Continuous Publication of Weighted Graphs with Local Differential Privacy

Wen Xu
Jinan University
wenxu018@gmail.com

Pengpeng Qiao*
Institute of Science Tokyo
peng2qiao@gmail.com

Shang Liu
China University of Mining and Technology
shang@cumt.edu.cn

Zhirun Zheng
Ajou University
zhengzhirun2019@gmail.com

Yang Cao
Institute of Science Tokyo
cao@c.titech.ac.jp

Zhetao Li
Jinan University
liztchina@gmail.com

ABSTRACT

Although a large amount of valuable knowledge can be obtained from the weighted graph snapshots modeled over time, it may cause privacy issues. Local differential privacy (LDP) provides a strong solution for private graph data publishing in decentralized networks. However, most existing LDP studies over graphs are only applicable to static unweighted graphs. This paper investigates the problem of continuous publication of weighted graph snapshots and proposes a graph publication framework, WGT-LDP, under w-event edge weight LDP, which can protect the privacy of edges and weights over any w consecutive time steps. WGT-LDP consists of four key components: population division-based sampling that overcomes the problem of over-segmentation of the privacy budget, data range estimation that mitigates noise on edge weights, aggregate information collection that obtains important information about the graph structure and edge weights, and graph snapshot generation that reconstructs weighted graph snapshot at each time step. We provide theoretical guarantees on privacy and utility, and perform extensive experiments on three real-world and two synthetic datasets, using four commonly used metrics. Our experiments show that WGT-LDP produces high-quality synthetic weighted graphs and significantly outperforms baseline methods.

PVLDB Reference Format:
Wen Xu, Pengpeng Qiao, Shang Liu, Zhirun Zheng, Yang Cao, and Zhetao Li. Continuous Publication of Weighted Graphs with Local Differential Privacy. PVLDB, 18(11): 4214-4226, 2025.
doi:10.14778/3749646.3749688

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/xuwen22/WGT-LDP.

1 INTRODUCTION

Due to graphs providing an excellent ability to represent relational data, they have been widely used in many real-world complex systems, such as social media and computer networks [24]. In many cases, the interaction behaviors between entities in these systems often change dynamically over time, which can be modeled as continuous graph snapshots. Moreover, these snapshots may carry more valuable information beyond connecting different entities (called weights), e.g., interaction frequency, which can be collected and analyzed for further complex tasks. The following are two real examples of publishing and analyzing weighted graph snapshots.

Example 1. ***Social Network Analytics***. *On professional social platforms such as LinkedIn, the edges and weights in weighted graph snapshots can reflect the actual dynamic relevance and connection strength between users, thereby increasing the efficiency of job search and recruitment [15].*

Example 2. ***Epidemiological Network Analytics***. *Analyzing continuous weighted graph data is also great popularity in epidemiological network systems, e.g., respiratory infectious diseases. By studying the dynamic interactions (edges) and interaction strengths (weights) between individuals in continuous snapshots, their risk of contracting respiratory diseases can be estimated [10, 12].*

However, directly publishing these graph data for analysis without protection may pose privacy risks. Local differential privacy (LDP) [7, 20] is a strong privacy-preserving technique that can be used to collect sensitive data from users in a decentralized network without a complete central database. In the LDP setting, to ensure privacy, each user must send the data injected with random noise to the curator. Existing work on LDP-based graph publishing mainly focuses on static unweighted graphs [33, 42], while very little focus on weighted graph data in the temporal dimension. Despite this, providing privacy for dynamic networks with abundant valuable information may be more common in the near future [38].

More precisely, our scenario is concerned with the continuous publication of weighted graph snapshots, where both the graph structure and edge weights may be private information. We focus on methods that provide w-event-level privacy because they are applicable to infinite time steps while providing strong privacy protection [21]. In particular, w-event-level privacy aims to protect any event sequence occurring within any window of w time steps, which means that individuals can guarantee the privacy of their data over w consecutive snapshots. Figure 1 shows an example on a decentralized network where the curator regularly collects perturbation data from all users to continuously synthesize weighted

*Corresponding Author

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749688

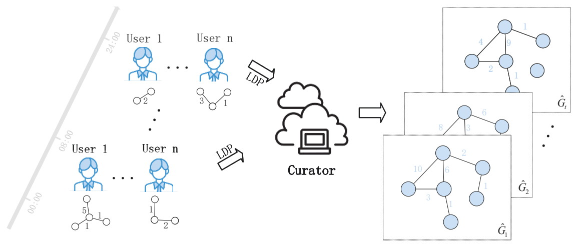


Figure 1: The example of continuously synthesizing weighted graph snapshots in decentralized networks.

graphs. To the best of our knowledge, previous works on w-event-level privacy have focused on tabular streaming data [26, 28, 34, 41], and have not explored continuous weighted graph publication under LDP. To this end, we have three technical challenges:

- **Over-segmentation of privacy budget.** To achieve w-event-level privacy, most existing methods on tabular data need to allocate privacy budgets for any sliding window of size w, so that the sum of the budgets within this window does not exceed the total privacy budget ε. However, the publishing mechanism of graph data usually contains multiple components that consume privacy budgets, which will cause further segmentation of the privacy budget and thus destroy the utility of the data.
- **High noise magnitude of edge weights.** In order not to violate the differential privacy protocol, the magnitude of the noise added to the edge weight should be proportional to its upper bound. In the study of central DP [44], the upper bound can be lowered based on the total error of all weights in the original graph. This is difficult under LDP since the curator cannot access the original data.
- **Bias of noisy adjacency matrix.** After adding noise to every bit of the adjacency matrix with weights, zero edges are converted to edges with real weights, making the results no longer useful. It is challenging to mitigate the negative impact of noise on the performance of synthetic weighted graph snapshots.

In this paper, we propose WGT-LDP, a new framework for continuous weighted graph publishing under w-event edge weight LDP. We first design a population division-based sampling scheme, which considers data changes and samples disjoint nodes in the window, so that each node has sufficient privacy budget to perturb its data. To effectively reduce the noise of edge weights, we propose a data range estimation mechanism where the curator collects the noisy local maximum weights of sampling nodes and runs a post-processing technique to estimate the local data range of each node. Since node degrees and adjacency lists carry knowledge about graph structure and edge weights, respectively, we collect these two aggregates of information from sampling nodes. By using the coarse-grained noisy node degrees as evidence and the fine-grained noisy adjacency lists as reference, we can recover the characteristics of the original snapshot and preserve structural sparsity. Our main contributions are summarized as follows:

New Perspective. We first explore the problem of weighted graph publication with LDP in the temporal dimension, which aims to generate continuous synthetic weighted graph snapshots while satisfying privacy protection requirements.

Simple yet Effective Solution. We propose a continuous weighted graphs publication framework WGT-LDP under w-event edge weight LDP. WGT-LDP first adopts a population division strategy to adaptively sample nodes with large data changes, and then uses a data range estimation method to reduce the impact of perturbation noise on edge weights. Finally, WGT-LDP collects noisy degrees as evidence and noisy adjacency lists as reference, and adopts different methods to reconstruct the current snapshot to ensure weighted graph utility.

Extensive Experimental Evaluations. We theoretically prove that the proposed WGT-LDP satisfies w-event edge weight LDP and analyze the utility. Extensive experiments on several datasets and metrics demonstrate the effectiveness of WGT-LDP.

2 PRELIMINARIES

2.1 Local Differential Privacy (LDP)

Differential privacy (DP) [8], a gold standard for data privacy, originally assumed a centralized model where a trusted curator holds the exact data of all users. This may lead to some privacy and security issues. For instance, a data curator may sell data for personal gain or suffer an attack that leads to data leakage [36]. Local differential privacy (LDP) [7] effectively solves the above problems by not assuming a trusted third party. In LDP, each user can use the DP mechanism to perturb personal sensitive data locally before the data is sent to the curator. Formally, LDP is defined as follows:

Definition 2.1 (LDP). A randomized algorithm $\mathcal{M}$ provides ε-LDP, where $\varepsilon > 0$, if and only if for any pair of input values $x, x' \in D$ and any possible output x^*,

$$\Pr\left[\mathcal{M}(x) = x^*\right] \le e^{\varepsilon} \Pr\left[\mathcal{M}(x') = x^*\right], \tag{1}$$

where ε is called the privacy budget. The smaller ε can provide stronger privacy guarantees.

Geometric Mechanism. The Geometric mechanism (GM) [13] is mainly used for queries with integer results. It satisfies the LDP requirements by adding random geometric noise to the query function f on the input x. The magnitude of noise is proportional to the global sensitivity, defined as,

$$\Delta f = \underset{x, x' \in D}{\text{maximize}} \|f(x) - f(x')\|_1. \tag{2}$$

For numerical data, the Geometric mechanism $\mathcal{M}$ is as follows:

$$\mathcal{M}(x) = f(x) + Geo(e^{-\varepsilon/\Delta f}), \tag{3}$$

where $Geo(\lambda)$ denotes a random integer noise drawn from two-sided geometric distribution $\Pr[Geo(\lambda) = z] = \frac{1-\lambda}{1+\lambda}\lambda^{|z|}$, and it has a mean of 0.

Square Wave Mechanism. The Square Wave mechanism (SW) [25] attempts to increase the probability that the noisy response value carries useful information about the true value, which extends the idea of generalized randomized response. That is, the probability that users report values closer to x is greater than values farther away from x for given input x.

In particular, we assume that the input domain is $[0, 1]$ and the output domain is $[-b, 1 + b]$, where $b = \frac{\varepsilon e^{\varepsilon} - e^{\varepsilon} + 1}{2e^{\varepsilon}(e^{\varepsilon} - 1 - \varepsilon)}$. For values in the range $[l, h]$, they are first mapped into $[0, 1]$ (by transforming value x to $\frac{x-l}{h-l}$), and then the estimated values are transformed back. The perturbation function of SW is defined as:

$$\forall_{\tilde{x}\in[-b,1+b]}, \Pr[\mathrm{SW}(x) = \tilde{x}] = \begin{cases} p, & \text{if } |x - \tilde{x}| \le b, \\ q, & \text{otherwise}, \end{cases} \quad (4)$$

where $p = \frac{e^{\varepsilon}}{2be^{\varepsilon}+1}$ and $q = \frac{1}{2be^{\varepsilon}+1}$.

