [31, 32, 34, 40]. In real-world scenario, edges in a directed graph may change at each timestamp. Therefore, the streaming directed graph [7, 30] is usually used to model the real-world asymmetric relationship.

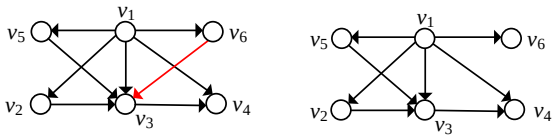
The degree list of a snapshot in a streaming directed graph contains the out degree and in degree of each vertex. For example,



v	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\deg^+_{G_t}(v)$	5	2	1	1	1	3	1

v	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\deg^-_{G_t}(v)$	0	2	5	3	1	2	1

Figure 1: Degree list of a snapshot



v	v_1	v_2	v_3	v_4	v_5	v_6
$\deg^+_{G_1}(v)$	5	1	1	0	1	1

(a) G_1's Out Degree List

v	v_1	v_2	v_3	v_4	v_5	v_6
$\deg^+_{G_2}(v)$	5	1	1	0	1	0

(b) G_2's Out Degree List

Figure 2: Two out degree lists

as shown in Fig. 1, the snapshot G's degree list contains two sublists, one is the out degree list, the other is the in degree list. The out degree list (resp. in degree list) contains all the vertices' out degrees (resp. in degrees) . Specifically, v_3's out and in degrees are 1 and 5, respectively. v_6's out and in degrees are 3 and 2, respectively.

Collecting the degree list of a streaming directed graph can support many applications. For example, social platform usually provide most popular users and most active users, which can recommend prospective interesting users to each user. One method to obtain these users is to recognize the high out degree vertices and the high in degree vertices in user relationship network. For example, in snapshot G_1 shown in Fig. 2, user v_3 has the highest in degree, then we can infer that v_3 is the most popular user. Moreover, user v_1 has the highest out degree, then we can also infer that v_1 is the most active user. Besides, the degree list collection can support downstream tasks. For example, graph collection [6, 36]. Therefore, collecting the degree list of streaming directed graphs is required.

However, directly collecting the degree list may lead to the privacy leakage, since the central server is often untrusted. For example, Fig. 2 shows two directed graphs G_1 and G_2. In G_1, v_6's out degree is 1, then the untrusted central server can infer that among (v_6, v_1), (v_6, v_2), (v_6, v_3), (v_6, v_4), (v_6, v_5), only one exists. While in G_2, v_6's out degree is 0, then the untrusted central server can infer that these edges do not exist. Moreover, existing works [38, 42] under privacy constraint mainly focus on static graphs or streaming undirected graphs. Moreover, existing algorithms cannot support the infinite streaming graphs. The case that multiple edges inserted and deleted at one timestamp cannot be supported neither. In this paper, we consider a streaming directed graph model. In fact, at each timestamp, the streaming directed graph can be inserted and

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

deleted multiple edges. In fact, famous open source systems, such as Apache Flink [1] and Apache Spark [2], always support batch update at one timestamp, i.e., at a timestamp, there might be multiple edges inserted and deleted. Therefore, the privacy concern of the batch updating scenario is noteworthy. We study the degree list collection problem in directed graphs under w-event local differential privacy [39, 47].

Differential privacy [15] is a leading framework for information release while ensuring the privacy of each user will not be leakage [45]. For streaming data, both event-level [22, 43, 46] and user-level [4, 10, 12] differential privacy have been explored. However, the constraint of event-level differential privacy is too loose, leading to that it can only protected the privacy of one timestamp. The constraint of user-level differential privacy is too strict, leading to the privacy budget may be exhausted during the computation. Therefore, w-event differential privacy [13, 28] is proposed. It requires that for any pair of inputs at most differing in w-length window (i.e. interval of time), the algorithm has similar outputs. Specially, if we set $w = +\infty$, then it is equality to the user-level differential privacy; if we set $w = 1$, then it is equality to the event-level differential privacy. By introducing the parameter w, the constraint become neither too strict nor too loose.

Existing works [39, 47] have been explored the w-event local differential privacy. Different from the central setting, it assumes that the central server is untrusted. Each user cannot send the exact information to the central server. Otherwise, the privacy might be leaked. In this paper, we employ w-event local differential privacy for collecting degree list in streaming directed graphs. There are some key technical challenges in algorithm design:

Challenge I: How to implement w-event local differential privacy online? Compare with user-level differential and w-event differential privacy, event-level differential privacy can be implement through some straight-forward mechanisms. When we employ w-event level differential privacy and user-level differential privacy, we generally require the output to be online. However, these mechanisms cannot directly support the online requirement. Therefore, existing work [10, 12] has explored that implementing user-level differential privacy through event-level differential privacy. Derived from this perspective, we explore how to implement w-event local differential privacy through event-level local differential privacy. We have proved the parallel composition of event-level local differential privacy and then designed a w-event locally differentially private algorithm for degree list collection in streaming directed graphs.

Challenge II: How to make the released degree list digraphic? A non-trivial algorithm satisfying w-event local differential privacy must not be deterministic. This means that the algorithmic output is usually inaccurate. Therefore, the output of the algorithm at each timestamp may not be digraphic, i.e., there is no directed graph having this degree list. The post processing for the degree list is required. This problem was studied by Berger from a mathematical perspective [5]. Berger prove a sufficient and necessary condition for determining whether a nonnegative sequence is digraphic. However, no one has yet studied an efficient and effective algorithm that can adjust a nonnegative list to be graphic. We firstly propose an algorithm based on the discrete mathematics result of [5]. The main idea of our proposed algorithm is adjusting the elements in the estimated degree list as little as possible, such that it satisfies this condition.

Challenge III: How to utilize the similarity between snapshots taken at different timestamps? For streaming directed graphs, the snapshots taken at different timestamps have a lot of correlations. Therefore, we can improve efficiency by utilizing this similarity. For some vertices, at each timestamp, the estimated out degree or estimated in degree do not need to be updated. Instead, the estimated out degree or estimated in degree from the previous timestamp is directly sent to the central server. However, the determination of whether a degree should be updated has to satisfy w-event local differential privacy. Otherwise, the whole collection algorithm does not generally satisfy w-event differential privacy. Therefore, we propose a novel technique to determine whether the out degree or in degree should be updated at each timestamp while satisfying w-event local differential privacy.

To implement locally differentially private degree list collection in streaming directed graphs, we make the following contributions in this paper:

- We propose a collection algorithm releasing the degree list at each timestamp. We prove the parallel composition theorem of event-level local differential privacy. In the algorithm, we divide the privacy budget in each window to implement the event-level local differential privacy for releasing degrees at one timestamp. We prove that the collection algorithm satisfies w-event local differential privacy for each user.
- We firstly design an effective and efficient algorithm for adjusting a nonnegtive sequence is digraphic, i.e.,there is a directed graph having this degree list. We use this algorithm for the post processing phase of proposed collection algorithm. This algorithm is based on the result of [5] in discrete mathematics, which is a sufficient and necessary condition for determining whether a nonnegative sequence is digraphic.
- We observe that there exists over-smoothing phenomenon in the exponential mechanism, which is used to implement the w-event local differential privacy in our collection algorithm, we use a step size to reduce the output spaces of the exponential mechanisms. Then, the over-smoothing phenomenon has been mitigated.
- We propose a novel efficiency optimization technique while satisfying w-event local differential privacy. This technique can determine whether the estimated out degree and estimated in degree should be updated at each timestamp for each user while satisfying w-event local differential privacy.
- We conduct several experiments to test the efficiency and utility of proposed algorithms. Experimental results show that the proposed algorithms can achieve high efficiency and utility.

2 RELATED WORKS

We review the related works from two aspects: differential privacy with streaming data and differential privacy on graph analysis.

Differential Privacy with Streaming Data. In streaming differential privacy, there are three common used types of differential privacy: (1) event-level differential privacy; (2) user-level differential privacy; (3) w-event differential privacy. For event-level differential privacy, Dwork et al. [16] first studied the event-level differential privacy with streaming data, and Chan et al. [8] first considered the infinite streaming data. Following these works, Henzinger et al. [22–24] proposed algorithms for counting, sum, and observation. Xie et al. [46] proposed an efficient and accurate cardinality estimation method. Wang et al. [43] designed observation algorithms under both centralized and local differential privacy. For user-level differential privacy, Dong et al. [12] proposed an observation algorithm and then extended it to a join algorithm. Bao et al. [4] devised a collection algorithm with local differential privacy. For w-event differential privacy, Kellaris et al. [28] first defined w-event differential privacy, and designed two observation algorithms. Li et al. [29] proposed two algorithms for observation. For w-event local differential privacy, Ren et al. [39] proposed observation algorithms for infinite streaming data. Du et al. [13] proposes two algorithms for observation, which are different in privacy budget allocation.

Differential Privacy on Graph Analysis. For graph analysis, there are many differentially private algorithms. These algorithms for the computation of a graph analysis function, such as, subgraph counting [19, 21, 25–27], degree hist [9], and degree list [37]. However, most of these works are focused on the static graph. Recently, Raskhodnikova et al. [38] proposes algorithms for computing edge count, high degree, degree list, degree hist, maximum matching, and connected components in streaming undirected graph. However, these algorithms can only handle finite streaming graphs and there is only one edge difference between two snapshots of neighboring timestamps.

3 PRELIMINARIES

We introduce some notations, definitions, and, theorems in this section. A streaming directed graph is a finite sequence $\mathbf{G} = (G_1, G_2, \cdots, G_T)$ or an infinite sequence $\mathbf{G} = (G_1, G_2, \cdots)$. The streaming directed graph $\mathbf{G}$ contains $T \in \mathbb{N} \cup \{+\infty\}$ timestamps. For all timestamps $t \in [T]$, the snapshot G_t of $\mathbf{G}$ has the same vertex set V.[1] At each timestamp t, edges may be inserted into or deleted from $\mathbf{G}$, we use E_t denote the edge set of G_t. Let $n = |V|$ and $m_t = |E_t|$. For each vertex (user) $v_i \in V$, the neighbor list of v_i is denoted as $\mathbf{u}_i \in (\{0,1\}^n \times \{0,1\}^n)^T$, if $T < +\infty$ or $\mathbf{u}_i \in (\{0,1\}^n \times \{0,1\}^n)^{\mathbb{N}}$, if $T = +\infty$. At each timestamp t, let $\mathbf{u}_{i,t} \in \{0,1\}^n \times \{0,1\}^n$ denote the neighbor list of user v_i in snapshot G_t. Let $\mathbf{u}^+_{i,t} = (u^+_{i,t,1}, u^+_{i,t,2}, \cdots, u^+_{i,t,n}) \in \{0,1\}^n$ and $\mathbf{u}^-_{i,t} = (u^-_{i,t,1}, u^-_{i,t,2}, \cdots, u^-_{i,t,n}) \in \{0,1\}^n$ denote the out neighbor list and in neighbor list of user v_i at timestamp t, respectively. Let $d^+_t = (d^+_{t,1}, d^+_{t,2}, \cdots, d^+_{t,n}) \in \mathbb{Z}^n_n$ and $d^-_t = (d^-_{t,1}, d^-_{t,2}, \cdots, d^-_{t,n}) \in \mathbb{Z}^n_n$ be the out degree and in degree lists of G_t, respectively.

3.1 w-event Local Differential Privacy

We introduce w-event local differential privacy. First, we introduce the definition of w-neighboring streaming neighbor lists.

Definition 3.1 (w-neighboring Neighbor Lists). *For each user v_i, given two neighbor lists $\mathbf{u}_i$ and $\mathbf{u}'_i$, and a parameter $w \in \mathbb{N}$, $\mathbf{u}_i$ and $\mathbf{u}'_i$ are w-neighboring, if there exists $\mathcal{I} \in \{[s, s+w-1] : s \in [1, T-w+1]\}$, such that $\{t \in [T] : \mathbf{u}_{i,t} \neq \mathbf{u}'_{i,t}\} \subseteq \mathcal{I}$.*[2]

[1] $[+\infty] := \mathbb{N}$.

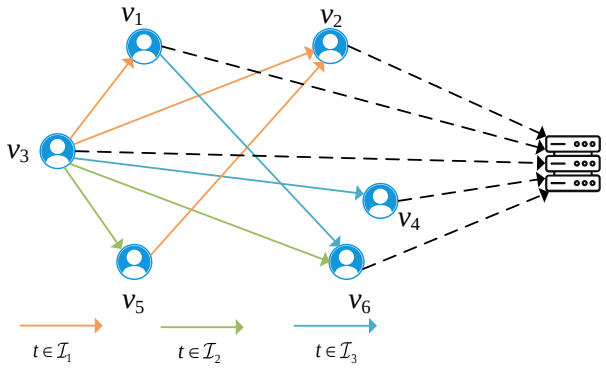


Figure 3: System model

Based on this definition, we introduce w-event local differential privacy.

Definition 3.2 (w-event Local Differential Privacy). *A randomized algorithm $\mathcal{A}$ satisfies w-event local differential privacy with privacy budget $\epsilon \in \mathbb{R}_{\geq 0}$, if for any pair of w-neighboring streaming neighbor lists $\mathbf{u}_i$ and $\mathbf{u}'_i$, and any measurable set $O \subseteq \text{Range}(\mathcal{A})$,*

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

Here, we introduce the exponential mechanism, which is a well-known implement mechanism of differential privacy. In Section 4 and Section 5, we will show that how to implement w-event local differential privacy by using the exponential mechanism.