The LDP mechanisms satisfy the properties of sequential/parallel composition and post-processing [8, 29], which provide privacy guarantees for building complex LDP algorithms.

2.2 LDP for Weighted Graphs

In the context of weighted graphs, both the edges and edge weights may be private information. Depending on the privacy requirement, we introduce a formal definition of LDP for weighted graphs.

Specifically, let $V = \{v_1, v_2, \cdots, v_n\}$ be the set of all nodes (i.e., users) in a undirected weighted graph. The adjacency list of a node v_i can be denoted as an n-dimensional bit vector $(a(v_i, v_1), \cdots, a(v_i, v_n))$, where $a(v_i, v_j)$ is the edge weight between nodes v_i and v_j. Note that if edge (v_i, v_j) exists in the weighted graph, then $a(v_i, v_j) \ge 1$; otherwise $a(v_i, v_j) = 0$. The LDP for weighted graphs is formally defined as follows:

Definition 2.2 (edge weight LDP). A randomized algorithm $\mathcal{M}$ provides ε-edge weight LDP, where $\varepsilon > 0$, if and only if for any two adjacency lists $\mathbf{a}$ and $\mathbf{a}'$ that only differ in one bit (called neighbors), and for any possible output $s \in Range(\mathcal{M})$,

$$\Pr[\mathcal{M}(\mathbf{a}) = s] \le e^{\varepsilon} \Pr[\mathcal{M}(\mathbf{a}') = s]. \quad (5)$$

Edge weight LDP is the same as edge LDP [33] except that the former (resp. latter) considers weighted graph (resp. unweighted graph). Under this definition, two adjacency lists differ by exactly one bit, indicating that they can differ on an edge and can also differ on the weight of an edge. Therefore, if a mechanism satisfies edge weight LDP, the impact of any edges or any edge weights on the final output is bounded, which guarantees the privacy of edges and edge weights.

2.3 Problem Statement

In our paper, we are interested in weighted graph data with temporal attribute. In particular, consider a decentralized network system where a data curator periodically collects information from n users. Based on the collected data at each time step, the curator sequentially publishes weighted graph snapshots at these time steps. Let $G_t = (V, E_t, a_t)$ be a undirected weighted graph snapshot at time steps t. $V = \{v_1, v_2, \cdots, v_n\}$ is the set of all n users. $E_t \subseteq V \times V$ is the set of edges, where an edge $(v_i, v_j) \in E_t$ denotes a relationship between users v_i and v_j at time step t. The weight function $a_t : E_t \to \mathbb{R}$ maps edge (v_i, v_j) at time step t to a real weight. For each node v_i in G_t, the degree $(d_t)_i$ denotes the number of edges connected to the node and $(A_t)_i = (a_t(v_i, v_1), \cdots, a_t(v_i, v_n))$ denotes its adjacency list, where $a_t(v_i, v_j) \ge 1$ if $(v_i, v_j) \in E_t$, and otherwise $a_t(v_i, v_j) = 0$. The adjacency lists of all nodes at time step t form the adjacency matrix of graph G_t, formalized as $A_t = \{(A_t)_1, (A_t)_2, \cdots, (A_t)_n\}$. Since the data curator is untrustworthy, our goal is to design a LDP solution that helps the curator to collect user information and then construct the synthetic weighted graph snapshot $\hat{G}_t$ at each time step t. The resulting synthetic snapshot sequence $\hat{\mathcal{G}} = \langle \hat{G}_1, \hat{G}_2, \cdots \rangle$ is representative, i.e., it can support any downstream graph statistical analysis task while preserving individual privacy.

To balance utility loss and privacy loss over infinite sequences, we follow w-event-level privacy model [21] and extend its definition to the local setting for weighted graphs. Before that, we first introduce some notions. We call two adjacency lists $(A_t)_i$, $(A_t)'_i$ at time step t are neighboring if they only differ in one bit. Let a time-series $\mathcal{X} = \langle (A_1)_i, (A_2)_i, \cdots \rangle$, the definition of w-neighboring time-series is described as follows:

Definition 2.3 (w-neighboring time-series). For a positive integer w, two time-series $\mathcal{X}, \mathcal{X}'$ of length T are w-neighboring, if for each $\mathcal{X}[t], \mathcal{X}'[t]$ such that $t \in [T]$ and $\mathcal{X}[t] \ne \mathcal{X}'[t]$, it holds that $\mathcal{X}[t], \mathcal{X}'[t]$ are neighboring; and for each $\mathcal{X}[t_1], \mathcal{X}[t_2], \mathcal{X}'[t_1], \mathcal{X}'[t_2]$ with $t_1 < t_2$, $\mathcal{X}[t_1] \ne \mathcal{X}'[t_1]$ and $\mathcal{X}[t_2] \ne \mathcal{X}'[t_2]$, it holds that $t_2 - t_1 + 1 \le w$.

That is to say, if $\mathcal{X}, \mathcal{X}'$ are w-neighboring time-series, then their elements are the same or neighboring, and all their neighboring elements can fit in a window of at most w time steps. Hence, we define w-event edge weight LDP below.

Definition 2.4 (w-event edge weight LDP). Let $\mathcal{M}$ be a randomized algorithm that takes as input time-series $\mathcal{X}$ consisting of a single user's consecutive adjacency lists. We say that $\mathcal{M}$ provides w-event ε-edge weight LDP if and only if for any w-neighboring time-series $\mathcal{X}, \mathcal{X}'$, and for any possible output $S \in Range(\mathcal{M})$,

$$\Pr[\mathcal{M}(\mathcal{X}) = S] \le e^{\varepsilon} \Pr[\mathcal{M}(\mathcal{X}') = S]. \quad (6)$$

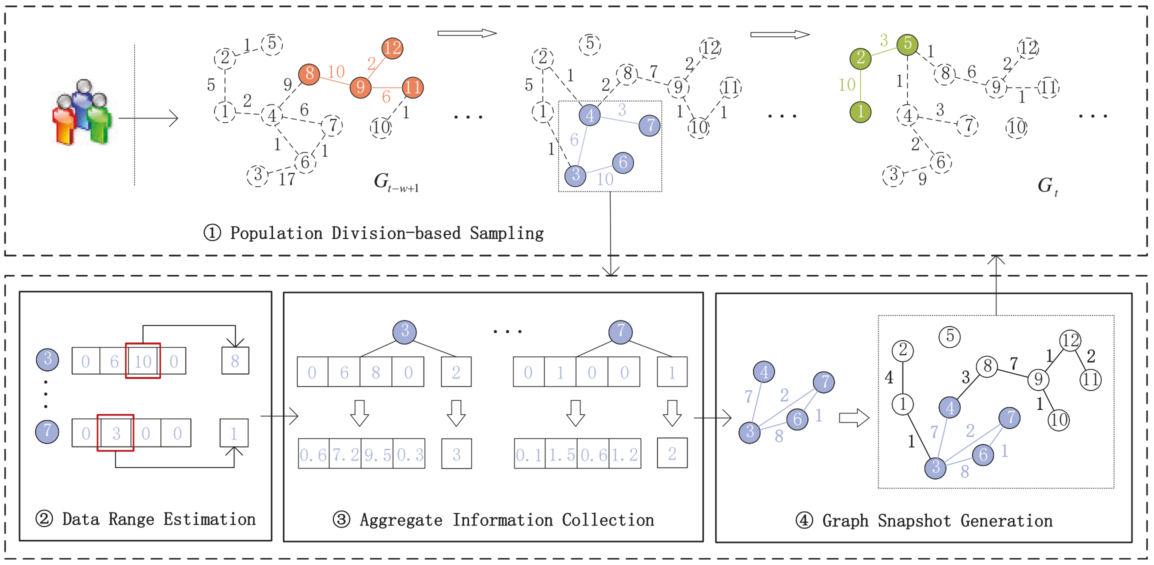
Discussion. Definition 2.4 aims to guarantee each user ε-edge weight LDP for any sliding window including w consecutive time steps, where ε can be regarded as the total available privacy budget in this sliding window. In other words, w-event edge weight LDP ensures that the impact of the user's events in any w consecutive graph snapshots on the query result is limited. Therefore, an attacker cannot infer individual's information at any w consecutive time steps by observing the final sequence of synthetic graph snapshots. Note that when $w = 1$, the privacy protection level degenerates to event-level.

Example. We assume that w is 2. If the time-series of node v_1 with length 4 is $\mathcal{X} = \langle (0, 1, 3), (0, 0, 6), (0, 2, 3), (0, 2, 1) \rangle$, then one of its w-neighboring time-series can be $\mathcal{X}' = \langle (0, 1, 3), (0, 1, 6), (0, 2, 5), (0, 2, 1) \rangle$. This is because elements $(0, 1, 6)$ and $(0, 2, 5)$ in $\mathcal{X}'$ (i.e., adjacency lists) are neighboring to elements $(0, 0, 6)$ and $(0, 2, 3)$ in $\mathcal{X}$, respectively, and these neighboring elements are within the time window of size 2. The w-event edge weight LDP makes the probability that an attacker can distinguish between $\mathcal{X}$ and $\mathcal{X}'$ by observing the output of the randomized algorithm $\mathcal{M}$ controlled by the privacy budget ε.

3 OUR APPROACH

3.1 Overview

To publish weighted graphs at each time step with w-event edge weight LDP, we propose a new framework WGT-LDP. Figure 2 shows the workflow of WGT-LDP, which consists of four phases. Algorithm 1 gives the complete synthesis process of WGT-LDP.



① Population Division-based Sampling
G_{t-w+1}
G_t
② Data Range Estimation
③ Aggregate Information Collection
④ Graph Snapshot Generation

Figure 2: The overview of our WGT-LDP framework.

Algorithm 1 Overall protocol of WGT-LDP framework

```
Input: Original adjacency matrices $\langle A_1, A_2, \cdots, A_t, \cdots \rangle$, Node set $V$, total privacy budget $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4$, window size $w$, threshold $\delta$, transformation probabilities $M$, bucket size $r$
Output: Synthetic snapshots $\hat{\mathcal{G}} = \langle \hat{G}_1, \hat{G}_2, \cdots, \hat{G}_t, \cdots \rangle$ for publication
1:  Initialize $\hat{G}_0$ with adjacency matrix $A_0 \in 0^{n \times n}$;
2:  for each time step $t$ do
3:      Obtain remaining node set $V_t = V \setminus \sum_{k=t-w+1}^{t-1} V_s^k$;
4:      $V_s^t = Sampling(\hat{G}_{t-1}, V_t, A_t, \delta, w, \varepsilon_1)$;    ▷ Algorithm 2
5:      for each node $v_i^t \in V_s^t$ do
6:          Construct sub-adjacency list $(A_s^t)_i$ involving $V_s^t$;
7:          $h_i^t = \max((A_s^t)_i)$;
8:          $(d_s^t)_i = |\{j \mid (A_s^t)_{ij} \neq 0\}|$;
9:      end for
10:     $\hat{h}_t = Estimation((h_1^t, \cdots h_{s_t}^t), M, r, \varepsilon_2)$;    ▷ Algorithm 3
11:     $\tilde{A}_s^t, \hat{d}_s^t = Collection(d_s^t, A_s^t, \hat{h}_t, \varepsilon_3, \varepsilon_4)$;    ▷ Algorithm 4
12:     $\hat{G}_t = Generation(\hat{G}_{t-1}, \tilde{A}_s^t, \hat{d}_s^t, \hat{h}_t, M, r)$;    ▷ Algorithm 5
13: end for
14: return $\hat{\mathcal{G}}$
```

- **Population Division-based Sampling.** This component aims to sample nodes with large data changes at each time step. We sample nodes at the current time step that have never been sampled in the previous $w-1$ time steps. To this end, the curator calculates the number of nodes whose noisy change error of the adjacency list are greater than a threshold δ among the current remaining nodes and adaptively adjusts the node sampling ratio. Then, the curator focuses on the information in the subgraph composed of all sampling nodes.
- **Data Range Estimation.** This component aims to estimate the local range of each sampling node's data in the subgraph to reduce the impact of perturbation noise on edge weights. In particular, each sampling node sends the perturbed local maximum weight to the curator, who adopts Expectation Maximization with Smoothing (EMS) algorithm [25] and Bayesian estimation to determine their noisy maximum weights.
- **Aggregate Information Collection.** This component aims to collect the privatized degrees and adjacency lists of the sampling nodes in the subgraph. For each sampling node, it independently perturbs its degree and adjacency list based on the estimated local data range, and then sends them to curator. The curator performs post-processing to the collected perturbation information.
- **Graph Snapshot Generation.** This component aims to generate a weighted graph at the current time step. For the edges and weights between sampling nodes, the curator first adopts the noisy degree sequence as a characterization of the graph structure and the values in the noisy adjacency lists as a reference to rebuild the edges. Then the EMS algorithm and Bayesian estimation are applied again to rebuild the weights on these edges. For the edges and weights of non-sampling nodes, the curator uses the data of the previous time step to rebuild them.

3.2 Population Division-based Sampling

For continuous publication of infinite sequences with w-event-level privacy, a general approach is to allocate privacy budgets for any sliding window containing w time steps. However, in our scenario, multiple components for publishing graph snapshots have to consume privacy budget, so the privacy budget of the sampling time steps also needs to be redistributed. This may lead to high noise level that destroy the utility of the synthetic graph. LDP-IDS [34] proposed to assign users to multiple groups, each of which uses the entire privacy budget in a window. This method provides w-event-level privacy under parallel composition. Unfortunately, random user sampling scheme ignores the differences in user data changes, which will introduce bias to the generation of graph data.

Algorithm 2 Population Division-based Sampling

```
Input: Published weighted graph $\hat{G}_{t-1}$, remaining node set $V_t$, adjacency list $(A_t)_i \in R^n$ of node $v_i \in V$, threshold $\delta$, window size $w$, privacy budget $\varepsilon_1$
Output: Sampling node set $V_s^t$
    // Collect change errors
1:  for each node $v_i \in V_t$ do
2:      $E_i^t = |\{j \mid a_t(v_j, v_i) \neq \hat{a}_{t-1}(v_j, v_i), v_j \in V_t\}|$;
3:      $\tilde{E}_i^t = GM(E_i^t, \frac{\varepsilon_1}{w}, \Delta f_{err})$;
4:  end for
    // Sampling strategy
5:  $m_t = |\{i \mid \tilde{E}_i^t \geq \delta, v_i \in V_t\}|$;
6:  $P_t = 1 - \exp(-\frac{n}{w \cdot m_t})$;
7:  $s_t = m_t \cdot P_t$;
8:  Sort $V_t$ in descending order of $\tilde{E}_i^t$;
9:  Select top $s_t$ in $V_t$ as sampling node set $V_s^t$;
10: return $V_s^t$
```

Algorithm 3 Data Range Estimation

```
Input: Maximum $h_i^t$ in sub-adjacency list of node $v_i^t \in V_s^t$, transformation probabilities $M$, bucket size $r$, privacy budget $\varepsilon_2$
Output: Noisy maximum values $\hat{h}_t = \{\hat{h}_1^t, \cdots, \hat{h}_{s_t}^t\}$
    // Collect values
1:  for each node $v_i^t \in V_s^t$ do
2:      $x_i^t = h_i^t / B$;
3:      $\tilde{x}_i^t = SW(x_i^t, \varepsilon_2)$;
4:  end for
    // Post-processing
5:  $\tilde{\mathbf{x}} = \{\tilde{x}_1^t, \cdots, \tilde{x}_{s_t}^t\}$;
6:  $\tilde{\mathbf{z}} = EMS(\tilde{\mathbf{x}}, M, r)$;
7:  for $i \in \{1, 2, \cdots, s_t\}$ do
8:      Calculate $\hat{x}_i^t$ based on $\tilde{\mathbf{z}}$ by Eq. 10 and 11;
9:      $\hat{h}_i^t = B \cdot \hat{x}_i^t$;
10: end for
11: return $\hat{h}_t$
```

To strike the trade-off between high noise and information bias, we design a new population division-based sampling method that considers data changes. Specifically, a portion of the entire privacy budget ε is divided evenly to each time step in the time window for estimating the data dynamics. After that, the curator decides which nodes need to be sampled at each time step based on the number of remaining nodes and the noisy data change error of these nodes. Subsequently, this group of sampling nodes will use another portion of the privacy budget to perturb the local data between them. Since any node is only allowed to be sampled once in each sliding window, further allocation of the perturbation budget in the window is avoided, and the privacy budget cost of each node does not exceed ε.

Let V_t be the remaining node set at time step t, which is calculated by removing the already sampled nodes in the previous $w-1$ time steps from the node set V. For each node v_i in V_t with adjacency list $(A_t)_i$, we first define its data change error E_i^t as the number of unequal bits involving V_t between adjacency lists $(A_t)_i$ and $(\hat{A}_{t-1})_i$, where $(\hat{A}_{t-1})_i$ is the noisy adjacency list of v_i in the previous synthetic graph $\hat{G}_{t-1}$. That is,

$$E_i^t = \left|\{j \mid a_t(v_j, v_i) \neq \hat{a}_{t-1}(v_j, v_i), v_j \in V_t\}\right|. \tag{7}$$

When there is no previous synthetic graph, i.e., at the first time step, we directly adopt $E_i^t = \left|\{j \mid a_t(v_j, v_i) \neq 0, v_j \in V_t\}\right|$ as the change error. In other words, we assume that the edges and weights in the previous synthetic graph are all 0. Then, We let each node locally add Geometric noise to E_i^t with privacy budget $\frac{\varepsilon_1}{w}$ to ensure w-event edge weight LDP, and send the privatized error $\tilde{E}_i^t$ to the curator. Note that the global sensitivity Δf_{err} of this query is 1 because changing one bit in the adjacency list result in the change of E_i^t by at most 1.

After receiving the noisy errors from all remaining nodes, the curator calculates the number of nodes whose noisy errors are greater than a threshold δ:

$$m_t = \left|\{i \mid \tilde{E}_i^t \geq \delta, v_i \in V_t\}\right|. \tag{8}$$

A simple solution is to sample all these m_t nodes. However, too large m_t will result in very few nodes that can be sampled at the next $w-1$ time steps since any node only participates once in a window. In other words, the data between the large number of non-sampling nodes in the subsequent graph need to be approximated by the previous time step, which may cause excessive approximation deviation for graphs with significant dynamic changes. Therefore, the nodes that can be sampled at each time step in the sliding window are limited and should be carefully allocated. To this end, the curator adaptively adjusts the node sampling ratio based on m_t. We choose $\frac{n}{w}$ as the adjustment threshold since it implies the theoretical average number of sampling nodes at each time step in the window. In particular, when m_t is less than $\frac{n}{w}$, the m_t nodes should be sampled as much as possible to effectively capture data changes. When m_t is greater than $\frac{n}{w}$, the number of samples should be reduced to ensure that there are still enough nodes available for sampling in future time steps, which can maintain the utility of the synthetic graph. Based on the above analysis, we set the sampling ratio as follows:

$$P_t = 1 - \exp(-\frac{n}{w \cdot m_t}). \tag{9}$$

The final number of sampling nodes obtained at the current time step is calculated as $s_t = m_t \cdot P_t$, and the corresponding set of sampling node, denoted as $V_s^t = \{v_i^t \mid i \in [s_t]\}$, consists of the nodes with the top s_t data change errors among the m_t nodes. This process is described in Algorithm 2.