Definition 3.3 (Global Sensitivity [18]). *Given a utility function $f : \mathcal{U} \times \mathcal{R} \to \mathbb{R}$, where $\mathcal{R}$ is a finite set, the global sensitivity of f is defined as:*

$$GS_f = \max_{r \in \mathcal{R}} \sup_{\mathbf{u}, \mathbf{u}' \in \mathcal{U}} |f(\mathbf{u}, r) - f(\mathbf{u}', r)|.$$

Theorem 3.4 (Exponential Mechanism [18]). *Given a randomized algorithm $\mathcal{A} : \mathcal{U} \to \mathcal{R}$ ($\mathcal{R}$ is a finite set), an input $\mathbf{u} \in \mathcal{U}$ and a privacy budget $\epsilon \in \mathbb{R}_{\geq 0}$, if for any $r \in \mathcal{R}$,*

$$\Pr[\mathcal{A}(\mathbf{u}) = r] = \frac{\exp(\frac{\epsilon f(\mathbf{u}, r)}{2GS_f})}{\sum_{i \in R} \exp(\frac{\epsilon f(\mathbf{u}, i)}{2GS_f})},$$

then $\mathcal{A}$ satisfies differential privacy with privacy budget ϵ.

Theorem 3.5 ([18]). *For any $\epsilon \in \mathbb{R}_{\geq 0}$ and $\beta \in \mathbb{R}_{\geq 0}$, fix $\mathbf{u}$, then*

$$\Pr[f(\mathbf{u}, \mathcal{A}(\mathbf{u})) \leq \max_{x \in \mathcal{R}} f(\mathbf{u}, x) - \frac{2GS_f}{\epsilon}(\log|\mathcal{R}| + \beta)] \leq \exp(-\beta).$$

Differential privacy has some nice properties, including compositions and post-processing.

Theorem 3.6 (Sequential Composition [14]). *For two algorithms $\mathcal{A}_1 : \mathcal{D} \to \mathcal{R}_1$ and $\mathcal{A}_2 : \mathcal{R}_1 \times \mathcal{D} \to \mathcal{R}_2$, such that $\mathcal{A}_1$ satisfies differential privacy with privacy budget ϵ_1 and $\mathcal{A}_2$ satisfies differential privacy with privacy budget ϵ_2, the sequential composition $(\mathcal{A}_1, \mathcal{A}_2) : \mathcal{D} \to \mathcal{R}_1 \times \mathcal{R}_2$ satisfies differential privacy with privacy budget $(\epsilon_1 + \epsilon_2)$.*

[2] For any $a, b \in \mathbb{Z}_{\geq 0}$ and $a \leq b$, $[a, b] := \{a, a+1, \cdots, b\}$.

Theorem 3.7 (Parallel Composition [11, 35]). *For k algorithms $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_k$ depend on disjoint subsets of input datasets, such that for any $i \in [k]$, $\mathcal{A}_i$ satisfies differential privacy with privacy budget ϵ_i, the parallel composition $(\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_k)$ satisfies differential privacy with privacy budget $\max_{i\in[k]} \epsilon_i$.*

Theorem 3.8 (Post Processing [18]). *For any algorithm $\mathcal{A}$ satisfying differential privacy with privacy budget ϵ and any measurable mapping f, $f \circ \mathcal{A}$ also satisfies differential privacy with privacy budget ϵ.*

3.2 System Model and Threat Model

As shown in Fig. 3, given a streaming directed graph $\mathbf{G}$, the system model contains an untrusted server, and each vertex v in $\mathbf{G}$ corresponds to a user. For each user v_i, v_i holds a view of v_i's out-neighborhoods and in-neighborhoods at each timestamp. If user v_1 is the out-neighborhood of user v_2, then user v_2 is the in-neighborhood of user v_1.

For each user v_i, if user v_i releases the exact information to the untrusted central server, then the central server may access the privacy information of user v_i through the received information. Specifically, the out-neighborhoods and in-neighborhoods of user v_i.

3.3 Problem Definition

To support the real-world application, we aim to release the degree list online, thus at each timestamp t, the algorithm should release the degree list $d_t = (d_t^+, d_t^-) \in \mathbb{Z}_n^n \times \mathbb{Z}_n^n$ of the snapshot G_t.

Problem 3.9. *Given a streaming directed graph $\mathbf{G}$ containing T timestamps, find an algorithm $\mathcal{A}$, such that at each timestamp $t \in [T]$, $\mathcal{A}$ outputs the snapshot G_t's degree list $d_t = (d_t^+, d_t^-)$, while $\mathcal{A}$ satisfies w-event local differential privacy for each user v_i in $\mathbf{G}$.*

4 COLLECTION ALGORITHM

In this section, we propose a degree list collection algorithm in streaming directed graphs satisfying w-event local differential privacy. Due to the space limitation, some proofs can be found in our technical report [].

4.1 Overview

To implement w-event local differential privacy, we first show that the composition theorem.

Theorem 4.1 (Parallel Composition of Event-Level Differential Privacy). *Given n algorithms $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_w$ depend on different elements of input neighbor list, such that for any $i \in [n]$, $\mathcal{A}_i$ satisfies 1-event local differential privacy (i.e. event-level local differential privacy) with privacy budget ϵ_i, then the parallel composition $(\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_w)$ satisfies w-event local differential privacy with privacy budget $\sum_{i=1}^n \epsilon_i$.*

According to Theorem 4.1, the algorithm divides the total privacy budget ϵ during any w-length interval. At each timestamp t, the algorithm releases the degree list of G_t under 1-event local differential privacy with privacy budget ϵ/w for each user. Then, by Theorem 4.1, the algorithm satisfies w-event differential privacy with the privacy budget ϵ for each user. At each timestamp t, the algorithm contains three phase: (1) neighbor list projection; (2) degree collection; (3) post processing.

Phase 1: Neighbor List Projection. At a timestamp, the number of insertions and deletions of incident outgoing edges for a vertex can be up to $n-1$, and likewise for incident incoming edges. Therefore, it can verify that the global sensitivities of utility functions are $n-1$. Then, employing exponential mechanism will lead to low utility. To reduce the global sensitivity, we use neighbor list projection technique. After the projection, the out degree (resp. in degree) of any vertex will not exceed the threshold $\tilde{d}_{\max}^+$ (resp. $\tilde{d}_{\max}^-$). Then, the global sensitivities of utility functions can be reduced to $\tilde{d}_{\max}^+$ and $\tilde{d}_{\max}^-$, respectively.

Phase 2: Degree Collection. In this phase, each user first computes his/her out and in degrees. Then, the user gets the estimated out and in degrees through exponential mechanisms. Then, the user reports the estimated out and in degrees to the central server. Finally, the central server aggregates all the results to obtain a estimated degree list.

Phase 3: Post Processing. The degree list returned from **Phase 2** contains all users' degrees, but this degree list is not necessarily digraphic, i.e., there is no directed graph has this degree list. We adjust the degree list according to the mathematics result of [5], which is a necessary and sufficient condition for a degree list to be digraphic. The algorithmic output of the post processing phase satisfies this condition, which means the algorithmic output is digraphic.

4.2 Neighbor List Projection

For any user v_i, at each timestamp t, to implement local differential privacy with privacy budget ϵ, the exponential mechanism is used. For out degree releasing, we define utility function,

$$f^+ : \{0,1\}^n \backslash \{\mathbf{1}\} \times \mathbb{Z}_n \to \mathbb{Z}, \quad (\mathbf{u}_{i,t}^+, r) \mapsto -|\sum_{j=1}^n u_{i,t,j}^+ - r|.$$

Similarly, for in degree releasing, the utility function is defined as,

$$f^- : \{0,1\}^n \backslash \{\mathbf{1}\} \times \mathbb{Z}_n \to \mathbb{Z}, \quad (\mathbf{u}_{i,t}^-, r) \mapsto -|\sum_{i=1}^n u_{i,t,j}^- - r|.$$

Here, the domains of $\mathbf{u}_{i,t}^+$ and $\mathbf{u}_{i,t}^-$ are both $\{0,1\}^n \backslash \{\mathbf{1}\}$, since the self-loop is banned from appearing in the input graph. Therefore, if we choose the estimated out degree $r \in \mathbb{Z}_n$ closer to $\sum_{j=1}^n u_{i,t,j}^+$ (i.e., the out degree of v_i in G_t), the function value f^+ becomes larger. Similarly, if we choose the estimated in degree $r \in \mathbb{Z}_n$ closer to $r \in \mathbb{Z}_n$ closer to $\sum_{i=1}^n u_{i,t,j}^-$ (i.e., the in degree of v_i in G_t), the function value f^- become larger. Notice that since the exponential mechanism tends to choose r leading to large utility function value, the definitions of f^+ and f^- are reasonable to solve the degree estimation problem.

Proposition 4.2 (Global Sensitivity). *The global sensitivities of f^+ and f^- are both $n-1$.*

Intuitively, according to Theorem 3.5, smaller the cardinality of the candidate set $\mathcal{R}$ and smaller global sensitivity GS_{f^+} can achieve better utility. Therefore, we employ neighbor list projection technique for the out degree list $\mathbf{u}_{i,t}^+$, to reduce both the cardinality

of the candidate set $\mathcal{R}$ and the global sensitivity GS_{f^+}. Similarly, we also employ neighbor list projection technique for the in degree list $\mathbf{u}^-_{i,t}$.

In the neighbor list projection phase, given two thresholds $\tilde{d}^+_{t,\max}$ and $\tilde{d}^-_{t,\max}$, if the number of 1 elements of out neighbor list $\mathbf{u}^+_{i,t}$ (i.e., $\sum_{j=1}^n u^+_{i,t,j}$) is at most $\tilde{d}^+_{t,\max}$, then we do nothing. Otherwise, we randomly change some 1 elements to 0 leading to that $\sum_{j=1}^n u^+_{i,t,j} = \tilde{d}^+_{t,\max}$. Similarly, we do the same operation on $\mathbf{u}^-_{i,t}$ leading to that the number of 1 elements of in neighbor list $\mathbf{u}^-_{i,t}$ (i.e., $\sum_{j=1}^n u^-_{i,t,j}$)) is at most $\tilde{d}^-_{t,\max}$.

Notice that since during the neighbor list projection, the input neighbor list $\mathbf{u}_{i,t}$ will become the projected neighbor list $\tilde{\mathbf{u}}_{i,t}$. Moreover, the estimated out degree only need to be chosen from $[0, \tilde{d}^+_{t,\max}]$ and the estimated in degree only need to be chosen from $[0, \tilde{d}^-_{t,\max}]$. Therefore, the domains of utility functions are both changed. We consider the new utility functions $\tilde{f}^+$ and $\tilde{f}^-$. The domain of $\tilde{f}^+$ is $\{\mathbf{a} \subseteq \{0,1\}^n : \sum_{j=1}^n a_j \le \tilde{d}^+_{\max}\} \times [0, \tilde{d}^+_{\max}]$. Similarly, the domain of $\tilde{f}^-$ is $\{\mathbf{a} \subseteq \{0,1\}^n : \sum_{j=1}^n a_j \le \tilde{d}^-_{\max}\} \times [0, \tilde{d}^-_{\max}]$.

Theorem 4.3. *The global sensitivity of the $\tilde{f}^+$ and $\tilde{f}^-$ is $\tilde{d}^+_{\max}$ and $\tilde{d}^-_{\max}$, respectively.*

Example 4.4. *Assume that $V = \{v_1, v_2, v_3, v_4, v_5\}$, the thresholds $\tilde{d}^+_{\max} = 2$ and $\tilde{d}^-_{\max} = 1$. For v_3 at timestamp t, consider two neighbor lists $\mathbf{u}_{i,t} = (\boldsymbol{\lambda}, \boldsymbol{\mu}) = ((1,1,0,1,1),(1,1,0,1,1))$ and $\mathbf{u}'_{i,t} = (\boldsymbol{\lambda}', \boldsymbol{\mu}') = ((1,0,0,0,0),(1,0,0,0,1))$. Then, it is easy to find that $\max_{r\in\mathbb{Z}_5} |f^+(\boldsymbol{\lambda}, 5) - f^+(\boldsymbol{\lambda}', r)| = 3$ and $\max_{r\in\mathbb{Z}_5} |f^-(\boldsymbol{\mu}, r) - f^-(\boldsymbol{\mu}', r)| = 2$. After the projection, the neighbor lists are not unique. The neighbor lists are possibly $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}) = ((1,1,0,0,0),(1,0,0,0,0))$ and $(\tilde{\boldsymbol{\lambda}}', \tilde{\boldsymbol{\mu}}') = ((1,0,0,0,0),(1,0,0,0,0))$. Then, we find that $\max_{r\in[0,2]} |\tilde{f}^+(\tilde{\boldsymbol{\lambda}}, r) - \tilde{f}^+(\tilde{\boldsymbol{\lambda}}', r)| = 1 < \tilde{d}^+_{\max}$ and $\max_{r\in\{0,1\}} |\tilde{f}^-(\tilde{\boldsymbol{\mu}}) - \tilde{f}^-(\tilde{\boldsymbol{\mu}}')| = 0 < \tilde{d}^-_{t,\max}$.*

Notice that in the we only need to obtain the out degree and in degree of each user at each timestamp. Thus. the neighbor list projection technique does not need to process while we implementing the algorithm. For each user v_i at timestamp t, we only need to obtain the out degree and in degree after neighbor list projection through,

$$\tilde{d}^+_{t,i} \leftarrow \min\{\tilde{d}^+_{t,i}, \tilde{d}^+_{\max}\},$$
$$\tilde{d}^-_{t,i} \leftarrow \min\{\tilde{d}^-_{t,i}, \tilde{d}^-_{\max}\}.$$

For user v_i at timestamp t, if $d^+_{t,i} < \tilde{d}^+_{\max}$ (resp. $d^-_{t,i} < \tilde{d}^-_{\max}$), then the neighbor list projection does not make sense, the output of the neighbor list projection phase is $\tilde{d}^+_{t,i}$ (resp. $d^-_{t,i}$). Otherwise, if $d^+_{t,i} > \tilde{d}^+_{\max}$ (resp. $d^-_{t,i} > \tilde{d}^-_{\max}$), then the output of the neighbor list projection phase is $\tilde{d}^+_{\max}$ (resp. $\tilde{d}^-_{\max}$).

4.3 Degree Collection

The degree collection requires each user reports their estimated out degree and in degree to central server while satisfying local differential privacy. We have mentioned that we use the exponential mechanism to implement local differential privacy.