3.3 Data Range Estimation

Given the sampling nodes at time step t, the curator focuses on the data of the subgraph G_s^t composed of these nodes, which has a public data-independent upper bound B, i.e., the maximum possible weight in G_s^t. To satisfy edge weight LDP, the noise added to the local edge weight of each node needs to be proportional to B, which may lead to large edge weight errors. If users are only required to perturb the edge weights based on their true local data domain, differential privacy cannot be satisfied, because the upper bound of

Algorithm 4 Aggregate Information Collection

```
Input: Degree $(d_s^t)_i$, sub-adjacency list $(A_s^t)_i$, noisy maximum
    values $\hat{h}_t = \{\hat{h}_1^t, \cdots, \hat{h}_{s_t}^t\}$, privacy budget $\varepsilon_3, \varepsilon_4$
Output: Noisy degree sequence $\hat{d}_s^t$ and sub-adjacency matrix $\tilde{A}_s^t$
    // Collect degrees and sub-adjacency lists
1:  for each node $v_i^t \in V_s^t$ do
2:      $(\tilde{d}_s^t)_i = GM((d_s^t)_i, \varepsilon_3, \Delta f_d)$;
3:      $a_t(v_i^t, v_j^t) = Truncate(a_t(v_i^t, v_j^t), \hat{h}_i^t)$;
4:      $b_t(v_i^t, v_j^t) = a_t(v_i^t, v_j^t) / \hat{h}_i^t$;
5:      $\tilde{b}_t(v_i^t, v_j^t) = SW(b_t(v_i^t, v_j^t), \varepsilon_4)$;
6:  end for
    // Degrees adjustment
7:  $\hat{d}_s^t = NormSub(\tilde{d}_s^t)$;
8:  if $\sum_{i=1}^{s_t} (\hat{d}_s^t)_i$ is odd then
9:      randomly select a node degree $(\hat{d}_s^t)_i$;
10:     flip a coin to decide whether to add 1 or subtract 1;
11: end if
12: return $\tilde{A}_s^t, \hat{d}_s^t$
```

the domain itself is private information. In the paper, we propose a method to estimate the local data domain of each sampling node.

In particular, let the sub-adjacency matrix of G_s^t be A_s^t, where row vector $(A_s^t)_i$ is the sub-adjacency list of the sampling node v_i^t. The local data range of v_i^t is $[0, h_i^t]$, where $h_i^t \in [0, B]$ is the maximum value in $(A_s^t)_i$. Given the privacy budget ε_2 and parameter $b = \frac{\varepsilon_2 e^{\varepsilon_2} - e^{\varepsilon_2} + 1}{2e^{\varepsilon_2}(e^{\varepsilon_2} - 1 - \varepsilon_2)}$, each sampling node transforms h_i^t to $x_i^t = \frac{h_i^t}{B} \in [0, 1]$, and uses the SW to report a value close to x_i^t with probability $p = \frac{e^{\varepsilon_2}}{2be^{\varepsilon_2}+1}$, which carries useful information about x_i^t, and then sends the perturbed value $\tilde{x}_i^t$ to the curator.

In order to improve the estimation accuracy and reduce the bias, we perform a post-processing step on the perturbed values. First, we use Expectation Maximization with smoothing (EMS) algorithm [25] to infer the probability distribution $\tilde{\mathbf{z}}$ of the original value x_i^t of all sampling nodes. To run EMS, the perturbed value $\tilde{x}_i^t$ reported by the user need to be discretized into r buckets in output domain $[-b, 1+b]$. In this phase, we set the number of buckets to $r = \lfloor B \rfloor$.

After that, the probability distribution $\tilde{\mathbf{z}}$ is used as a prior, such that the curator can apply Bayes' theorem to calculate the corresponding posterior probability distribution, i.e., for each $i \in [s_t]$,

$$P(x_i^t \in b_k \mid \tilde{x}_i^t \in b_j) = \frac{M_{j,k} \cdot \tilde{\mathbf{z}}_k}{\sum_{k=1}^{r} M_{j,k} \cdot \tilde{\mathbf{z}}_k}. \tag{10}$$

For each sampling node v_i^t, the curator selects the upper bound of the bucket with the maximum posterior probability as its noisy value, i.e.,

$$\hat{x}_i^t = \sup(\arg\max_{b_k} P(x_i^t \in b_k \mid \tilde{x}_i^t \in b_j)), \tag{11}$$

and calculates the corresponding noisy local maximum weight as $\hat{h}_i^t = B \cdot \hat{x}_i^t$. Algorithm 3 shows the estimation procedure.

3.4 Aggregate Information Collection

Based on the estimated data range, the sampling nodes locally perturb important information about the topology of the subgraph G_s^t at time step t, including node degrees and sub-adjacency lists. In particular, the node degrees reflect the subgraph density and have low sensitivity, which can better denoise the subgraph topology. The sub-adjacency lists contain fine-grained information of weighted connections between sampling nodes, which can be used as a reference for edge generation and reconstruct edge weights.

Let $(d_s^t)_i$ be the degree of sampling node v_i^t in the subgraph G_s^t. As shown in Algorithm 4, given the privacy budget ε_3, each sampling node v_i^t use Geometric mechanism to inject unbiased noise into its degree. The sensitivity of degree Δf_d is 1 because adjusting one bit from the adjacency list of a node changes its degree by at most 1. For sub-adjacency list $(A_s^t)_i$, the sampling node v_i^t maps each bit $a_t(v_i^t, v_j^t)$ to $b_t(v_i^t, v_j^t) \in [0, 1]$ according to the estimated data range $[0, \hat{h}_i^t]$, and then uses the SW to randomly perturb these bit with the privacy budget ε_4. Note that when the value of a bit is larger than $\hat{h}_i^t$, it will first be truncated, i.e., $a_t(v_i^t, v_j^t) = \min(a_t(v_i^t, v_j^t), \hat{h}_i^t)$. Since the SW exploits the ordinal property of edge weights, a report that is different from but close to the true weight also carries useful information about the weight, which is exactly what we expect.

After adding Geometric noise, some perturbed degree values may appear negative and their sum may be odd, which makes the subsequent graph generation infeasible. Thus, we first adopt NormSub [40] to post-process the negative value problem. Given perturbed degree sequence $\tilde{d}_s^t$, our goal is to find an optimal integer $\alpha^* = \arg\min_{\alpha} |\sum_{i \in [s_t]} \max((\tilde{d}_s^t)_i + \alpha, 0) - \sum_{i \in [s_t]} (\tilde{d}_s^t)_i|$. After obtaining α^*, each node degree $(\tilde{d}_s^t)_i$ is updated to $(\hat{d}_s^t)_i = \max((\tilde{d}_s^t)_i + \alpha^*, 0)$, which satisfies non-negative constraint. To solve the odd sum problem, we randomly select a node degree in $\hat{d}_s^t$ and flip a coin to decide whether to add 1 or subtract 1 to this node degree.

3.5 Graph Snapshot Generation

Now, the curator generates the synthetic subgraph $\hat{G}_s^t$ at time step t based on the noisy degree sequence $\hat{d}_s^t$ and the noisy sub-adjacency matrix $\tilde{A}_s^t$ (consisting of the noisy sub-adjacency lists from all sampling nodes). We first use $\hat{d}_s^t$ to characterize the graph structure and then use $\tilde{A}_s^t$ as a reference to reconstruct the edges in $\hat{G}_s^t$. The intuition of this method is that the bits with high noisy values in the sub-adjacency matrix are more likely to correspond to non-zero edges in the original subgraph.

As shown in Algorithm 5, the curator picks a node v_i^t with the minimum degree and considers other nodes with non-zero degree in $\hat{d}_s^t$ as candidate nodes that may be connected to v_i^t. For potential edges between v_i^t and each candidate node v_j^t, the curator refers to the bits $\tilde{b}_t(v_i^t, v_j^t)$ and $\tilde{b}_t(v_j^t, v_i^t)$ in $\tilde{A}_s^t$ related to their existence and adds a potential edge with the largest bit product to $\hat{G}_s^t$. That is, the index of the node v_j^t connected to v_i^t is determined to be

$$j = \arg\max_{j} \tilde{b}_t(v_i^t, v_j^t) \cdot \tilde{b}_t(v_j^t, v_i^t). \tag{12}$$

Then, the degrees corresponding to nodes v_i^t and v_j^t in $\hat{d}_s^t$ are both subtracted by 1. The above process will be repeated until $\hat{d}_s^t$ is reduced to 0 or no additional edges can be added.

To reconstruct the weights on the edges, the curator considers the bit $\tilde{b}_t(v_i^t, v_j^t)$ corresponding to each edge (v_i^t, v_j^t) in $\hat{G}_s^t$, and

Algorithm 5 Graph Snapshot Generation

```
Input: Published weighted graph $\hat{G}_{t-1}$, noisy degree sequence $\hat{d}_s^t$, noisy sub-adjacency matrix $\tilde{A}_s^t = (\tilde{b}_t(v_i^t, v_j^t))$, noisy maximum values $\hat{h}_t = \{\hat{h}_1^t, \cdots, \hat{h}_{s_t}^t\}$, transformation probabilities $M$, bucket size $r$
Output: Synthetic weighted graph $\hat{G}_t$ for publication
1:  Initialize $\hat{G}_t = (\hat{G}_s^t, \hat{G}_t \backslash \hat{G}_s^t)$ as a null graph;
2:  while $\hat{d}_s^t \neq 0$ or $|\hat{d}_s^t > 0| \neq 1$ do
3:      Choose a node $v_i^t$ with $(\hat{d}_s^t)_i$ is minimal positive entry in $\hat{d}_s^t$;
4:      for each $v_j^t \neq v_i^t$ with $(\hat{d}_s^t)_j$ is the positive entry do
5:          Pick $v_j^t$ with $j = \arg\max_j \tilde{b}_t(v_i^t, v_j^t) \cdot \tilde{b}_t(v_j^t, v_i^t)$;
6:      end for
7:      Add edge $(v_i^t, v_j^t)$ to $\hat{G}_s^t$;
8:      $(\hat{d}_s^t)_i = (\hat{d}_s^t)_i - 1$, $(\hat{d}_s^t)_j = (\hat{d}_s^t)_j - 1$;
9:  end while
10: Construct sequence $\tilde{\mathbf{b}}$ by bits corresponding to all edges in $\hat{G}_s^t$;
11: $\tilde{\mathbf{z}} = EMS(\tilde{\mathbf{b}}, M, r)$;
12: for each $\tilde{b}_t(v_i^t, v_j^t) \in \tilde{\mathbf{b}}$ do
13:     Calculate $\hat{b}_t(v_i^t, v_j^t)$ based on $\tilde{\mathbf{z}}$ by Eq. 10 and 11;
14:     $\hat{a}_t(v_i^t, v_j^t) = \hat{h}_i^t \cdot \hat{b}_t(v_i^t, v_j^t)$;
15:     Add weight $\hat{a}_t(v_i^t, v_j^t)$ to $\hat{G}_s^t$;
16: end for
    // Generate edges and weights in $\hat{G}_t \backslash \hat{G}_s^t$
17: Approximate with the corresponding values $\hat{G}_{t-1}$;
18: return $\hat{G}_t$
```

then post-processes $\tilde{b}_t(v_i^t, v_j^t)$ by applying the EMS algorithm and Bayesian estimation described in Section 3.3. In this phase, we set the number of buckets to $r = \lfloor \max(\hat{h}_t) \rfloor$. After obtaining the post-processed noisy bit $\hat{b}_t(v_i^t, v_j^t)$, the noisy weight on the edge (v_i^t, v_j^t) is calculated as $\hat{a}_t(v_i^t, v_j^t) = \hat{h}_i^t \cdot \hat{b}_t(v_i^t, v_j^t)$.

For the edges and weights of non-sampling nodes, they approximate the values in the last generated graph. At this point, the synthetic weighted graph snapshot $\hat{G}_t$ at time step t is generated.

4 THEORETICAL ANALYSIS

4.1 Privacy Analysis

Recall that WGT-LDP allocates the total budget into three components: ε_1 to population division-based sampling, ε_2 to data range estimation, ε_3 and ε_4 to aggregate information collection ($\varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 = \varepsilon$). The graph snapshot generation component only post-processes the perturbed data and does not need to allocate any budget. Then, WGT-LDP has the following privacy guarantee.

Theorem 4.1. *WGT-LDP satisfies w-event ε-edge weight LDP, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4$.*

Proof. See the technical report [45] for proof details. □

4.2 Utility Analysis

We use the expected $\ell 1$-distance between $\hat{A}_t$ and A_t at any time step t, i.e., $E[\|\hat{A}_t - A_t\|_{1,1}]$, to evaluate the utility of WGT-LDP. Due to space constraints, please refer to the technical report of this paper [45] for a detailed analysis of determining utility metric.

Unlike the error estimation of tabular stream data with w-event LDP [34], the expected $\ell 1$-distance of WGT-LDP is complicated. To simplify the utility analysis, we assume that the edge weights between non-sampling nodes are approximated by 0. Besides, for the subgraph G_s^t composed of sampling nodes, its maximum edge weight h_{max}^t is at most $\hat{h}_{max}^t = \max\{\hat{h}_1^t, \cdots, \hat{h}_{s_t}^t\}$, where $\hat{h}_i^t$ represents the noisy maximum value in sub-adjacency list of sampling node v_i^t. Then WGT-LDP has the following utility guarantee.

Theorem 4.2. *Assume that the edge weights between non-sampling nodes are approximated by 0. Then, for any time step $t \in \mathbb{Z}_{\geq 0}$, $\hat{h}_{max}^t \in \mathbb{R}_{\geq 0}$, and subgraph $G_s^t \in G_t$ composed of sampling nodes such that the maximum edge weight h_{max}^t of G_s^t is at most $\hat{h}_{max}^t$, we have*

$$E[\|\hat{A}_t - A_t\|_{1,1}] \leq \frac{\hat{h}_{max}^t(b+1)}{2}\|A_t\|_{0,0} + \|A_t\|_{1,1}, \quad (13)$$

where $b = \frac{\varepsilon_4 e^{\varepsilon_4} - e^{\varepsilon_4} + 1}{2e^{\varepsilon_4}(e^{\varepsilon_4} - 1 - \varepsilon_4)}$ and $\|A_t\|_{0,0}$ denotes the number of non-zero elements in A_t.

Proof. Refer to the technical report [45] for proof details. □

Discussion. Theorem 4.2 shows that under w-event edge weight LDP, the expected $\ell 1$ error of WGT-LDP is positively correlated with the original graph G_t itself and its maximum edge weight. In fact, G_t is usually sparse, and only a small number of edges have large weights. In addition, the privacy budget ε_4 associated with the utility is independent of w, i.e., it does not need to be allocated to the sliding window of size w. Therefore, the synthetic graph $\hat{G}_t$ generated by WGT-LDP at any time step t is a reasonable representation of the original graph G_t. This benefits from the fact that the population division-based sampling increases the available privacy budget at each time step, the data range estimation reduces the upper bound of the maximum possible edge weight (from B to $\hat{h}_{max}^t$), and the graph snapshot generation based on noisy degree sequence preserves the sparse structure of the graph.

5 EVALUATION

We conducted extensive experiments to evaluate the effectiveness of WGT-LDP. All experiments are conducted in Python on a laptop with Intel Core i5-1135G7 CPU, 16GB RAM. For each experiment, we performed it ten times and presented the average results.

5.1 Experimental Setup

Datasets. We first conduct experiments on three real-world graph datasets. Note that we regard the connection of nodes occurring within an hour is regarded as single connection.

- Email-Eu [32] contains 61046 dynamic email communications between 319 members. Each member on the network represents as a node. We use the number of communications between members to construct weighted topology of each graph snapshot.
- Forum [31] contains 33720 dynamic interaction records between 899 students in the community. We abstract each student as a node and construct weighted topology based on the number of interactions between students in a specific time window.

Table 1: Basic Information of Datasets

Datasets	n	T	B	Type
Email-Eu	319	173	72	communication
Forum	899	24	168	Social
Tech-AS	5000	24	24	Autonomous System
Synthetic-I & II	10000	100	200	Synthetic

- Tech-AS [35] records 171403 dynamic connections between 34761 autonomous systems. We randomly selected 5000 autonomous systems from the network as nodes, and construct the weighted topology of each snapshot according to the number of connections between these nodes in a specific time window.

Secondly, we use NetworkX [14], an open source package in python, to generate two synthetic datasets that follow a power-law distribution, namely Synthetic-I and Synthetic-II. Both synthetic datasets contain 100 weighted graph snapshots with 10000 nodes, and the edge weight of each snapshot is bounded by 200. The difference is that the severity of snapshot changes between adjacent time steps in Synthetic-I is 20%, while that in Synthetic-II is 90%.

Table 1 summarizes the basic information of all datasets, where n is the numbers of nodes, T is the number of snapshots and B is the common upper bound of the edge weights.

Parameter Settings. In our experiments, we set the threshold $\delta = 1$ for node sampling. The sliding window size w is set to 5. For the allocation of privacy budget, we set $\varepsilon_1 = \frac{1}{2}\varepsilon$ and $\varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \frac{1}{6}\varepsilon$, where the total privacy budget ε varies from 0.5 to 2.5.

Baselines. We compare WGT-LDP with the following baselines:

- **BDG.** This method is a baseline solution based on budget division, which does not perform node sampling but uniformly distributes the total privacy budget ε to w time steps in the sliding window.
- **RPG.** The baseline RPG adopts the population division-based solution. However, it does not consider the data changes of nodes, i.e., it randomly assigns all nodes to w time steps in the window.
- **Swg-NS.** The mechanism SwgDP [44] considers the dynamic weighted graphs publishing mechanism with DP. Since the node adaptive sampling component in SwgDP can be easily modified to the LDP setting, we use this component to perform node sampling and synthesize weighted graphs in the subsequent phases. We denote this baseline as Swg-NS.
- **HMG.** Li et al. [27] proposed HMG as an LDP mechanism to achieve dynamic graph publishing for decentralized applications. Since HMG does not consider the edge weights, we add noisy weights perturbed by the Laplace mechanism [9] to each edge after HMG generates the edges in the weighted graph.

Note that to ensure the rationality of the comparison, the Swg-NS and HMG are reproduced in the w-event-level privacy model.

In addition, we also implement two static unweighted graph mechanisms LDPGen [33] and Blink [48] for comparison, which can show the performance of our method on event-level privacy ($w = 1$). Since Blink is originally designed for training GNN models, we use their proposed variant Blink-Hard to generate synthetic graph at any time step. Both mechanisms are implemented under the privacy definition of edge LDP [33]. This definition can be viewed as the case where the value of each bit in Definition 2.2 is 0 or 1. Note that when $w = 1$, WGT-LDP does not sample users, but directly executes the last three phases to generate the entire weighted graph, which is equivalent to BDG and RPG.

Metrics. We evaluate the quality of the synthetic weighted graph snapshots from the following four aspects: degree distribution, weight distribution, clustering coefficient and path condition, where the first three are graph statistical queries and the last one is a graph structure query. Since these snapshots are continuously released, we calculate the average of each metric over all time steps. Note that for the clustering coefficient, we consider the Root Mean Squared Error (RMSE) over time. Smaller results indicate higher utility.

- **Degree Distribution.** We adopt Kullback-Leibler (KL) divergence [23] to evaluate the error of the degree distributions between the original and synthetic weighted graphs. To avoid the denominator in the KL divergence being zero, we add a small value to the degree distributions.
- **Weight Distribution.** We bucketize the weights into B bins and count the number of edges that fall into each bin to calculate the weight distribution. Similar to the degree distribution, we use KL divergence to measure the error.
- **Clustering Coefficient.** It is a statistical metric that characterizes the community structure of a graph. We use the RMSE of the clustering coefficient over all time steps to measure the error. See the technical report [45] for details.
- **Path Condition.** The path length of a graph measures its connectivity, which is denoted as the maximum number of edges between all nodes with a connected path. We use the Relative Error (RE) of path to evaluate the error. See the technical report [45] for details.

5.2 Comparison under w-Event Privacy

For each dataset, we first evaluate the utility of WGT-LDP with all baseline methods under the privacy guarantee of w-event edge weight LDP.