Algorithm 1: Degree Collection

Input : G_t represented as $\mathbf{u}_{1,t}, \mathbf{u}_{2,t}, \cdots, \mathbf{u}_{n,t}$; the thresholds $\tilde{d}^+_{\max}$ and $\tilde{d}^-_{\max}$; the number of users n; privacy budget $\epsilon \in \mathbb{R}_{\ge 0}$

Output : The estimated out degrees $\hat{d}^+_{t,i}$ and in degrees $\hat{d}^-_{t,i}$ of user v_i on v_i's side, $\forall i \in [n]$

// On each user side

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

2 $\quad d^+_{t,i} \leftarrow \sum_{j=1}^n u^+_{i,t,j}$;

3 $\quad d^-_{t,i} \leftarrow \sum_{j=1}^n u^-_{i,t,j}$;

4 $\quad$ **for** $j = 0$ *to* $\tilde{d}^+_{\max}$ **do**

5 $\quad\quad p^+(j) \leftarrow \exp(-\frac{\epsilon|\min\{d^+_{t,i}, \tilde{d}^+_{\max}\} - j|}{2\tilde{d}^+_{\max}})$;

6 $\quad p^+_{\text{sum}} \leftarrow \sum_{j=0}^{\tilde{d}^+_{\max}} p^+(j)$;

7 $\quad$ Sample $\hat{d}^+_{t,i}$ from Categorical$(\frac{1}{p^+_{\text{sum}}}(p^+(0), p^+(1), \cdots, p^+(\tilde{d}^+_{\max})))$;

8 $\quad$ **for** $j = 0$ *to* $\tilde{d}^-_{\max}$ **do**

9 $\quad\quad p^-(j) \leftarrow \exp(-\frac{\epsilon|\min\{d^-_{t,i}, \tilde{d}^-_{\max}\} - j|}{2\tilde{d}^-_{\max}})$;

10 $\quad$ Sample $\hat{d}^-_{t,i}$ from Categorical$(\frac{1}{p^-_{\text{sum}}}(p^+(0), p^+(1), \cdots, p^+(\tilde{d}^-_{\max})))$;

11 $\quad$ Report $\hat{d}^+_{t,i}$ and $\hat{d}^-_{t,i}$ to the central server;

// On the central sever side

12 Aggregate $\hat{d}^+_t = (\hat{d}^+_{t,1}, \hat{d}^+_{t,2}, \cdots, \hat{d}^+_{t,n})$;

13 Aggregate $\hat{d}^-_t = (\hat{d}^-_{t,1}, \hat{d}^-_{t,2}, \cdots, \hat{d}^-_{t,n})$;

14 **return** $\hat{d}_t = (\hat{d}^+_t, \hat{d}^-_t)$;

As shown in Algorithm 1, the algorithm first runs on each user side. For each user v_i, v_i first obtains the out degree $d^+_{t,i}$ and in degree $d^-_{t,i}$, respectively (Lines 2, 3). Then, user v_i implement the exponential mechanism, for all $j \in [0, \tilde{d}^+_{\max}]$, v_i computes the weight $p^+(j)$ and then obtains p^+_{sum} by summing all $p^+(j)$ (Lines 4-6). Then, the algorithm obtain the estimated out degree $\hat{d}^+_{t,i}$ sampled from the categorical distribution generated by all the weights (Line 7). Next, user v_i implement the exponential mechanism, once again, for all $j \in [0, \tilde{d}^-_{\max}]$, v_i computes the weight $p^-(j)$ and then obtains p^+_{sum} by summing all $p^-(j)$ (Lines 8-10). Then, the algorithm obtain the estimated out degree $\hat{d}^-_{t,i}$ sampled from the categorical distribution generated by all the weights (Line 10). After obtain the estimated out degree $\hat{d}^+_{t,i}$ and $\hat{d}^-_{t,i}$, user v_i reports them to the central server (Line 11). Next, the algorithm runs on the central server side, the central server aggregates all the estimated out degrees and in degrees reported from each user and obtains the estimated degree list $\hat{d}_t = (\hat{d}^+_t, \hat{d}^-_t)$ (Lines 12-14).

Complexity. On each user side, Algorithm 1 obtains all the weights $p^+(j)$, p^+_{sum}, and the estimated out degree $\hat{d}^+_{t,i}$. It takes $O(\tilde{d}^+_{\max})$ time. Similarly, the algorithm obtains all the weights $p^-(j)$, p^-_{sum}, and the estimated out degree $\hat{d}^-_{t,i}$. It takes $O(\tilde{d}^-_{\max})$ time. Therefore, on the user side, the computation complexity is $O(\tilde{d}^+_{\max} + \tilde{d}^-_{\max})$. On the central server, the central server takes $O(n)$ time to aggregate all the reported out degrees and in degrees. Each user should report his/her estimated out degree and in degree to the central server. Thus, the communication complexity of this process is $O(n)$.

Algorithm 2: Post Processing

Input : The estimated degree list $\hat{d}_t = (\hat{d}_t^+, \hat{d}_t^-)$; the number of vertices n

Output : The post processing degree list $\overline{d}_t = (\overline{d}_t^+, \overline{d}_t^-)$

Find $\sigma \in S_n$, s.t. $\hat{d}^-_{t,\sigma(1)} \geq \hat{d}^-_{t,\sigma(2)} \geq \cdots \geq \hat{d}^-_{t,\sigma(n)}$;
$\overline{d}_t \leftarrow ((\hat{d}^+_{t,\sigma(1)}, \hat{d}^+_{t,\sigma(2)}, \cdots, \hat{d}^+_{t,\sigma(n)}), (\hat{d}^-_{t,\sigma(1)}, \hat{d}^-_{t,\sigma(2)}, \cdots, \hat{d}^-_{t,\sigma(n)}))$;
Queue $Q \leftarrow \emptyset$;
for $k = n-1$ *to* 1 **do**
  if $\overline{d}^-_{t,k} == \overline{d}^-_{t,k+1}$ **then**
    continue;
  $l \leftarrow \sum_{i=1}^{k} \min\{\overline{d}^+_{t,i}, k-1\} + \sum_{i=k+1}^{n} \min\{\overline{d}^+_{t,i}, k\}$;
  $r \leftarrow \sum_{i=1}^{k} \overline{d}^-_{t,i}$;
  if $l \leq r$ **then**
    $\overline{d}^-_{t,k} \leftarrow \overline{d}^-_{t,k+1}$;
  else
    $Q \leftarrow Q \cup \{k\}$;
$\overline{d}^+_{t,\text{sum}} \leftarrow \sum_{i=1}^{n} \overline{d}^+_{t,i}$, $\overline{d}^-_{t,\text{sum}} \leftarrow \sum_{i=1}^{n} \overline{d}^-_{t,i}$;
if $\overline{d}^+_{t,\text{sum}} > \overline{d}^-_{t,\text{sum}}$ **then**
  while $Q \neq \emptyset$ **do**
    Pop k from Q;
    for $i = 1$ *to* n **do**
      $\tau \leftarrow k - \mathbb{I}(i \leq k)$;
      if $\overline{d}^+_{t,i} > \tau$ **then**
        $\Delta \leftarrow \overline{d}^+_{t,i} - \tau$;
        if $\Delta \geq \overline{d}^+_{t,\text{sum}} - \overline{d}^-_{t,\text{sum}}$ **then**
          $\overline{d}^+_{t,i} \leftarrow \overline{d}^+_{t,i} - (\overline{d}^+_{t,\text{sum}} - \overline{d}^-_{t,\text{sum}})$, $\overline{d}^+_{t,\text{sum}} \leftarrow \overline{d}^-_{t,\text{sum}}$;
          break while;
        else
          $\overline{d}^+_{t,i} \leftarrow \tau$, $\overline{d}^+_{t,\text{sum}} \leftarrow \overline{d}^+_{t,\text{sum}} - \Delta$;
  if $\overline{d}^+_{t,\text{sum}} > \overline{d}^-_{t,\text{sum}}$ **then**
    for $i = 1$ *to* n **do**
      if $\overline{d}^+_{t,i} \geq \overline{d}^+_{t,\text{sum}} - \overline{d}^-_{t,\text{sum}}$ **then**
        $\overline{d}^+_{t,i} \leftarrow \overline{d}^+_{t,i} - (\overline{d}^+_{t,\text{sum}} - \overline{d}^-_{t,\text{sum}})$, $\overline{d}^+_{t,\text{sum}} \leftarrow \overline{d}^-_{t,\text{sum}}$;
        break;
      else
        $\overline{d}^+_{t,\text{sum}} \leftarrow \overline{d}^+_{t,\text{sum}} - \overline{d}^+_{t,i}$, $\overline{d}^+_{t,i} \leftarrow 0$;
else if $\overline{d}^+_{t,\text{sum}} < \overline{d}^-_{t,\text{sum}}$ **then**
  for $i = 1$ *to* n **do**
    $\Delta \leftarrow (n-1) - \overline{d}^+_{t,i}$;
    if $\Delta \geq \overline{d}^-_{t,\text{sum}} - \overline{d}^+_{t,\text{sum}}$ **then**
      $\overline{d}^+_{t,i} \leftarrow \overline{d}^+_{t,i} + (\overline{d}^-_{t,\text{sum}} - \overline{d}^+_{t,\text{sum}})$, $\overline{d}^+_{t,\text{sum}} \leftarrow \overline{d}^-_{t,\text{sum}}$;
      break;
    else
      $\overline{d}^+_{t,i} \leftarrow n-1$, $\overline{d}^+_{t,\text{sum}} \leftarrow \overline{d}^+_{t,\text{sum}} + \Delta$;
return $((\overline{d}^+_{t,\sigma^{-1}(1)}, \overline{d}^+_{t,\sigma^{-1}(2)}, \cdots, \overline{d}^+_{t,\sigma^{-1}(n)}), (\overline{d}^-_{t,\sigma^{-1}(1)}, \overline{d}^-_{t,\sigma^{-1}(2)}, \cdots, \overline{d}^-_{t,\sigma^{-1}(n)}))$;

4.4 Post Processing

Notice that, the result $\hat{d}_t$ returned from **Phase 2** may not be digraphic, i.e., there exists a directed graph G contains n vertices, such that G's degree list is $\hat{d}_t$. The post processing phase aims to repair estimated degree list digraphic. After the central server aggregates the reported degrees, we employ post processing phase on the central server side.

To make the degree list digraphic, we use the discrete mathematics result of [5],

Theorem 4.5. *Given a sequence* $d = ((d_1^+, d_2^+, \cdots, d_n^+), (d_1^-, d_2^-, \cdots, d_n^-)) \in \mathbb{Z}^n_{\geq 0} \times \mathbb{Z}^n_{\geq 0}$, *such that* $d_1^- \geq d_2^- \geq \cdots \geq d_n^-$, d *is digraphic if and only if*

- $\sum_{i=1}^{n} d_i^+ = \sum_{i=1}^{n} d_i^-$.
- $\forall k \in \{k : 1 \leq k \leq n \wedge (d_k^- > d_{k+1}^- \vee k = n)\}$, $\sum_{i=1}^{k} \min\{d_i^+, k-1\} + \sum_{i=k+1}^{n} \min\{d_i^+, k\} \geq \sum_{i=1}^{k} d_i^-$.

Theorem 4.5 states a sufficient and necessary condition, thus we only need to make the degree list satisfy this condition, then the degree list will be digraphic. Observe this condition, it has two constraint. Similar to the handshaking lemma in undirected graphs, the first constraint is trivial. Therefore, in our algorithm design, we consider that how to establish the connection between two constraints. Assume that the first constraint is satisfied, we find why the second constraint may not be satisfied. The answer is that for some k, such that $d_k^- > d_{k+1}^-$, there exists $i \in [k]$, such that $d_i^+ > k-1$ or $i \in [k+1, n]$, such that $d_i^+ > k$. These elements are regarded as $k-1$ or k while calculating the left hand of the second constraint. The way to make the degree list satisfied the second constraint is to broken these elements. Then, the degree list only need to satisfied the first constraint, then the condition will be satisfied.