Utility on Different Privacy Budgets ε. Figure 3 shows the comparison of four metrics between WGT-LDP and all baselines when ε varies from 0.5 to 2.5. In general, we observe that WGT-LDP outperforms its competitors in most cases. This is because WGT-LDP determines the sampling nodes based on data change and the number of remaining nodes at each time step, which brings two benefits: 1) each node can perturb their data with sufficient privacy budget, reducing the perturbation error; 2) nodes with large change errors are preferentially sampled, reducing the approximation error. When the privacy budget ε is less than 1.5, WGT-LDP has higher RE on Email-Eu than RPG. This is because small graphs are more sensitive to noise and too small privacy budget reduces the sampling accuracy. We notice that the KL divergence of weight distribution of RPG on the Tech-AS is close to WGT-LDP, which may be due to the fact that the weights of most nodes on the Tech-AS vary greatly, resulting in similar approximation errors. In some cases, error slightly increases with larger ε due to the interaction between sampling randomness (e.g., in RPG) and graph structure. As noise decreases, the approximation error introduced by random sampling may dominate, leading to unexpected error patterns.

Utility on Different Window Sizes w. Figure 4 shows the utility of WGT-LDP and all baselines on four metrics, with different w. In these experiments, the privacy budget ε is fixed to 2. We can see

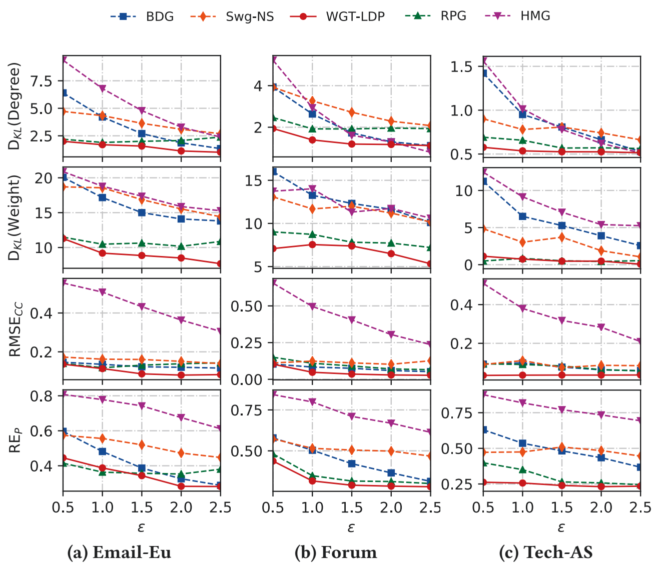


Figure 3: Utility comparison of four metrics on varying ε.

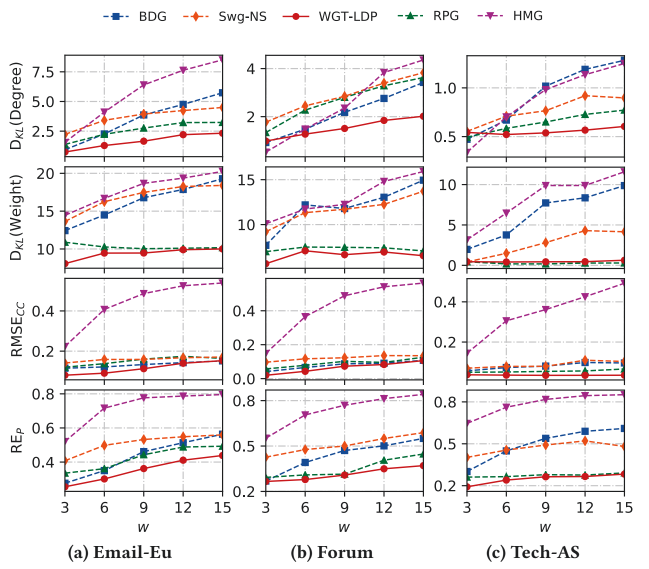


Figure 4: Utility comparison of four metrics on varying w.

that the errors of all methods on the four metrics generally increase with w. This is because the increase of w means that less privacy budget or nodes will be allocated to each time step. We can also observe that WGT-LDP acquires smaller error than the competitors on three datasets, which shows the effectiveness of our population division-based sampling and subsequent graph synthesis phases in improving utility. For HMG, we find that as w increases, its errors on many metrics such as clustering coefficient and path condition are increasingly different from our scheme. The reason is that the increase of w leads to greater noise, which causes this prediction based on historical data to become very inaccurate. Comparing the baselines BDG and RPG, although BDG performs better than RPG on the degree distribution of Forum, it performs worse than RPG on Tech-AS, which suggest that the relative utility of the two baselines depends on the dataset.

Effect of Threshold δ. As described in Section 3.2, WGT-LDP selects nodes with significant data changes using a threshold δ at each time step. Figure 5 presents the effect of varying δ (0.5–9) on the degree distribution under $\varepsilon = 2$. We find that the effect of varying δ on utility tends to level off as the dataset grows. The reason is as follows: When the total number of nodes n increases, the number of nodes m_t with data changes greater than δ increases significantly. At this time, we limit the number of sampling nodes to near $\frac{n}{w}$ by setting the sampling ratio, so that the adjustment of δ has little effect on the determination of the final sampling nodes. In Email-Eu and Forum, the utility first improves and then degrades as δ increases. This reflects a trade-off between perturbation noise and approximation error. When δ is small, many unchanged nodes are included, increasing the data dimension and noise. As δ grows, more relevant nodes are selected, improving utility. However, excessively large δ may exclude important changes, increasing approximation error. Overall, WGT-LDP achieves stable performance when δ is around 1 across datasets. Similar trends were observed for other metrics and are reported in the technical report of this paper [45].

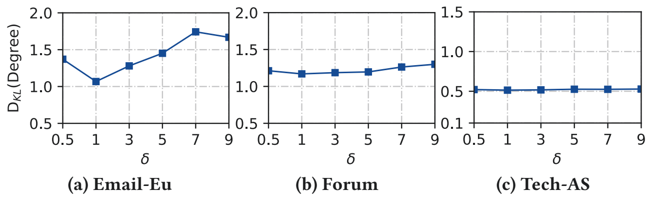


Figure 5: D_{KL}(Degree) metrics vs. Different threshold δ.

Effect of Privacy Budget Allocation. We evaluate the effect of different privacy budget allocation strategies using the Forum dataset. Specifically, we vary the ratio of ε_1, ε_2 and ε_3 to ε from 0.1 to 0.7 with step size 0.2. Then ε_4 is automatically calculated by $\varepsilon - \varepsilon_1 - \varepsilon_2 - \varepsilon_3$. Figure 6 shows the performance of WGT-LDP when the total privacy budget ε is 1. When ε_1 is small, the errors of most metrics are large. The reason is that the first phase needs to divide ε_1 into w time steps, and a small ε_1 further reduces the sampling accuracy. If the sampled nodes are inaccurate, it is difficult to reconstruct a high-quality synthetic graph. In addition, small ε_2, ε_3 and ε_4 have poor effects on different metrics. The above observations provide guidance for privacy budget allocation. That is, we should allocate more privacy budget to ε_1. In our experiments, we set ε_1 as $\frac{1}{2}\varepsilon$, and set ε_2, ε_3 and ε_4 as $\frac{1}{6}\varepsilon$. The results for the Email-Eu and Tech-AS datasets are shown in [45] due to space constraints.

Effect of the Number of Experimental Repetitions. Figure 7 illustrates the average utility of multiple repeated experiments on degree distribution. We can observe that when the number of runs is small, the utility of degree distribution varies greatly. However, as the number of runs increases, the results of most methods tend to be stable. This is because the random perturbations of the differential privacy mechanism will cause large fluctuations in the results of a single experiment. Therefore, by averaging multiple independent

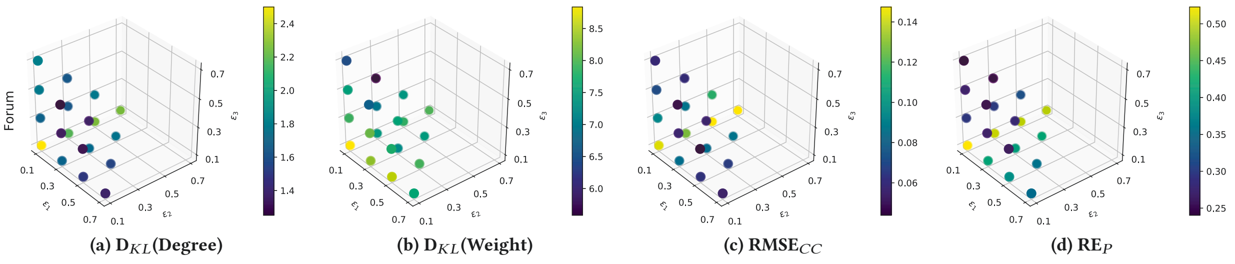

(a) D$_{KL}$(Degree) (b) D$_{KL}$(Weight) (c) RMSE$_{CC}$ (d) RE$_P$

Figure 6: Four evaluation metrics vs. Different privacy budget allocations.

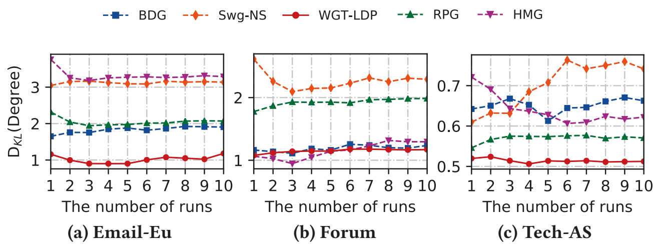


Figure 7: Average utility of multiple repeated experiments. The number of repetitions ranging from 1 to 10.

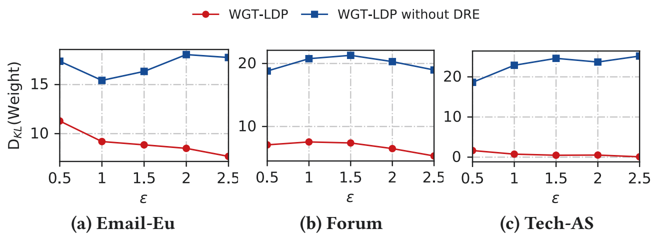


Figure 8: KL divergence of weight distribution with and without data range estimation.

experiments, the performance of the algorithm in the sense of expectation can be more reliably evaluated. In addition, WGT-LDP achieves the best performance in most cases, which proves the effectiveness of our mechanism on synthetic dynamic weighted graphs. The results of the other three metrics are provided in [45].
Effect of Data Range Estimation. We conduct experiments of WGT-LDP with and without data range estimation on three datasets to evaluate the effect of data range estimation. As shown in Figure 8, we can observe that data range estimation significantly reduces the KL divergence of weight distribution. The reason is that WGT-LDP with data range estimation avoids pathological worst case of considering edge weight perturbations by estimating the local upper bound for each node, which reduces the noise of edge weights in any snapshot. When the privacy budget is small, the estimation is not accurate enough, thus causing higher KL divergence.