Therefore, in our algorithm, we first break all the $\overline{d}^-_{t,k}$, such that $\overline{d}^-_{t,k} > \overline{d}^-_{t,k+1}$ and $\sum_{i=1}^{k} \min\{\overline{d}^+_{t,i}, k-1\} + \sum_{i=k+1}^{n} \min\{\overline{d}^+_{t,i}, k\} < \sum_{i=1}^{k} \overline{d}^-_{t,i}$. We only need to set $\overline{d}^-_{t,k} = \overline{d}^-_{t,k+1}$. Notice that we consider k from $n-1$ to 1, the reason is that the order of in degree list should be maintained. Then, we make the degree list $\overline{d}_t$ satisfy the first constraint. We consider two cases, (1) the left hand of the first constraint is strictly lower than the right hand; (2) the left hand of the second constraint is strictly larger than the right hand. For case (1), we will reduce some $\overline{d}^+_{t,i}$ to make the equation hold. Mind that when we reduce some $\overline{d}^+_{t,i}$, the second constraint cannot be broken. Therefore, for a fixed $k \in [n-1]$, such that $\overline{d}^-_{t,k} > \overline{d}^-_{t,k+1}$, we only reduce $\overline{d}^+_{t,i}$, such that $(i \in [k]) \wedge (\overline{d}^+_{t,i} > k-1)$ or $i \in [k+1, n] \wedge \overline{d}^+_{t,i} > k$. Next, we introduce how to find these $\overline{d}^+_{t,i}$. Observe that for $k_1, k_2 \in [n-1]$, such that $k_1 < k_2$, $\overline{d}^-_{t,k_1} > \overline{d}^-_{t,k_1+1}$, and $\overline{d}^+_{t,k_2} > \overline{d}_{t,k_2+1}$, if for some $i \in [k_2]$, $\overline{d}^+_{t,i} > k_2 - 1$, then $\overline{d}^+_{t,i} > k_1$; if for some $i \in [k_2+1, n]$, $\overline{d}^+_{t,i} > k_2$, then $\overline{d}^+_{t,i} > k_1$. Thus, for these $i \in [k_1]$, $\overline{d}^+_{t,i} > k_2 - 1 > k_1 - 1$; for these $i \in [k_2+1, n]$, $\overline{d}^+_{t,i} > k_2 > k_1$; for these $i \in [k_1+1, k_2]$, $\overline{d}^+_{t,i} > k_2 - 1 \geq k_1$. Based on this observation, to find these $\overline{d}^+_{t,i}$, we enumerate k, such that $\overline{d}^-_{t,k_1} > \overline{d}^-_{t,k_1+1}$ and $\sum_{i=1}^{k} \min\{\overline{d}^+_{t,i}, k-1\} + \sum_{i=k+1}^{n} \min\{\overline{d}^+_{t,i}, k\} \geq \sum_{i=1}^{k} \overline{d}^-_{t,i}$ from the largest to the smallest. However, there still exists a concern. For a fixed k, if for all $i \in [k]$, $\overline{d}^+_{t,i} \leq k-1$, and for all $i \in [k+1, n]$, $\overline{d}^+_{t,i} \leq k$, but we will find $\overline{d}^+_{t,i}$ can be reduced while dealing with a smaller k', then the concern is whether the inequality $\sum_{i=1}^{k} \min\{d_i^+, k-1\} + \sum_{i=k+1}^{n} \min\{d_i^+, k\} \geq \sum_{i=1}^{k} d_i^-$ is still satisfied. Our algorithm does not break this inequality, because for a fixed k, we will not consider a smaller k' unless there is no $\overline{d}^+_{t,i}$ that can be reduced. If for a fixed k, we reduced all the $\overline{d}_{t,i}$, then the satisfaction of the first constraint implies that inequality $\sum_{i=1}^{k} \min\{\overline{d}^+_{t,i}, k-1\} + \sum_{i=k+1}^{n} \min\{\overline{d}^+_{t,i}, k\} \geq \sum_{i=1}^{k} \overline{d}^-_{t,i}$ holds. For

case (2), we will increase some $\overline{d}^{+}_{t,i}$ to make the equation hold. We choose some $\overline{d}^{+}_{t,i}$, such that $\overline{d}^{+}_{t,i} < n-1$ for increasing. Since increasing $\overline{d}^{+}_{t,i}$ cannot break the second constraint, we choose $\overline{d}^{+}_{t,i}$ following the order of i.

As shown in Algorithm 2, the post processing algorithm first sorts the vertices in non-increasing order of their in degrees by finding a permutation $\sigma \in S_n$ such that $\hat{d}^{-}_{t,\sigma(1)} \geq \hat{d}^{-}_{t,\sigma(2)} \geq \cdots \geq \hat{d}^{-}_{t,\sigma(n)}$. Here, S_n is the symmetric group of n letters. (Line 1) Then, the algorithm rearranges the degree list according to this permutation and obtain $\overline{d}_t$ (Line 2). Next, the algorithm initializes an empty queue Q (Line 3). For each k from $n-1$ to 1, if the in degree at position k equals the in-degree at position $k+1$, it continues to the next k. Otherwise, it computes the left hand l and right hand r of the inequality in the second constraint. If $l < r$, the algorithm decreases the in degree at position k to match the in degree at position $k+1$ (Line 10). Otherwise, it pushes k into the queue Q (Line 12). After processing all k, the algorithm calculates the sum of all the out degrees $\overline{d}^{+}_{t,\text{sum}}$ and the sum of all the in degrees $\overline{d}^{-}_{t,\text{sum}}$ (Line 13). If $\overline{d}^{+}_{t,\text{sum}}$ exceeds $\overline{d}^{-}_{t,\text{sum}}$, it processes the queue Q to reduce out degrees. For each k popped from Q, it iterates through vertices and reduces their out-degrees to at most k (if the vertex index $\leq k$) or $k-1$ (if the vertex index $> k$), ensuring the $\overline{d}^{-}_{t,\text{sum}}$ is equal to the $\overline{d}^{+}_{t,\text{sum}}$. (Lines 14-25) If the queue is exhausted and the $\overline{d}^{+}_{t,\text{sum}}$ still exceeds $\overline{d}^{-}_{t,\text{sum}}$, it further reduces out degrees sequentially starting from the first vertex until the totals match. (Lines 26-32) If the $\overline{d}^{+}_{t,\text{sum}}$ is less than $\overline{d}^{-}_{t,\text{sum}}$, the algorithm increases out-degrees sequentially, raising each vertex's out degree up to $n-1$ until the $\overline{d}^{-}_{t,\text{sum}}$ equals $\overline{d}^{-}_{t,\text{sum}}$ (Lines 33-40). Finally, the algorithm restores the original vertex order by applying the inverse permutation σ^{-1} to the degree list and returns the adjusted degree list (Line 41).

Complexity. We analyze the time complexity of post processing phase. Firstly, the algorithm sorted the degree list $\hat{d}_t$ by using the key of positive degree, which takes $O(n \log n)$ time. Then, the algorithm takes $O(n)$ time to enumerate k. Notice that only for $k \in [n-1]$, such that $\overline{d}^{-}_{t,k} > \overline{d}^{-}_{t,k+1}$, the loop body will be executed. Thus, the loop body will be executed at most $\tilde{d}^{-}_{\max} - 1$ times. Each iteration takes $O(n)$ time, leading to the whole loop takes $O(n\tilde{d}^{-}_{\max})$ time. Then, the algorithm reduce or increase some $\overline{d}^{+}_{t,i}$. This process takes at most $O(n\tilde{d}^{-}_{\max})$ time, since the cardinality $|Q|$ is at most $\tilde{d}^{-}_{\max} - 1$. Finally, the algorithm applying the inverse permutation σ^{-1} to restore the original vertex order, which takes $O(n)$ time. Therefore, the whole algorithm takes $O(n \log n + n\tilde{d}^{-}_{\max})$ time on the central server side. Here, $\tilde{d}^{-}_{\max}$ is the projection threshold used for in neighbor lists on each user side.

4.5 Overall Algorithm

Given a streaming directed graph $\mathbf{G} = (G_1, G_2, \cdots, G_T)$ or $\mathbf{G} = (G_1, G_2, \cdots)$ represented as neighbor lists $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n$, projection thresholds $\tilde{d}^{+}_{\max}$ and $\tilde{d}^{-}_{\max}$, and window length w, overall privacy budget ϵ. We apply **Phase 1**, **Phase 2**, and **Phase 3** to the timestamp G_t with privacy budget ϵ/w at each timestamp t.

As shown in Algorithm 3, the algorithm first initialize the timestamps $t = 1$ (Line 1). Then, at each timestamp t, the algorithm

Algorithm 3: Overall Algorithm

Input : Streaming directed graph $\mathbf{G}$ represented as $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n$; the thresholds $\tilde{d}^{+}_{\max}$ and $\tilde{d}^{-}_{\max}$; the number of users n; window length w; privacy budget $\epsilon \in \mathbb{R}_{\geq 0}$

Output : Released degree lists $\overline{d}_t = (\overline{d}^{+}_t, \overline{d}^{-}_t), \forall t \in [T]$

1 $t \leftarrow 1$;
2 **while** G_t *exists* **do**
3 Estimated Degree list $\hat{d}_t \leftarrow$Degree Collection(G_t, $\tilde{d}^{+}_{\max}$, $\tilde{d}^{-}_{\max}$,n , ϵ/w);
4 Post processing degree list $\overline{d}_t \leftarrow$Post Processing($\hat{d}_t$, n);
5 **Output** $\overline{d}_t$;
6 $t \leftarrow t+1$;

deals with the snapshot of the input streaming directed graph $\mathbf{G}$ at timestamp t, i.e., G_t. When dealing with a snapshot G_t, the algorithm runs Algorithm 1 to obtain the estimated degree list $\hat{d}_t$ (Line 3). Then, the algorithm runs Algorithm 2 to obtain the post processing degree list $\overline{d}_t$ (Line 4). Finally, the algorithm outputs the post processing degree list $\overline{d}_t$ at timestamp t (Line 5).

Theorem 4.6. *Algorithm 3 satisfies w-event local differential privacy for each user with privacy budget ϵ.*

Complexity. Due to the complexity of Algorithm 1 and Algorithm 2, at each timestamp t, the overall algorithm takes $O(\tilde{d}^{+}_{\max} + \tilde{d}^{-}_{\max})$ time on each user side, and $O(n)$ time on the central server side. Besides, the communication cost is $O(n)$ at each timestamp t.

5 OPTIMIZED ALGORITHM

In this section, we propose a optimized algorithm for degree list collection in streaming directed graphs satisfying w-event local differential privacy. Compared to the collection algorithm, there are two optimizations. The first optimization is solving the over-smoothing problem of the exponential mechanism. Moreover, the second optimization is improving the efficiency by using the correlation of different timestamps of streaming graphs.

5.1 Beyond Over-Smoothing

Given a discrete probability distribution $\mathcal{P}$, if it has too many possible values, then each value is likely to correspond to a very small probability. For example, let $\mathcal{P}$ be a uniform distribution with n possible values, so that the probability of each value is $1/n$, which is very small when n is large. When n is equal to 10000, the probability of each value is only 0.0001. We refer to this phenomenon as over-smoothing. Notice that, in the exponential mechanism, the distribution of the algorithmic output has $|\mathcal{R}|$ possible values. Here, $|\mathcal{R}|$ is the cardinality of the output space. If the output space $\mathcal{R}$ contains too many elements, then the output distribution of the algorithm may become over-smoothed, exhibiting the over-smoothing phenomenon. The over-smoothing phenomenon may reduce the utility of the algorithmic output, since each element in the output space is selected with approximately equal probability.

In fact, in our collection algorithm, the output space of the exponential mechanisms are $[0, \tilde{d}^{+}_{\max}]$ and $[0, \tilde{d}^{-}_{\max}]$, respectively. Here, $\tilde{d}^{+}_{\max}$ and $\tilde{d}^{-}_{\max}$ are the projection thresholds. Although $\tilde{d}^{+}_{\max}$ and $\tilde{d}^{-}_{\max}$ are both much less than the number of vertices, the output spaces $[0, \tilde{d}^{+}_{\max}]$ and $[0, \tilde{d}^{-}_{\max}]$ are still large, especially for input streaming graph containing a number of vertices. To beyond the

over-smoothing phenomenon, we use a step size θ. For a fixed $\theta \in \mathbb{Z}_{\geq 1}$, we set the output spaces of the exponential mechanisms as $\{i\theta : 0 \leq i \leq \tilde{d}^+_{\max}/\theta, i \in \mathbb{Z}\}$ and $\{i\theta : 0 \leq i \leq \tilde{d}^-_{\max}/\theta, i \in \mathbb{Z}\}$, respectively. Therefore, the output spaces are reduced, thereby mitigating the over-smoothing phenomenon and enhancing the utility of the algorithmic output utility.

Note that the domain of the utility functions $\tilde{f}$ and $\tilde{f}$ have been changed, since we introduce the step size θ in the algorithm. We consider the new utility functions $\tilde{f}^+_\theta$ and $\tilde{f}^-_\theta$. The domain of $\tilde{f}^+_\theta$ is $\{\mathbf{a} \subseteq \{0,1\}^n : \sum_{j=1}^n a_j \leq \tilde{d}^+_{\max}\} \times \{i\theta : 0 \leq i \leq \tilde{d}^+_{\max}/\theta, i \in \mathbb{Z}\}$. Similarly, the domain of $\tilde{f}^-$ is $\{\mathbf{a} \subseteq \{0,1\}^n : \sum_{j=1}^n a_j \leq \tilde{d}^-_{\max}\} \times \{i\theta : 0 \leq i \leq \tilde{d}^-_{\max}/\theta, i \in \mathbb{Z}\}$. Therefore, to use the exponential mechanism correctly, we calculate the global sensitivity of the f^+_θ and f^-_θ, respectively.

Proposition 5.1. *The global sensitivities of the $\tilde{f}^+_\theta$ and $\tilde{f}^-_\theta$ are at most $\tilde{d}^+_{\max}$ and $\tilde{d}^-_{\max}$, respectively.*

5.2 Efficiency Optimization

In streaming directed graphs, snapshots at different timestamps share a lot of same edges. In fact, at each timestamp, part of edges will be updated. However, there are still a number of edges not changed. To reduce the computation cost of the collection algorithm by using this observation, we propose a efficiency optimization technique. Then, each user only updates the estimated out degree and estimated in degree when the degree is significant at a timestamp.

Notice that in a streaming directed graphs, the out degree and in degree of a vertex are often stable, i.e., the out degree and the in degree will not be significant changed at different timestamps. Besides, these degrees are generally neither too large nor too small. Therefore, in the locally differentially private degree list collection, the estimations of these degrees do not need to update at every timestamp. At each timestamp, we only need to update the significant degrees by using the exponential mechanism. Therefore, the computation cost of the exponential mechanisms is reduced.

Inspired by the sparse vector technique [17, 18, 33], at each timestamp, each user is required to determine whether the estimated out degree and the estimated in degree should be updated, respectively. Notice that determining the operation of update (denoted by $\top$) or the operation of maintain (denoted by $\perp$) should also satisfy w-event local differential privacy. Otherwise, the privacy information might be leakage. To demonstrate why this matters, we show an incorrect approach and explain how privacy leaks can occur: each user v_i use two fixed thresholds η_1 and η_2, such that $\eta_1, \eta_2 \in \mathbb{Z}_{\geq 0}$ and $\eta_1 < \eta_2$, then at timestamp t, v_i determines that the estimated out degree (resp. estimated in degree) should be updated if and only if $d^+_{t,i} \leq \eta_1$ or $d^+_{t,i} \geq \eta_2$ (resp. $d^-_{t,i} \leq \eta_1$ or $d^-_{t,i} \geq \eta_2$). This is a deterministic algorithm and does not satisfy w-event local differential privacy. Then, we explain that adopting the $\top/\perp$ determination from this algorithm causes the degree list collection algorithm proposed in Section 4 to violate the w-event local differential privacy for user v_i. Let $\mathcal{A}_{\text{error}}$ denote this error collection algorithm. Consider v_i's two possible neighbor lists $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, such that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is w-neighboring, and

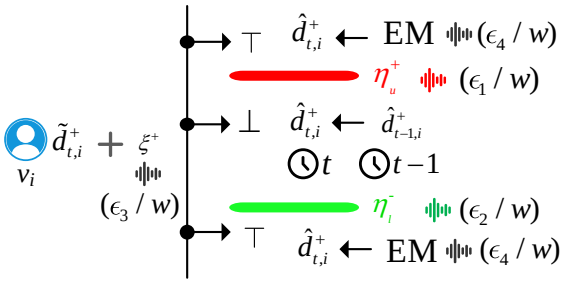


Figure 4: The determination process workflow

- $\forall t \in \mathcal{I}, d^+(t, \boldsymbol{\alpha}) \geq \eta_2$,
- $\forall t \in \mathcal{I}, d^-(t, \boldsymbol{\alpha}) \geq \eta_2$,
- $\forall t \in \mathcal{I}, \eta_1 < d^+(t, \boldsymbol{\beta}) < \eta_2$,
- $\forall t \in \mathcal{I}, \eta_1 < d^-(t, \boldsymbol{\beta}) < \eta_2$.

Here, $\mathcal{I} = \{t : \boldsymbol{\alpha}_t \neq \boldsymbol{\beta}_t\}$. d^+ and d^- denote the out degree function and in degree function, respectively. Given a neighbor list $\mathbf{u}_i$ and timestamp t, the out degree function (resp. in degree function) maps $(\mathbf{u}_i, t)$ to the out degree (resp. in degree) of v_i at timestamp t. Next, let $l = \min \mathcal{I}$, consider the algorithm $\mathcal{A}_{\text{error}}$, and take $O = \{o \in \text{Range}(\mathcal{A}_{\text{error}}) : \exists t \in \mathcal{I}, o^+_{t,i} \neq o^+_{t,l-1} \vee o^-_{t,i} \neq o^-_{t,l-1}\}$ (o_t denote the released degree list $\bar{d}_t$ by $\mathcal{A}_{\text{error}}$ at timestamp t) , then

$$\Pr[\mathcal{A}_{\text{error}}(\boldsymbol{\alpha}) \in O] = 0,$$
$$\Pr[\mathcal{A}_{\text{error}}(\boldsymbol{\beta}) \in O] > 0.$$

Consider the constraint in Definition 3.2, if $\mathcal{A}_{\text{error}}$ satisfies w-event differential privacy,

$$\Pr[\mathcal{A}_{\text{error}}(\boldsymbol{\beta}) \in O] \leq e^\epsilon \Pr[\mathcal{A}_{\text{error}}(\boldsymbol{\alpha}) \in O] = 0.$$

A contradiction arises, thus $\mathcal{A}_{\text{error}}$ does not satisfy w-event differential privacy.

Therefore, to make the optimized collection algorithm satisfy w-event local differential privacy for each user, we must consume a little privacy budget to determine whether the estimated out degree and estimated in degree should be updated. At each timestamp t, for each user v_i, we determine in turn whether to update the estimated out degree and the estimated in degree. For the determination of estimated out degree, we use three random variables, η^+_u, η^+_l, and ξ^+, which follow Laplace distributions with location parameter 0 and scale parameters $\frac{w\tilde{d}^+_{\max}}{\epsilon_1}$, $\frac{w\tilde{d}^+_{\max}}{\epsilon_2}$, and $\frac{2w\tilde{d}^+_{\max}}{\epsilon_3}$, respectively. Here, w is the window length, $\tilde{d}^+_{\max}$ is the projection threshold used in the neighbor list projection, and $\epsilon_1, \epsilon_2, \epsilon_3$ are three separate privacy budgets.

As shown in Fig. 4, the decision whether to update $\hat{d}^+_{t,i}$ via the exponential mechanism or to retain the prior value $\hat{d}^+_{t-1,i}$ is made as follows: after the neighbor list projection, the user has the out degree $\tilde{d}^+_{t,i}$ (i.e., $\min\{d^+_{t,i}, \tilde{d}^+_{\max}\}$). The algorithm first generates random thresholds η^+_u and η^+_l, respectively. Meanwhile, the privacy budgets ϵ_1/w and ϵ_2/w are consumed, respectively. Then, the algorithm generates the Laplace noise ξ^+ and adds it to the out degree $\tilde{d}^+_{t,i}$, which consumes the privacy budget ϵ_3/w. Then, the algorithm compares $\tilde{d}^+_{t,i} + \xi^+$ with the thresholds η^+_u and η^+_l, respectively. If

Algorithm 4: Optimized Algorithm

Input : Streaming directed graph $\mathbf{G}$ represented as $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n$; the thresholds $\tilde{d}^+_{\max}$ and $\tilde{d}^-_{\max}$; the number of users n; window length w; the step size θ; privacy budget $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4 \in \mathbb{R}_{\geq 0}$

Output : Released degree lists $\overline{d}_t = (\overline{d}^+_t, \overline{d}^-_t), \forall t \in [T]$

1 $t \leftarrow 1$;
2 **while** G_t *exists* **do**
  // On each user side
3 **for** $i = 1$ *to* n **do**
4 **for** *sign* s *in* $\{+, -\}$ **do**
5 $\eta^s_u \leftarrow \text{Lap}(\frac{w\tilde{d}^s_{\max}}{\epsilon_1})$, $\eta^s_l \leftarrow \text{Lap}(\frac{w\tilde{d}^s_{\max}}{\epsilon_2})$;
6 $\xi^s \leftarrow \text{Lap}(\frac{2w\tilde{d}^s_{\max}}{\epsilon_3})$;
7 $\chi^s \leftarrow \min\{d^s_{t,i}, \tilde{d}^s_{\max}\} + \xi^s$;
8 **if** $\chi^s \leq \eta^s_l$ *or* $\chi^s \geq \eta^s_u$ **then**
9 Let $c \leftarrow 0$;
10 **for** $j = 0$ *to* $\tilde{d}^s_{\max}$ **step** θ **do**
11 $p^s(j) \leftarrow \exp(-\frac{\epsilon_4 |\min\{d^s_{t,i}, \tilde{d}^s_{\max}\} - j|}{2w\tilde{d}^s_{\max}})$;
12 $c \leftarrow c + 1$;
13 $p^s_{\text{sum}} \leftarrow \sum_{j=0}^{c-1} p^s(j\theta)$;
14 Sample $\hat{d}^+_{t,i}$ from Categorical$(\frac{1}{p^+_{\text{sum}}}(p^+(0), p^+(\theta), \cdots, p^+((c-1)\theta)))$;
15 Report $\hat{d}^s_{t,i}$ to the central server;
16 **else**
  // Define $\hat{d}^s_{t,0} = 0$
17 $\hat{d}^s_{t,i} \leftarrow \hat{d}^s_{t-1,i}$;
18 Report $\hat{d}^s_{t,i}$ to the central server;
  // On the central server side
19 Aggregate $\hat{d}^+_t = (\hat{d}^+_{t,1}, \hat{d}^+_{t,2}, \cdots, \hat{d}^+_{t,n})$;
20 Aggregate $\hat{d}^-_t = (\hat{d}^-_{t,1}, \hat{d}^-_{t,2}, \cdots, \hat{d}^-_{t,n})$;
21 Post processing degree list $\overline{d}_t \leftarrow$Post Processing$(\hat{d}_t)$;
22 Output $\overline{d}_t$;
23 $t \leftarrow t + 1$;

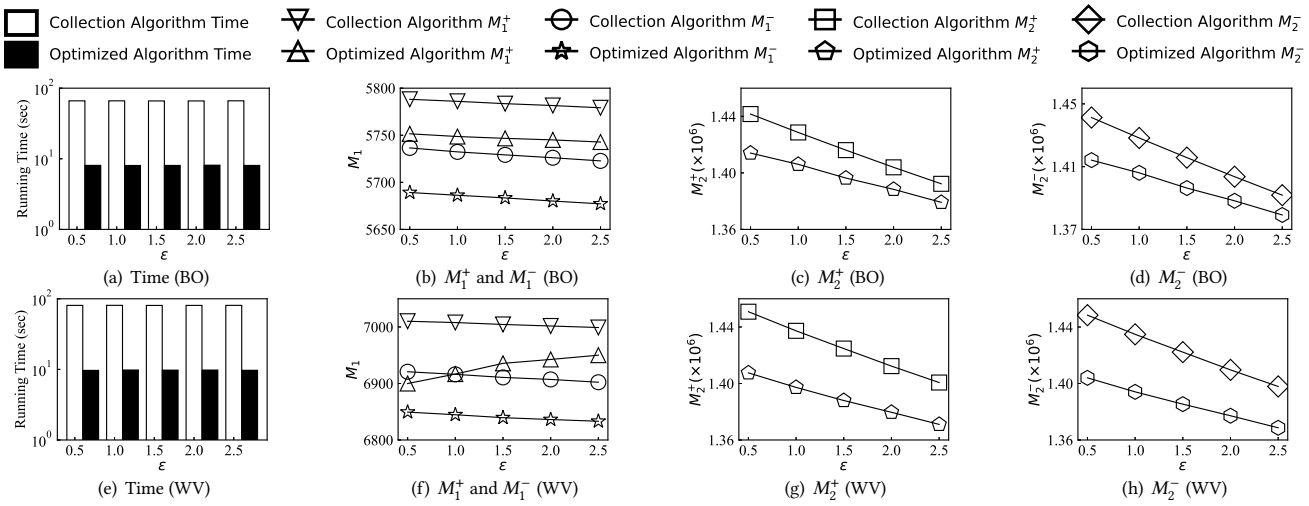
$\tilde{d}^+_{t,i} + \xi^+ \geq \eta^+_u$ or $\tilde{d}^+_{t,i} + \xi^+ \leq \eta^-_l$, the algorithm will employ exponential mechanism to update the estimated out degree $\hat{d}^+_{t,i}$, which consumes the privacy budget ϵ_3/w. Otherwise $\eta^+_l < \tilde{d}^+_{t,i} < \eta^+_u$, the algorithm maintain the estimated out degree $\hat{d}^+_{t,i}$ as $\hat{d}^+_{t-1,i}$. The decision whether to update $\hat{d}^-_{t,i}$ via the exponential mechanism or to retain the prior value $\hat{d}^-_{t-1,i}$ follows the same rule as that for $\hat{d}^+_{t,i}$.

As shown in Algorithm 4, the algorithm first initializes the time step t to 1 (Line 1). Then, for each existing graph G_t at timestamp t, the algorithm performs operations on both the user side and the central server side. On the user side, for each user v_i, $i \in [n]$, and for each sign $s \in \{+, -\}$ (s is used to denote the variables related to out degree and in degree, respectively), the algorithm first determines the values of the random variables η^s_u, η^s_l, and ξ^s through sampling, with scales based on the window length w, the threshold used for neighbor list projection $\tilde{d}^s_{\max}$, and the separate privacy budgets ϵ_1, ϵ_2, ϵ_3 (Lines 5, 6). Then, it computes the noisy degree χ^s as the degree after the neighbor list projection $\min d^s_{t,i}, \tilde{d}^s_{\max}$ plus the noise ξ^s (Line 7). Next, the algorithm checks whether χ^s satisfies $\chi^s \leq \eta^s_l$ or $\chi^s \geq \eta^s_u$ (Line 8). If so, it prepares a categorical distribution by initializing c as 0 (Line 9), and for $j \in \{k\theta : 0 \leq k \leq \tilde{d}^s_{\max}/\theta, k \in \mathbb{Z}\}$, it computes the weight $p^s(j) \leftarrow \exp(-\frac{\epsilon_4 |\min\{d^s_{t,i}, \tilde{d}^s_{\max}\} - j|}{2w\tilde{d}^s_{\max}})$ and updates c (Lines 11, 12). After obtain the sum of all the weights $p^s_{\text{sum}} = \sum_{j=0}^{c-1} p^s(j\theta)$ (Line 13), the algorithm samples the estimated degree $\hat{d}^s_{t,i}$ from the distribution Categorical$(\frac{1}{p^+_{\text{sum}}}(p^+(0), p^+(\theta), \cdots, p^+((c-1)\theta)))$ and reports it to the central server (Lines 14, 15). Otherwise, if $\eta^s_l < \chi^s < \eta^s_u$, the algorithm sets the estimated degree $\hat{d}^s_{t,i}$ as the previous estimated degree $\hat{d}^s_{t-1,i}$ (Lines 17, 18). On the central server side, the algorithm aggregates all reported out degrees to aggregate the estimated out degree lists $\hat{d}^+_t = (\hat{d}^+_{t,1}, \hat{d}^+_{t,2}, \cdots, \hat{d}^+_{t,n})$ (Line 19). Similarly, the algorithm aggregates all reported in degrees to aggregate the estimated in degree lists $\hat{d}^-_t$ in-degree lists $\hat{d}^-_t = (\hat{d}^-_{t,1}, \hat{d}^-_{t,2}, \cdots, \hat{d}^-_{t,n})$ (Line 20). Then, it applies a post-processing algorithm (i.e. Algorithm 2) to adjust the degree list, resulting in $\overline{d}t$ (Line 21). Finally, the algorithm outputs $\overline{d}_t$ and increments the timestamps t (Lines 22, 23).