5.3 Comparison under Event Privacy

To show the advantages of the last three stages of WGT-LDP for graph generation, we further provide performance comparisons

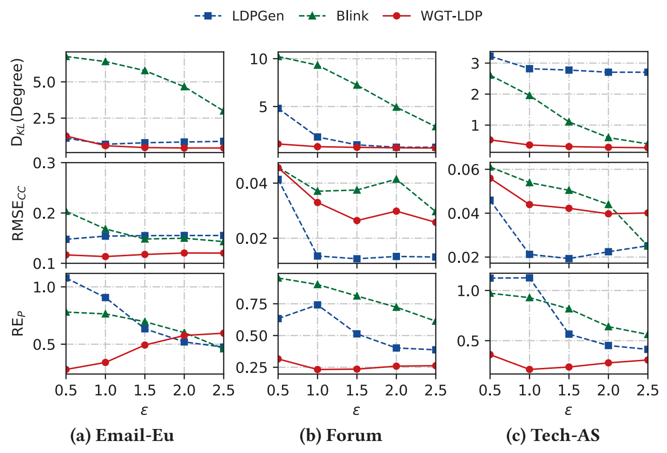


Figure 9: Utility comparison under event-level privacy.

when $w = 1$. In other words, we consider the scenario of event-level edge weight LDP. In this scenario, WGT-LDP protects the privacy of edges and weights on any single graph snapshot, which can be regarded as a static graph publication at any time step. Since static graph mechanisms LDPGen and Blink only focus on unweighted graphs, we compare all methods on three metrics, i.e., degree distribution, path condition, and clustering coefficient.

Figure 9 illustrates the experimental results on three datasets. Since WGT-LDP is designed for generating weighted graphs, it is necessary to allocate additional privacy budget for the reconstruction of edge weights. Nevertheless, we observe that WGT-LDP still achieves better accuracy than the baselines based on unweighted graphs in most cases. The reason is that WGT-LDP exploits the relationship between edge weights and graph structure, which can be used to guide the synthesis of weighted graphs. For clustering coefficient, LDPGen performs best on Forum and Tech-AS because it uses the BTER [37] model to generate a synthetic graph, which is specifically optimized for returning accurate clustering coefficients.

5.4 Evaluation on Synthetic Datasets

To evaluate the impact of snapshot variability and the larger window size w, we compare the utility of WGT-LDP with baselines on two synthetic datasets given different privacy budgets and larger w. For efficiency reasons, we omit the baseline BDG due to its high computational overhead at this scale.

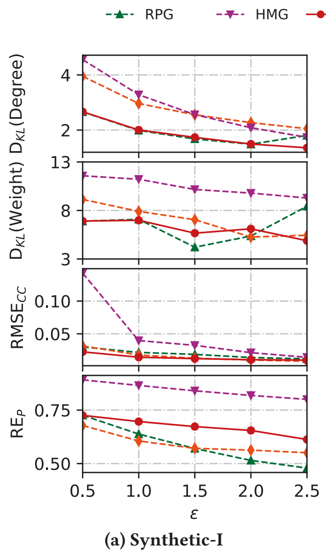


Figure 10: Evaluation on synthetic datasets given different ε.

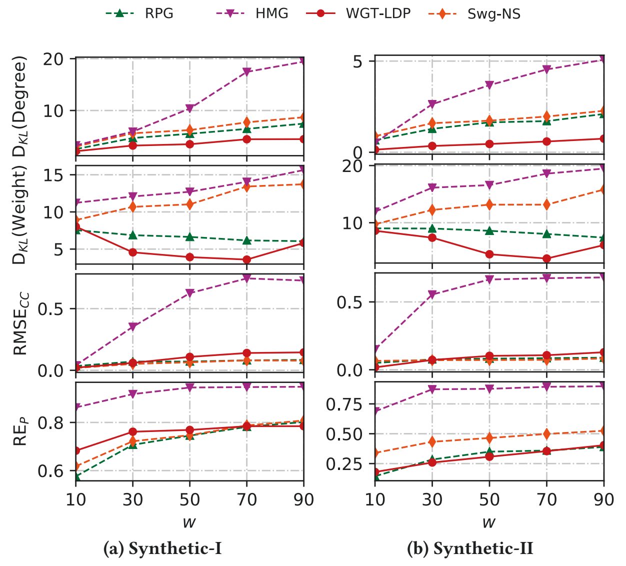


Figure 11: Evaluation on synthetic datasets given different w.

Figure 10 shows the utility across four metrics under different privacy budgets. We can find that WGT-LDP performs better than other methods on Synthetic-II, but its advantage on Synthetic-I is not obvious. This is because the severity of snapshot changes in Synthetic-II is higher than that in Synthetic-I. For WGT-LDP, when the degree of data change is large, it can effectively capture nodes with large change errors, which improves the utility of graph synthesis. For the baseline RPG, we observed that when the privacy budget is large, the baseline RPG shows an upward trend on some metrics as the privacy budget increases. The reason is that although the large privacy budget limits the impact of injected noise, the random sampling method of RPG causes some areas with important graph features to produce larger approximation errors.

Figure 11 shows the impact of increasing w from 10 to 90. We also find that WGT-LDP has a more obvious advantage on Synthetic-II than on Synthetic-I. For clustering coefficient, RPG and Swg-NS slightly outperform our method. This is because the clustering coefficient of the synthetic dataset is very low, i.e., there is no significant community structure, and the sampling strategy of RPG and Swg-NS increases the dispersion of the graph structure, which in turn leads to a smaller RMSE. Nevertheless, the overall performance of WGT-LDP remains competitive, especially on Synthetic-II. In addition, we observe that as w increases, our method first performs better and then worse on weight distribution, which is caused by the combined effect of perturbation error and approximation error. When w is small, more nodes are sampled and we set a larger upper bound on weights in the synthetic graph, making the perturbation noise of weights large. When w is large, more data is approximated, which aggravates the approximation error.

5.5 Case Study

We apply WGT-LDP to an end-to-end use case, i.e., the influence maximization (IM) problem [6], to evaluate the utility of synthetic weighted graph snapshots for network analytics. IM aims to locate nodes from the network that can achieve the maximum impact spread, and has applications in viral marketing [6], epidemic control [5], network monitoring [30], and so on. For example, in the event of an infectious disease outbreak (such as COVID-19), super spreaders can be identified to prioritize quarantine or vaccination.

Specifically, we apply the Degree Discount (DD) method [4] to select the top 20 most influential nodes from each synthetic graph snapshot, where node degree is defined as the sum of edge weights. We then simulate the Independent Cascade (IC) model [22] to estimate influence spread, using spread probability $1-(1-p)^{a(u,v)}$ for each edge (u,v) with weight $a(u,v)$ and $p=0.01$. Higher influence spread values indicate higher utility, i.e., the locations of the most influential nodes are more accurate.

Figure 12 shows the influence spread under varying privacy budgets. WGT-LDP consistently achieves the highest spread across all datasets. This demonstrates the effectiveness of our scheme in recovering graph structure and edge weights, which is crucial for locating influential nodes. For other baselines, we find that RPG significantly outperforms the other three baselines when the privacy budget is small. This is because BDG, HMG and Swg-NS further split the already small privacy budget, resulting in a significant increase in noise, while the random population division of RPG allows each user to hold the entire privacy budget.

Figure 13 shows influence spread under varying window sizes w. WGT-LDP also maintains excellent performance relative to all baselines. In addition, the influence spread of all methods generally decreases with the increase of w, mainly due to the available privacy budget or nodes at each time step decreases, posing a challenge to synthetic graphs.

5.6 Utility on Different Time Steps

To achieve a more comprehensive analysis, we present the utility of WGT-LDP and all baselines in different time steps when ε is 2. Due to space constraints, we provide the degree distribution

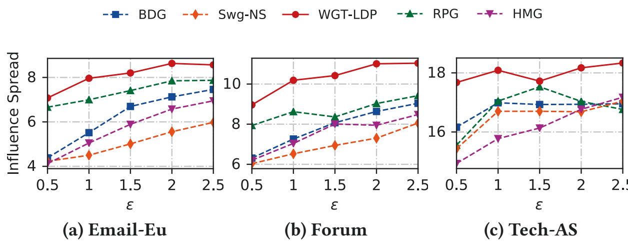


Figure 12: The influence spread given different ε.

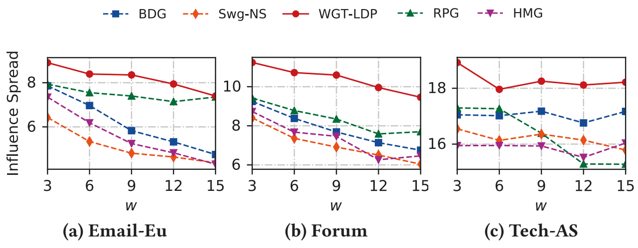


Figure 13: The influence spread given different w.

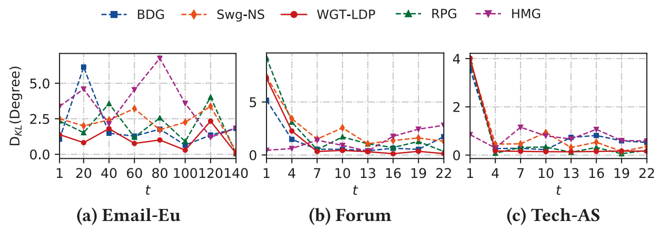


Figure 14: Utility of methods with different time steps.

performance and defer the results of other metrics to [45]. As shown in Figure 14, we observe that WGT-LDP consistently outperforms other methods in most time steps. This highlights the key role that our node sampling mechanism, weight optimization method, and synthetic graph snapshot step play in preserving graph features.