Theorem 5.2. *Algorithm 4 satisfies w-event local differential privacy for each user with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4)$.*

Complexity. We analyze the complexity of the optimized algorithm. For each timestamp t, on each user v_i's side, Algorithm 4 first determines whether the estimated out degree should be updated, which takes $O(1)$ time. Then, if the determination is $\perp$, i.e., the estimated out degree $\hat{d}^+_{t,i}$ retains the prior estimated out degree $\hat{d}^+_{t-1,i}$, it only takes $O(1)$ time. Otherwise, if the determination is $\top$, i.e., the estimated out-degree $\hat{d}^+_{t,i}$ should be updated via the exponential mechanism. Then, the algorithm uses the exponential mechanism. It should be noted that the step size θ can influence the output space of the exponential mechanism. The algorithm takes $O(\tilde{d}^+_{\max}/\theta)$ time to obtain all the weights, and then takes $O(1)$ time to sample the estimated out degree $\hat{d}^+_{t,i}$ from the categorical distribution. Therefore, at each timestamp t, on each user v_i's side, the algorithm takes at most $O(\tilde{d}^+_{\max}/\theta)$ time to obtain the estimated out degree. Similarly, the algorithm takes at most $O(\tilde{d}^-_{\max}/\theta)$ time to obtain the estimated in degree. Thus, at each timestamp t, the algorithm takes $O((\tilde{d}^+_{\max} + \tilde{d}^-_{\max})/\theta)$ time on each user v_i's side. On the central server side, the central server takes $O(n)$ time to aggregate all the reported out degrees and in degrees. Recall that the complexity of the post processing algorithm (i.e., Algorithm 2) is $O(n \log n + n\tilde{d}^-_{\max})$. Therefore, the algorithm takes $O(n \log n + n\tilde{d}^-_{\max})$ on the central server side at each timestamp. Each user reports his/her estimated out degree and in degree to the central server, which takes $O(n)$ communication complexity at each timestamp.

Discussion. Notice that at each timestamp, on each user side, the collection algorithm proposed in Section 4 (i.e., Algorithm 3) takes $O(\tilde{d}^+_{\max} + \tilde{d}^-_{\max})$ time, while the optimized algorithm (i.e., Algorithm 5) takes $O(\tilde{d}^+_{\max} + \tilde{d}^-_{\max})/\theta)$ time. Actually, the efficiency optimization proposed in Section 5.2 can make a lot of degrees should not need to be updated ar a timestamp, in this case, the optimized algorithm only takes $O(1)$ time on the user side.



(a) Time (BO) (b) M_1^+ and M_1^- (BO) (c) M_2^+ (BO) (d) M_2^- (BO)

(e) Time (WV) (f) M_1^+ and M_1^- (WV) (g) M_2^+ (WV) (h) M_2^- (WV)

Figure 5: Impact of privacy budget ϵ

6 EXPERIMENTS

In this section, 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.

6.1 Datasets, Algorithms, and Parameters

Table 1 shows the information of all datasets. All datasets are from SNAP [3]. For each timestamp, we randomly delete 2% edges and add 2% edges, then obtain a streaming graph. Here, $|V|$ denotes the number of vertices; $|E|$ denotes the number of average number of edges per timestamp; T denotes the number of timestamps; $\deg^+_{\max}$ denote the average of the maximum out degree of all the vertices per timestamp; $\deg^-_{\max}$ denote the average of the maximum in degree of all the vertices per timestamp.

We test the basic **collection algorithm** proposed in Section 4 and **optimized algorithm** proposed in Section 5, respectively. In each experiment, we run the algorithm 10 times and reported the average result. Privacy budget ϵ, the window length w, the graph size $|V_{\mathrm{G}}|$, and the number of timestamps T_{G} are 1.0, 5, $|V|$, and T, respectively. The projection thresholds $\tilde{d}^+_{\max}$ and $\tilde{d}^-_{\max}$ are $\lceil \deg^+_{\max} \rceil$ and $\lceil \deg^-_{\max} \rceil$, For optimized algorithm, the step size θ is 15 and the privacy budget allocation ratio $\epsilon_1 : \epsilon_2 : \epsilon_3 : \epsilon_4 = 1 : 1 : 1 : 7$.

6.2 Utility Metric

To evaluate the utility of proposed methods, we use two metrics at each timestamp. The first one is ℓ_1-norm of the difference between the exact and released degree list, which is also used in [37]. Moreover, we count the number of vertices corresponding to different elements in exact and released degree list. Here, if the absolute value of the difference of two elements is lower than $\log |V_G|$, we consider them to be equal. We consider the out degree list and in degree list, separately. We report the average result of all the timestamps. We use M_1^+, M_2^+, M_1^-, and M_2^- to denote these metrics: $T_{\mathrm{G}} \sum_{t=1}^{T_{\mathrm{G}}} \sum_{i=1}^{|V_{\mathrm{G}}|} \mathbb{I}(d^+_{t,i} - \overline{d}^+_{t,i} > \log |V_{\mathrm{G}}|)$, $\frac{1}{T_{\mathrm{G}}} \sum_{t=1}^{T_{\mathrm{G}}} \|d^+_t - \overline{d}^+_t\|_1$, $T_{\mathrm{G}} \sum_{t=1}^{T_{\mathrm{G}}} \sum_{i=1}^{|V_{\mathrm{G}}|} \mathbb{I}(d^-_{t,i} - \overline{d}^-_{t,i} > \log |V_{\mathrm{G}}|)$, and $\frac{1}{T_{\mathrm{G}}} \sum_{t=1}^{T_{\mathrm{G}}} \|d^-_t - \overline{d}^-_t\|_1$. Here, d^+_t denotes the exact out degree list, $\overline{d}^+_t$ denotes the released out degree list, d^-_t denotes the exact in degree list, and $\overline{d}^-_t$ denotes the released in degree list.

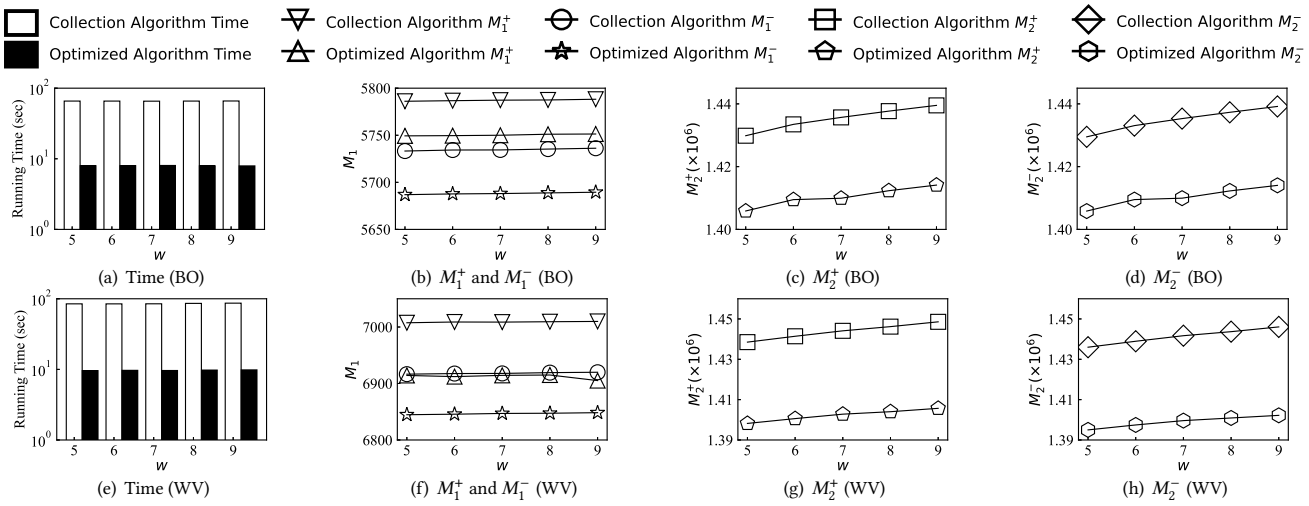
Table 1: Statistics of the datasets

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	T	$\deg^+_{\max}$	$\deg^-_{\max}$
Bitcoin OTC (BO)	5.9**K**	35.6**K**	100	762.98	504.88
Wiki-Vote (WV)	7.1**K**	103.7**K**	100	892.97	434
Math Overflow (MO)	24.8**K**	228**K**	100	1847.96	912.73
As-Caida (AC)	26.5**K**	106.8**K**	100	2627.97	2531.87
Cit-HepPh (CH)	34.5**K**	421.5**K**	100	411.05	828.88
P2P-Gnutella (PG)	36.7**K**	88.3**K**	100	53.78	52.45

6.3 Experimental Results

Exp-1: Impact of privacy budget ϵ. As shown in Fig 5, when the privacy budget ϵ increases, the running time of the collection algorithm and that of the optimized algorithm both remains stable. The reason is that the cardinality of output space $|\mathcal{R}|$ in the exponential mechanism does not change with the variation of ϵ. Moreover, the post processing phase does not relate to the privacy budget ϵ. Besides, it shows that our efficiency optimization technique proposed in Section 5.2 scales well with ϵ. We also find that the optimized algorithm is about an order of magnitude faster than the collection algorithm. There are two main reasons for this. First, the optimized algorithm uses the step size when implementing the exponential mechanism, then the cardinality of the output space $|\mathcal{R}|$ is smaller than the collection algorithm. Second, the optimized algorithm uses the efficiency optimization technique, then at some timestamps, the optimized algorithm does not update estimated out degrees or in degrees for some users. In terms of utility, an increase in ϵ leads to a decrease in M_1^+, M_1^-, M_2^+, and M_2^- of two algorithms, which improves the utility of the algorithm. An exception is M_1^+ of the optimized algorithm on WV, because at small ϵ values, the output of the exponential mechanism may have a relatively large error. If one run of the exponential mechanism result in a smaller error and the out degree is not updated, it may lead to a smaller overall error compared to updating it at each timestamp.

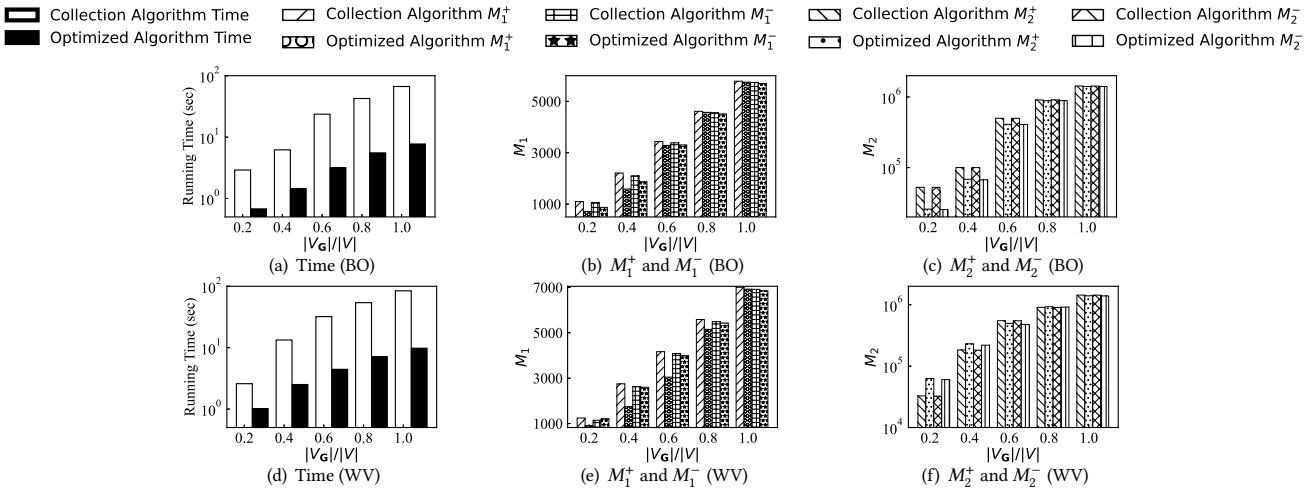
Exp-2: Impact of window length w. As shown in Fig. 6, when the window length w increases, the running times of both the collection



(a) Time (BO) (b) M_1^+ and M_1^- (BO) (c) M_2^+ (BO) (d) M_2^- (BO)

(e) Time (WV) (f) M_1^+ and M_1^- (WV) (g) M_2^+ (WV) (h) M_2^- (WV)

Figure 6: Impact of window length w



(a) Time (BO) (b) M_1^+ and M_1^- (BO) (c) M_2^+ and M_2^- (BO)

(d) Time (WV) (e) M_1^+ and M_1^- (WV) (f) M_2^+ and M_2^- (WV)

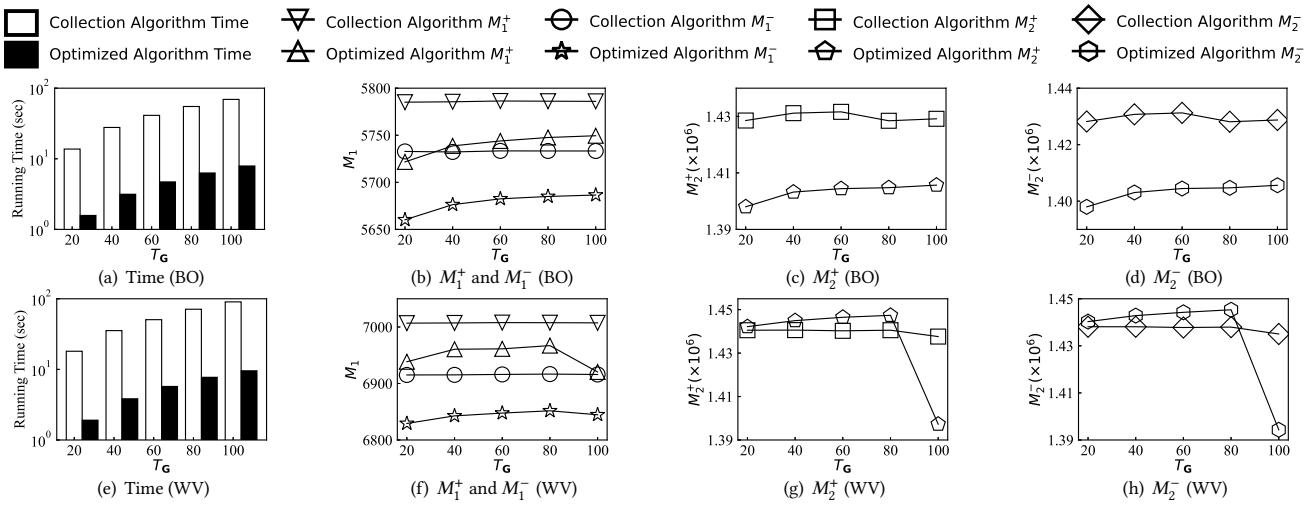
Figure 7: Impact of graph size $|V_G|$

algorithm and the optimized algorithm remain stable. The reason is same to **Exp-1**, that the window length w has no influence on the cardinality of the output space $|\mathcal{R}|$ and the post processing phase does not relate to the window length w. Meanwhile, the proposed efficiency optimization technique scales well with w. For utility, when w increases, M_1^+ and M_1^- of two algorithms are slow growth, while M_2^+ and M_2^- increase at a faster rate. This means when w increases, the utility decreases. The reason is Keeping the total privacy budget constant, a larger window length w results in a smaller privacy budget that can be consumed at each timestamp, which will lead to greater error.

Exp-3: Impact of graph size $|V_G|$. As shown in Fig. 7, when $|V_G|$ increases, Both the collection and the optimized algorithms exhibit an increase in running time. It stands to reason that as the input sizes of the two algorithms grow, their running times will inevitably increase. For utility, although M_1^+, M_1^-, M_2^+, M_2^- of both the collection algorithm and optimized algorithm increase with larger $|V_G|$, this does not imply poor scalability. Examining the calculation of M_1^+, M_1^-, M_2^+, and M_2^- reveals that when $|V_G|$ increases, the number of terms being summed is also increasing. The growth rates of M_1^+, M_1^-, M_2^+, and M_2^- demonstrate that our algorithm scales well with the graph size $|V_G|$.

Exp-4: Impact of the number of timestamps T_G. As shown in Fig. 8, when the number of timestamps T_G increases, the running times of both the collection algorithm and the optimized algorithm increases. The reason behind that is obvious. The more timestamps, the longer the computation time required. For dataset BO, M_1^+, M_1^-, M_2^+, and M_2^- of two algorithms both do not fluctuate much. For dataset WV, when $T_G < 100$, the M_2^+ and M_2^- of the optimized algorithm are larger than these of the collection algorithm, respectively. However, when T_G reaches 100, the situation is reversed. The reason is that the optimized algorithm uses less privacy budget in exponential mechanism than the collection algorithm, since the efficiency optimization technique also consumes the privacy budget. When the T_G becomes large, there will be more vertices maintaining the same degree at different times, thus the updating by using exponential mechanism is not necessary.

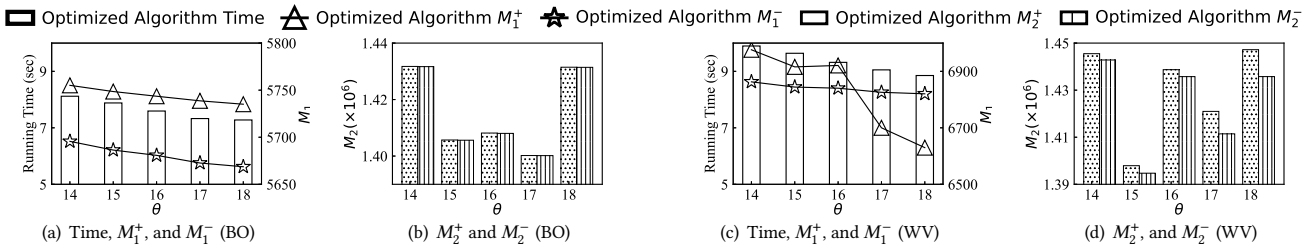
Exp-5: Impact of step size θ. As shown in Fig. 9, when θ increases, the running time of the optimized algorithm decreases. The reason behind that is larger θ will lead to a smaller output space $\mathcal{R}$, then the



(a) Time (BO) (b) M_1^+ and M_1^- (BO) (c) M_2^+ (BO) (d) M_2^- (BO)

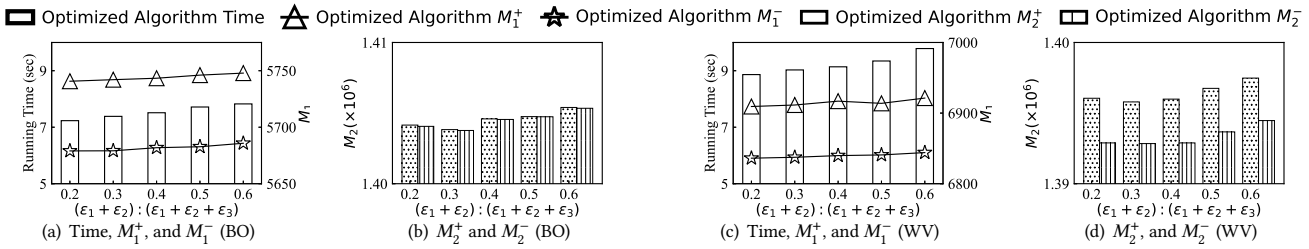
(e) Time (WV) (f) M_1^+ and M_1^- (WV) (g) M_2^+ (WV) (h) M_2^- (WV)

Figure 8: Impact of the number of timestamps T_G



(a) Time, M_1^+, and M_1^- (BO) (b) M_2^+ and M_2^- (BO) (c) Time, M_1^+, and M_1^- (WV) (d) M_2^+, and M_2^- (WV)

Figure 9: Impact of step size θ



(a) Time, M_1^+, and M_1^- (BO) (b) M_2^+ and M_2^- (BO) (c) Time, M_1^+, and M_1^- (WV) (d) M_2^+, and M_2^- (WV)

Figure 10: Impact of privacy budget allocation $(\epsilon_1+\epsilon_2):(\epsilon_1+\epsilon_2+\epsilon_3)$

number of elements to be traversed in the exponential mechanism has decreased. For utility, M_1^+ and M_1^- of the optimized algorithm both decreases. Besides, M_2^+ and M_2^- of the optimized algorithm first decrease, then increase. This is because as θ increases from a small value, it reduces the cardinality of the output space $|\mathcal{R}|$ in the exponential mechanism, preventing the output distribution from becoming overly smooth and thereby improving utility. However, once θ exceeds a threshold, the output space $|\mathcal{R}|$ becomes too small, leading to inaccurate estimation of the degree.

Exp-6: Impact of privacy budget allocation $(\epsilon_1+\epsilon_2):\epsilon_3$ We consider the privacy budget allocation in the efficiency optimization technique, i.e., $(\epsilon_1+\epsilon_2):(\epsilon_1+\epsilon_2+\epsilon_3)$. As shown in Fig. 10, when the privacy budget allocation ratio $(\epsilon_1+\epsilon_2):(\epsilon_1+\epsilon_2+\epsilon_3)$ increases, the running time of the optimized algorithm increases, since the privacy budget used in thresholds (i.e. $(\epsilon_1+\epsilon_2)$) increases, more degrees will be updated through exponential mechanism at each timestamp if $(\epsilon_1+\epsilon_2+\epsilon_3)$ is fixed. For utility, when $(\epsilon_1+\epsilon_2):(\epsilon_1+\epsilon_2+\epsilon_3)$ increases, the M_1^+, M_1^-, M_2^+, and M_2^- of the optimized algorithm increases. The reason behind this is also that more degrees will be updated through exponential mechanism, which introduces greater error than not updating.

7 CONCLUSION

In this paper, we study the locally differentially private degree list collection problem in streaming directed graphs. To solve this problem, we proposed two algorithms, collection algorithm and optimized algorithm. The collection algorithm based on the composition theorem. The optimized algorithm overcomes the oversmoothing problem of the exponential mechanism and uses a novel efficiency optimization technique to utilize the correlation of different timestamp snapshots. We also provide a novel post processing algorithm based on a discrete mathematics theorem. Experimental results show that the proposed algorithms have high efficiency and utility.

REFERENCES

[1] [n.d.]. https://flink.apache.org/
[2] [n.d.]. https://spark.apache.org/
[3] [n.d.]. https://snap.stanford.edu/data/index.html/
[4] Ergute Bao, Yin Yang, Xiaokui Xiao, and Bolin Ding. 2021. CGM: An Enhanced Mechanism for Streaming Data Collection with Local Differential Privacy. *Proc. VLDB Endow.* 14, 11 (2021), 2258–2270.
[5] Annabell Berger. 2014. A note on the characterization of digraphic sequences. *Discret. Math.* 314 (2014), 38–41.
[6] Felipe T. Brito, Victor A. E. de Farias, Cheryl J. Flynn, Subhabrata Majumdar, Javam C. Machado, and Divesh Srivastava. 2023. Global and Local Differentially Private Release of Count-Weighted Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 154:1–154:25.
[7] Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. 2020. Vertex Ordering Problems in Directed Graph Streams. In *SODA 2020*. SIAM, 1786–1802.
[8] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2011. Private and Continual Release of Statistics. *ACM Trans. Inf. Syst. Secur.* 14, 3 (2011), 26:1–26:24.
[9] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing Graph Degree Distribution with Node Differential Privacy. In *SIGMOD 2016*. ACM, 123–138.
[10] Wei Dong, Zijun Chen, Qiyao Luo, Elaine Shi, and Ke Yi. 2024. Continual Observation of Joins under Differential Privacy. *Proc. ACM Manag. Data* 2, 3 (2024), 128.
[11] Wei Dong, Qiyao Luo, Giulia Fanti, Elaine Shi, and Ke Yi. 2024. Almost Instance-optimal Clipping for Summation Problems in the Shuffle Model of Differential Privacy. In *CCS 2024*. ACM, 1939–1953.
[12] Wei Dong, Qiyao Luo, and Ke Yi. 2023. Continual Observation under User-level Differential Privacy. In *SP 2023*. IEEE, 2190–2207.
[13] Leilei Du, Peng Cheng, Lei Chen, Heng Tao Shen, Xuemin Lin, and Wei Xi. 2025. Infinite Stream Estimation under Personalized w-Event Privacy. *Proc. VLDB Endow.* 18, 6 (2025), 1905–1918.
[14] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT 2006 (Lecture Notes in Computer Science)*, Vol. 4004. Springer, 486–503.
[15] 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.
[16] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *STOC 2010*. ACM, 715–724.
[17] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. 2009. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC 2009*. ACM, 381–390.
[18] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.
[19] Juanru Fang and Ke Yi. 2025. Counting Subgraphs under Shuffle Differential Privacy. In *CCS 2025*. ACM, 2369–2383.
[20] Chengzhu He, Zhendong Wang, Zhaorui Meng, Junfeng Yao, Shihui Guo, and Huamin Wang. 2025. Automated Task Scheduling for Cloth and Deformable Body Simulations in Heterogeneous Computing Environments. In *SIGGRAPH 2025*. ACM, Article 24, 11 pages.
[21] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Wei Ni. 2025. Robust Privacy-Preserving Triangle Counting under Edge Local Differential Privacy. *Proc. ACM Manag. Data* 3, 3 (2025), 211:1–211:26.
[22] Monika Henzinger and Jalaj Upadhyay. 2025. Improved Differentially Private Continual Observation Using Group Algebra. In *SODA 2025*. SIAM, 2951–2970.
[23] Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. 2023. Almost Tight Error Bounds on Differentially Private Continual Counting. In *SODA 2023*. SIAM, 5003–5039.
[24] Monika Henzinger, Jalaj Upadhyay, and Sarvagya Upadhyay. 2024. A Unifying Framework for Differentially Private Sums under Continual Observation. In *SODA 2024*. SIAM, 995–1018.
[25] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2021. Locally Differentially Private Analysis of Graph Statistics. In *USENIX Security 2021*. USENIX Association, 983–1000.
[26] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-Efficient Triangle Counting under Local Differential Privacy. In *USENIX Security 2022*. USENIX Association, 537–554.
[27] 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.
[28] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially Private Event Sequences over Infinite Streams. *Proc. VLDB Endow.* 7, 12 (2014), 1155–1166.
[29] Xiaochen Li, Tianyu Li, Yitian Cheng, Chen Gong, Kui Ren, Zhan Qin, and Tianhao Wang. 2025. SPAS: Continuous Release of Data Streams under w-Event Differential Privacy. *Proc. ACM Manag. Data* 3, 1 (2025), 78a:1–78a:27.
[30] Xuankun Liao, Qing Liu, Xin Huang, and Jianliang Xu. 2024. Truss-based Community Search over Streaming Directed Graphs. *Proc. VLDB Endow.* 17, 8 (2024), 1816–1829.
[31] 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.
[32] 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.
[33] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proc. VLDB Endow.* 10, 6 (2017), 637–648.
[34] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks V. S. Lakshmanan, and Xiaolin Han. 2022. A Convex-Programming Approach for Efficient Directed Densest Subgraph Discovery. In *SIGMOD 2022*. ACM, 845–859.
[35] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD 2009*. ACM, 19–30.
[36] 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.
[37] Sofya Raskhodnikova and Adam D. Smith. 2016. Lipschitz Extensions for Node-Private Graph Statistics and the Generalized Exponential Mechanism. In *FOCS 2016*. IEEE Computer Society, 495–504.
[38] Sofya Raskhodnikova and Teresa Anna Steiner. 2025. Fully Dynamic Algorithms for Graph Databases with Edge Differential Privacy. *Proc. ACM Manag. Data* 3, 2 (2025), 99:1–99:28.
[39] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *SIGMOD 2022*. ACM, 1064–1077.
[40] 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.
[41] Anxin Tian, Alexander Zhou, Yue Wang, Xun Jian, Lei Chen, Yan Zhou, and Chen Zhang. 2025. Distributed Truss Decomposition over Large Directed Graphs. *VLDB J.* 34, 5 (2025), 59.
[42] Songlei Wang, Yifeng Zheng, Xiaohua Jia, and Haibo Hu. 2025. PrivAGM: Secure Construction of Differentially Private Directed Attributed Graph Models on Decentralized Social Graphs. *Proc. VLDB Endow.* 18, 11 (2025), 4682–4694.
[43] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous Release of Data Streams under both Centralized and Local Differential Privacy. In *CCS 2021*. ACM, 1237–1253.
[44] Haisong Xia and Zhongzhi Zhang. 2024. Fast Computation of Kemeny's Constant for Directed Graphs. In *KDD 2024*. ACM, 3472–3483.
[45] Xiaokui Xiao. 2024. Sharing Information with Differential Privacy: A Database Perspective. *Proc. VLDB Endow.* 17, 12 (2024), 4555.
[46] Dongdong Xie, Pinghui Wang, Quanqing Xu, Chuanhui Yang, and Rundong Li. 2025. Efficient and Accurate Differentially Private Cardinality Continual Releases. *Proc. ACM Manag. Data* 3, 3 (2025), 151:1–151:27.
[47] Wen Xu, Pengpeng Qiao, Shang Liu, Zhirun Zheng, Yang Cao, and Zhetao Li. 2025. Continuous Publication of Weighted Graphs with Local Differential Privacy. *Proc. VLDB Endow.* 18, 11 (2025), 4214–4226.