6 RELATED WORK

Time Series Data Analysis via DP. There are three levels of local differential privacy protection for time series data analysis: event-level, user-level and w-event-level. Event-level LDP aims to hide a single event in a time series [19, 39]. User-level LDP tries to hide all events of a single user [1, 11]. w-event-level LDP aims to protect any event sequence occurring within any window of w time steps. Since w-event-level LDP can balance the privacy loss and utility loss between event-level LDP and user-level LDP, it has been widely used by many works recently [16, 26, 28, 34, 41]. For example, Wang et al. [41] proposed a metric-based w-event-level LDP to protect the important patterns of time series. Li et al. [28] extended the above pattern protection to the scenario of finite data range. Ren et al. [34] introduced a budget division strategy and a population-based sampling mechanism for streaming tabular data publication under LDP. Hu et al. [16] explored real-time trajectory synthesis, which generates high-utility trajectory data by extracting movement patterns from users' trajectory streams. However, these algorithms in the above works are tailored to their respective data formats and do not account for graph topology, edge correlations or weight semantics.

Static Graph Publication via DP. Many prior works [2, 3, 17, 18, 33, 42, 43, 46] focus on releasing static graphs under differential privacy guarantees. For instance, Qin et al. [33] first proposed to generate a synthetic social graph under LDP. Wei et al. [42] additionally considered node attributes and generated attributed social graph in a decentralized network. Recently, Yuan et al. [46] utilized community information to publish a synthetic graph with DP, which improves the accuracy of graph reconstruction. Brito et al. [2] designed a DP algorithm to generate count-weighted graph while protecting one interaction between nodes. However, these studies overlook the temporal properties of graphs and may be challenging to apply in our continuous publication scenario.

Dynamic Graph Publication via DP. Several works [27, 44, 47] address the temporal or dynamic properties of graphs. Specifically, Yuan et al. [47] proposed a framework for publishing stream graphs under DP, which uses communities as granularity and achieves w-event-level privacy. Xu et al. [44] initially investigated the problem of dynamic weighted graph publication and proposed a new weighted graph snapshot publication mechanism called SwgDP. SwgDP guides current snapshot generation by leveraging historical graph data. However, SwgDP only provides event-level privacy in the central DP setting, which offers relatively weak guarantees and assumes a trusted curator. Li et al. [27] considered dynamic graph publishing with LDP, but similarly limited their privacy protection to event-level LDP. In summary, all these studies rely on event-level privacy under DP or LDP, or on w-event-level privacy in the central DP setting, without addressing the combination of dynamic graph structure, edge weights and local w-event-level privacy. To the best of our knowledge, this work is the first to support continuous publishing of dynamic weighted graphs under w-event-level LDP, bridging a crucial gap in the literature.

7 CONCLUSION

In this paper, we propose WGT-LDP, a novel framework for the continuous publication of weighted graph snapshots while ensuring w-event edge weight LDP. By incorporating population division-based sampling, data range estimation, aggregate information collection, and graph snapshot generation, our approach effectively balances privacy protection and utility preservation. Theoretical analysis and extensive experiments on real-world and synthetic datasets demonstrate that WGT-LDP significantly outperforms baseline methods by reducing perturbation noise and improving graph utility. Future work will explore personalized privacy protection and extensions to heterogeneous graphs, further advancing privacy-preserving graph generation in decentralized settings.

ACKNOWLEDGMENTS

This work was supported in part by the Key International Cooperation Project of NSFC (W2411053), NSFC (U23B2027), JSPS KAKENHI (JP23K24851), JST PRESTO (JPMJPR23P5), and JST CREST (JPMJCR21M2). Pengpeng Qiao is the corresponding author.

REFERENCES

[1] Ergute Bao, Yin Yang, Xiaokui Xiao, and Bolin Ding. 2021. CGM: an enhanced mechanism for streaming data collection with local differential privacy. *VLDB* 14, 11 (2021), 2258–2270.

[2] Felipe T Brito, Victor AE Farias, Cheryl Flynn, Subhabrata Majumdar, Javam C Machado, and Divesh Srivastava. 2023. Global and local differentially private release of count-weighted graphs. *SIGMOD* 1, 2 (2023), 1–25.

[3] Rui Chen, Benjamin CM Fung, Philip S Yu, and Bipin C Desai. 2014. Correlated network data publication via differential privacy. *The VLDB Journal* 23 (2014), 653–676.

[4] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 199–208.

[5] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. 2003. Efficient immunization strategies for computer networks and populations. *Physical review letters* 91, 24 (2003), 247901.

[6] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 57–66.

[7] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *2013 IEEE 54th annual symposium on foundations of computer science*. IEEE, 429–438.

[8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[10] Ken TD Eames, Jonathan M Read, and W John Edmunds. 2009. Epidemic prediction and control in weighted networks. *Epidemics* 1, 1 (2009), 70–76.

[11] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2468–2479.

[12] Stephen Eubank, Hasan Guclu, VS Anil Kumar, Madhav V Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. 2004. Modelling disease outbreaks in realistic urban social networks. *Nature* 429, 6988 (2004), 180–184.

[13] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2009. Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 351–360.

[14] Aric Hagberg, Pieter J Swart, and Daniel A Schult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

[15] Sudheendra Hangal, Diana MacLean, Monica S Lam, and Jeffrey Heer. 2010. All friends are not equal: Using weights in social graphs to improve search. In *Workshop on Social Network Mining & Analysis, ACM KDD*, Vol. 130.

[16] Yujia Hu, Yuntao Du, Zhikun Zhang, Ziquan Fang, Lu Chen, Kai Zheng, and Yunjun Gao. 2024. Real-Time Trajectory Synthesis with Local Differential Privacy. *arXiv preprint arXiv:2404.11450* (2024).

[17] Xun Jian, Yue Wang, and Lei Chen. 2021. Publishing graphs under node differential privacy. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 4164–4177.

[18] Zach Jorgensen, Ting Yu, and Graham Cormode. 2016. Publishing attributed social graphs with formal privacy guarantees. In *SIGMOD*. 107–122.

[19] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. 2018. Local differential privacy for evolving data. *NeuIPS* 31 (2018).

[20] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[21] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially private event sequences over infinite streams. (2014).

[22] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 137–146.

[23] Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

[24] Youhuan Li, Lei Zou, M Tamer Özsu, and Dongyan Zhao. 2020. Space-Efficient Subgraph Search Over Streaming Graph With Timing Order Constraint. *TKDE* 34, 9 (2020), 4453–4467.

[25] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. 2020. Estimating numerical distributions under local differential privacy. In *SIGMOD*. 621–635.

[26] Zhetao Li, Junru Wu, Saiqin Long, Zhirun Zheng, Chengxin Li, and Mianxiong Dong. 2024. User-Driven Privacy-Preserving Data Streams Release for Multi-Task Assignment in Mobile Crowdsensing. *TMC* (2024).

[27] Zhetao Li, Yong Xiao, Haolin Liu, Xiaofei Liao, Ye Yuan, and Junzhao Du. 2025. Dynamic Graph Publication with Differential Privacy Guarantees for Decentralized Applications. *IEEE Trans. Comput.* (2025).

[28] Zhetao Li, Xiyu Zeng, Yong Xiao, Chengxin Li, Wentai Wu, and Haolin Liu. 2024. Pattern-sensitive Local Differential Privacy for Finite-Range Time-series Data in Mobile Crowdsensing. *TMC* (2024).

[29] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*. 19–30.

[30] Adilson E Motter. 2004. Cascade control and defense in complex networks. *Physical Review Letters* 93, 9 (2004), 098701.

[31] Tore Opsahl. 2013. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social networks* 35, 2 (2013), 159–167.

[32] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *WSDM*. 601–610.

[33] 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*. 425–438.

[34] 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*. 1064–1077.

[35] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *AAAI*, Vol. 29.

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

[37] Comandur Seshadhri, Tamara G Kolda, and Ali Pinar. 2012. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 85, 5 (2012), 056109.

[38] Connor Wagaman, Palak Jain, and Adam Smith. 2024. Time-Aware Projections: Truly Node-Private Graph Statistics under Continual Observation. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 237–237.

[39] 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*. 1237–1253.

[40] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. 2019. Locally differentially private frequency estimation with consistency. *arXiv preprint arXiv:1905.08320* (2019).

[41] Zhibo Wang, Wenxin Liu, Xiaoyi Pang, Ju Ren, Zhe Liu, and Yongle Chen. 2020. Towards pattern-aware privacy-preserving real-time data collection. In *INFOCOM*. IEEE, 109–118.

[42] Chengkun Wei, Shouling Ji, Changchang Liu, Wenzhi Chen, and Ting Wang. 2020. AsgLDP: Collecting and generating decentralized attributed graphs with local differential privacy. *TIFS* 15 (2020), 3239–3254.

[43] Qian Xiao, Rui Chen, and Kian-Lee Tan. 2014. Differentially private network data release via structural inference. In *SIGKDD*. 911–920.

[44] Wen Xu, Zhetao Li, Haolin Liu, Yunjun Gao, Xiaofei Liao, and Kenli Li. 2024. Differentially Private Weighted Graphs Publication Under Continuous Monitoring. *TMC* (2024).

[45] Wen Xu, Pengpeng Qiao, Shang Liu, Zhirun Zheng, Yang Cao, and Zhetao Li. 2025. Technical Report. https://github.com/xuwen22/WGT-LDP/blob/main/technical_report.pdf.

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

[47] Quan Yuan, Zhikun Zhang, Linkang Du, Min Chen, Mingyang Sun, Yunjun Gao, Michael Backes, Shibo He, and Jiming Chen. 2024. PSGraph: Differentially Private Streaming Graph Synthesis by Considering Temporal Dynamics. *arXiv preprint arXiv:2412.11369* (2024).

[48] Xiaochen Zhu, Vincent YF Tan, and Xiaokui Xiao. 2023. Blink: Link Local Differential Privacy in Graph Neural Networks via Bayesian Estimation. In *CCS*. 2651–2664.