APPENDIX

A PROOF OF THEOREM 4.1

Proof. For any two w-neighboring neighbor lists $\mathbf{u}$ and $\mathbf{u}'$, for any $i \in [n]$, let $D_i(\mathbf{u})$ and $D_i(\mathbf{u}')$ be the elements of the algorithm $\mathcal{A}_i$ depends on. For any measurable subset O,

$$\begin{aligned}
&\Pr[(\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_w)(\mathbf{u}) \in O] \\
=&\prod_{i=1}^{w} \Pr[\mathcal{A}_i(D_1(\mathbf{u}), \cdots, D_i(\mathbf{u}), \cdots, D_w(\mathbf{u}))) \in O] \\
\leq&\prod_{i=1}^{w} e^{\epsilon_i} \Pr[\mathcal{A}_i(D_1(\mathbf{u}'), \cdots, D_i(\mathbf{u}'), \cdots, D_w(\mathbf{u}'))) \in O] \\
\leq&e^{\sum_{i=1}^{w} \epsilon_i} \Pr[\mathcal{A}(\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_w)(\mathbf{u}') \in O].
\end{aligned}$$

□

B PROOF OF PROPOSITION 4.2

Proof.

$$\begin{aligned}
GS_{f^+} &= \max_{r \in \mathbb{Z}_n} \max_{\boldsymbol{\lambda}, \boldsymbol{\mu} \in \{0,1\}^n \setminus \{\mathbf{1}\}} |f^+(\boldsymbol{\lambda}, r) - f^+(\boldsymbol{\mu}, r)| \\
&= \max_{r \in \mathbb{Z}_n} \max_{\boldsymbol{\lambda}, \boldsymbol{\mu} \in \{0,1\}^n \setminus \{\mathbf{1}\}} ||\sum_{j=1}^{n} \lambda_j - r| - |\sum_{j=1}^{n} \mu_j - r|| \\
&= \max_{r \in \mathbb{Z}_n} \max_{\boldsymbol{\lambda}. \boldsymbol{\mu} \in \{0,1\}^n \setminus \{\mathbf{1}\}} \max\{|\sum_{j=1}^{n} \lambda_j - r|, |\sum_{j=1}^{n} \mu_j - r|\} \\
&= \max_{r \in \mathbb{Z}_n} \max\{\max_{\boldsymbol{\lambda} \in \{0,1\}^n \setminus \{\mathbf{1}\}} |\sum_{j=1}^{n} \lambda_j - r|, \max_{\boldsymbol{\mu} \in \{0,1\}^n \setminus \{\mathbf{1}\}} |\sum_{j=1}^{n} \mu_j - r|\} \\
&= \max_{r \in \mathbb{Z}_n} \max\{r, n-1-r\} \\
&= n-1.
\end{aligned}$$

Similarly, it is easy to calculate that $GS_{f^-} = n-1$. □

C PROOF OF THEOREM 4.6

Proof. Let $\mathcal{A}_t$ be the sub-algorithm running at timestamp t, $\forall t \in [T]$. By Theorem 3.7, and Theorem 3.8, $\mathcal{A}_t$ satisfies 1-event local differential privacy for each user with privacy budget ϵ/w. Then, for any window $\mathcal{I} \in \{[s, s+w-1] : s \in [1, T-w+1]\}$, by Theorem 4.1, the parallel composition $\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t$ satisfies w-event local differential privacy for each user with privacy budget ϵ. Next, by Definition 3.2, for any pair of w-neighboring neighbor lists $\mathbf{u}_i$ and $\mathbf{u}_i'$, and any measurable set $O \subseteq \text{Range}(\bigotimes_{t \in [T]} \mathcal{A}_t)$,

$$\begin{aligned}
&\Pr[\bigotimes_{t \in [T]} \mathcal{A}_t(\mathbf{u}_i) \in O] \\
=&\Pr[\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t(\mathbf{u}_i) \in O_{\mathcal{I}}, \bigotimes_{t \in [T] \setminus \mathcal{I}} \mathcal{A}_t(\mathbf{u}_i) \in O_{\overline{\mathcal{I}}}] \\
=&\Pr[\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t(\mathbf{u}_i) \in O_{\mathcal{I}}] \cdot \Pr[\bigotimes_{t \in [T] \setminus \mathcal{I}} \mathcal{A}_t(\mathbf{u}_i) \in O_{\overline{\mathcal{I}}}] \\
\leq&e^{\epsilon} \Pr[\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t(\mathbf{u}_i') \in O_{\mathcal{I}}] \cdot \Pr[\bigotimes_{t \in [T] \setminus \mathcal{I}} \mathcal{A}_t(\mathbf{u}_i') \in O_{\overline{\mathcal{I}}}] \\
=&e^{\epsilon} \Pr[\bigotimes_{t \in [T]} \mathcal{A}_t(\mathbf{u}_i') \in O].
\end{aligned}$$

Here, $\mathcal{I} = \{t \in [T] : \mathbf{u}_{i,t} \neq \mathbf{u}_{i,t}'\}, O_{\mathcal{I}} = \{\bigotimes_{t \in \mathcal{I}} o_t : (o_1, o_2, \cdots, o_T) \in O\}$, and $O_{\overline{\mathcal{I}}} = \{\bigotimes_{t \in [T] \setminus \mathcal{I}} o_t : (o_1, o_2, \cdots, o_T) \in O\}$. Therefore, Algorithm 4.1 satisfies w-event local differential privacy for each user with privacy budget ϵ.

□

D PROOF OF THEOREM 4

Proof. Similar to Algorithm 3, Algorithm 4 can be divided into there phases: (1) neighbor list projection; (2) degree collection; (3) post processing. According to Theorem 3.8, we only need to prove that the first and second phases satisfies w-event local differential privacy for each user with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4)$.

The neighbor list projection and degree collection phases $\mathcal{A}$ can be divided into two sub-algorithms $\mathcal{A}^+$ and $\mathcal{A}^-$. We prove that both $\mathcal{A}^+$ and $\mathcal{A}^-$ satisfy w-event local differential privacy for each user with privacy budget ϵ. Similar to Theorem 4.6, the claim can be equivalent to that for any window $\mathcal{I} \in \{[s, s+w-1] : s \in [1, T-w+1]\}$, the parallel composition $\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t^+$ and $\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t^-$ satisfy w-event local differential privacy for each user.

For each user v_i, for any window $\mathcal{I} \in \{[s, s+w-1] : s \in [1, T-w+1]\}$, the algorithm $\bigotimes_{t \in \mathcal{I}} \mathcal{A}_t^+$ consists of two stages: (1) determine whether the result should be updated; (2) update the result. We use $\mathcal{A}_{\mathcal{I},1}^+$ and $\mathcal{A}_{\mathcal{I},2}^+$ to denote the two stages, respectively.

We prove that $\mathcal{A}_{\mathcal{I},1}^+$ satisfies local differential privacy with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3)$ for user v_i, and $\mathcal{A}_{\mathcal{I},2}^+$ satisfies local differential privacy with privacy budget ϵ_4 for user v_i. Notice that $\text{Range}(\mathcal{A}_{\mathcal{I},1}^+) \subseteq \{\top, \bot\}^w$. Here, $\top$ represents that Algorithm 4 updates the released result and $\bot$ represents that Algorithm 4 does not update.

Consider two neighboring neighbor lists $\mathbf{u}_j$ and $\mathbf{u}_j'$. For any $i \in \mathcal{I}, \forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^w \times \mathbb{R}^w$, let

$$\begin{aligned}
g_i(\mathbf{u}_j, x_i, y_i) &= \Pr[(\chi_{t,i}^+ \leq \eta_t^{l,+} + x_i) \bigvee (\chi_{t,i}^+ \geq \eta_t^+ + y_i)], \\
h_i(\mathbf{u}_j, x_i, y_i) &= \Pr[\eta_t^+ + x_i < \chi_{t,i}^+ < \eta_t^{l,+} + y_i].
\end{aligned}$$

$\rho_{t,i}^{u,+}$ has a probability density function $f_{\rho_{t,i}^{u,+}}(x) = \frac{\epsilon_1}{2w\tilde{d}_{t,\max}^+} \exp(-\epsilon_1|x|/w\tilde{d}_{t,\max}^+)$ and $\rho_{t,i}^{l,+}$ has a probability density function $f_{\rho_{t,i}^{l,+}}(x) = \frac{\epsilon_2}{2w\tilde{d}_{t,\max}^+} \exp(-\epsilon_2|x|/w\tilde{d}_{t,\max}^+)$

For any output $o \in \text{Range}(\mathcal{A}_{\mathcal{I},1}^+)$, let

$$\mathbb{I}_\top = \{k : o_k = \top\}, \mathbb{I}_\bot = \{k : o_k = \bot\}.$$

Then,

$$\begin{aligned}
&\Pr[\mathcal{A}_{\mathcal{I},1}^+(\mathbf{u}_j) = o] \\
=&\int_{\mathbb{R}^w \times \mathbb{R}^w} \prod_{i \in \mathcal{I}} f_{\rho_{t,i}^{u,+}}(x_i) \prod_{i \in \mathcal{I}} f_{\rho_{t,i}^{l,+}}(y_i) \prod_{i \in \mathbb{I}_\top} g_i(\mathbf{u}_j, x_i, y_i) \prod_{i \in \mathbb{I}_\bot} h_i(\mathbf{u}_j, x_i, y_i) \mathbf{dxdy},
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[\mathcal{A}_{\mathcal{I},1}^+(\mathbf{u}_j') = o] \\
=&\int_{\mathbb{R}^w \times \mathbb{R}^w} \prod_{i \in \mathcal{I}} f_{\rho_{t,i}^{u,+}}(x_i) \prod_{i \in \mathcal{I}} f_{\rho_{t,i}^{l,+}}(y_i) \prod_{i \in \mathbb{I}_\top} g_i(\mathbf{u}_j', x_i, y_i) \prod_{i \in \mathbb{I}_\bot} h_i(\mathbf{u}_j', x_i, y_i) \mathbf{dxdy}.
\end{aligned}$$

Notice that,

$$\Pr[\mathcal{A}^{+}_{\mathcal{I},1}(\mathbf{u}_j) = o]$$
$$= \int_{\mathbb{R}^w \times \mathbb{R}^w} \prod_{i \in \mathcal{I}} f_{\rho^{u,+}_{t,i}}(x_i + \tilde{d}^{+}_{t,\max}) \prod_{i \in \mathcal{I}} f_{\rho^{l,+}_{t,i}}(y_i - \tilde{d}^{+}_{t,\max}) \prod_{i \in \mathbb{I}_\top} g_i(\mathbf{u}_j, x_i + \tilde{d}^{+}_{t,\max}, y_i - \tilde{d}^{+}_{t,\max}) \prod_{i \in \mathbb{I}_\perp} h_i(\mathbf{u}_j, x_i + \tilde{d}^{+}_{t,\max}, y_i - \tilde{d}^{+}_{t,\max}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y},$$

and

$$\forall i \in \mathcal{I}, f_{\rho^{u,+}_{t,i}}(x_i + \tilde{d}^{+}_{t,\max}) \le e^{\epsilon_1/w} f_{\rho_{t,i}}(x_i),$$
$$\forall i \in \mathcal{I}, f_{\rho^{l,+}_{t,i}}(y_i - \tilde{d}^{+}_{t,\max}) \le e^{\epsilon_2/w} f_{\rho_{t,i}}(y),$$
$$\forall i \in \mathbb{I}_\top, g_i(\mathbf{u}_j, x_i + \tilde{d}^{+}_{t,\max}, y_i - \tilde{d}^{+}_{t,\max}) \le e^{\epsilon_3/w} g_i(\mathbf{u}'_j, x_i, y_i),$$
$$\forall i \in \mathbb{I}_\perp, h_i(\mathbf{u}_j, x_i + \tilde{d}^{+}_{t,max}, y_i - \tilde{d}^{+}_{t,\max}) \le h_i(\mathbf{u}'_j, x_i, y_i).$$

Then,

$$\Pr[\mathcal{A}^{+}_{\mathcal{I},1}(\mathbf{u}_j) = o] \le e^{\epsilon_1+\epsilon_2+\epsilon_3} \Pr[\mathcal{A}^{+}_{\mathcal{I},1}(\mathbf{u}'_j) = o].$$

$\mathcal{A}^{+}_{\mathcal{I},1}$ satisfies local differential privacy with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3)$ for user v_i. According to Theorem 3.4, $\mathcal{A}^{+}_{\mathcal{I},2}$ satisfies local differential privacy with privacy budget ϵ_4. Then, according to the Theorem 4.1, $\bigotimes_{t \in \mathcal{I}} \mathcal{A}^{+}_t$ satisfies w-event local differential privacy for user v_i with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4)$. Similarly, we can prove that $\bigotimes_{t \in \mathcal{I}} \mathcal{A}^{-}_t$ satisfies w-event local differential privacy for user v_i with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4)$.

Thus, both $\mathcal{A}^{+}$ and $\mathcal{A}^{-}$ satisfy w-event local differential privacy for each user with privacy budget $(\epsilon_1+\epsilon_2+\epsilon_3+\epsilon_4)$. Finally, according to Theorem 3.7, the algorithm $\mathcal{A}$ satisfies w-event local differential privacy for each user with privacy budget $(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4)$. □