

Local Distance Query with Differential Privacy

Weihong Sheng, Jiajun Chen, Bin Cai, Chunqiang Hu, Meng Han, Jiguo Yu

Abstract—Differential Privacy (DP) is commonly employed to safeguard graph analysis or publishing. Distance, a critical factor in graph analysis, is typically handled using curator DP, where a trusted curator holds the complete neighbor lists of all vertices and answers queries privately. However, in many real-world scenarios, such a curator may not be present, posing a significant challenge for implementing differentially private distance queries under Local Differential Privacy (LDP). This paper proposes two approaches to address this challenge. The first approach generates a synthetic graph by randomizing responses and applies bitwise operations to reduce noise interference. However, like other synthetic graph methods, this approach suffers from low utility. To overcome this limitation, we propose a second approach, the first LDP method specifically designed for distance queries, which captures the global graph structure by continuously aggregating local distance vectors from neighboring vertices. This process enables the accurate updating of global distances. We demonstrate the effectiveness of our method through comprehensive theoretical analysis and experimental evaluations on real-world datasets.

Index Terms—local differential privacy, distance, graph, neighbor aggregation.

I. INTRODUCTION

Amid frequent privacy breaches, ensuring privacy in data collection and analysis has become crucial to addressing public concerns. Without a privacy promise, users may hesitate to provide personal data that could improve service quality. Differential Privacy (DP), introduced formally by Dwork et al. [1], has attracted significant research interest due to its strong privacy guarantees and versatile applicability. Notable applications of DP include deployments by major corporations such as Microsoft [2], Apple [3] and Google [4], enhancing the privacy of their services.

Privacy-preserving analysis on graphs has become an important research trend, driven by the widespread use of graph-structured data and growing concerns about data privacy [5], [6]. However, when it comes to distance queries, such as computing the shortest path between two vertices or all-pairs shortest paths in unweighted graphs, most existing studies rely on the curator model of DP, in which a trusted curator possesses the entire graph and answers queries in a privacy-preserving manner [7], [8], [9], [10]. The existence of such a curator is not guaranteed in practical scenarios, and even if one is present, data owners may incur a trust cost, fearing that the curator could compromise their privacy. Alternatively, Local Differential Privacy (LDP) eliminates the need for a trusted curator by sanitizing data before it leaves the user's control.

Consider a distributed social network where vertices represent users and edges denote sensitive friendships, with each user holding their own friend list. A service provider maintains the network and often needs to collect statistics, such as average distance, to improve its services. While users may be

willing to contribute, they do so only if privacy is preserved. Furthermore, consider the two cases:

- **Case 1: Analyzing contact networks during an epidemic outbreak.** Distance queries on contact networks can help identify 3-hop contacts or assess transmission risks. However, due to privacy concerns, individuals may be unwilling to share data or may provide inaccurate information.
- **Case 2: Analyzing communication networks among sensitive populations.** For groups such as investigative journalists or sex workers, communication often occurs via decentralized encrypted networks. The challenge lies in assessing network robustness against censorship or attacks without revealing specific interpersonal connections.

Therefore, a significant challenge remains: how to perform differentially private distance queries without relying on a curator who possesses all neighbor lists.

One possible solution to this challenge is to generate synthetic graphs. Specifically, this involves privately collecting degree and other auxiliary information from each local vertex to construct a synthetic graph using random graph generation techniques [11], [12]. However, this approach tends to introduce substantial noise, which can severely distort the community structure [13]. As an alternative to synthetic graph approaches, one strategy is to directly collect statistical information from each local vertex. This approach avoids the complexity of synthetic graphs but faces challenges in accurately capturing distances, which require knowledge of the global graph structure, unlike simple metrics such as subgraph counts.

In this paper, we address the challenge of answering distance queries under LDP and propose two innovative approaches. The first approach involves generating synthetic graphs through randomized responses, with the aim of preserving the global graph structure. To mitigate the damage to the graph structure caused by randomized responses, we employ post-processing techniques using **AND** and **OR** operations. However, this approach still suffers from inherently low utility, as the added noise significantly disrupts structural integrity.

To address this limitation, we propose a second approach inspired by GNN. In this approach, each vertex maintains a local distance vector, initially containing distances only to its immediate neighbors. By continuously aggregating these local distance vectors from their neighbors, the vertices can reconstruct the global structure and obtain the global distances.

The main contributions of this paper are summarized as follows:

- We enhance randomized response-based synthetic graph generation using **AND** and **OR** operations, which help

reduce noise and more accurately preserve the underlying graph density. Furthermore, we provide a detailed theoretical analysis of the density errors introduced by the perturbation process.

- We design a novel distance aggregation mechanism that reconstructs global distances by iteratively aggregating local distance vectors from neighboring vertices. We further provide theoretical guarantees for both Laplace noise and randomized response, offering guidance on selecting optimal perturbation mechanisms.
- We demonstrate the effectiveness of the proposed distance query mechanisms through comprehensive numerical evaluations and empirical analysis on real-world datasets.

The rest of the paper is as follows. In Section II, we show the background knowledge covered in this paper. In Section III, we introduce our synthetic graph generation approach. In Section IV, we detail our distance aggregation approach. In Section V, we describe the experimental design and results. In Section VI, we present related work. Finally, in Section VII, we summarize this paper.

II. PRELIMINARIES

A. Problem Definition

Let \mathbf{N}_i represent the neighbor list of vertex $i \in [n]$, corresponding to the i -th row of the adjacency matrix. The query function f , a distance query function, returns all-pair distances when applied to $(\mathbf{N}_1, \mathbf{N}_2, \dots, \mathbf{N}_n)$. Each edge, connecting any vertices i and j , is undirected. Given that edges are considered private, vertices avoid disclosing their neighbors when responding to the query f . Let \mathcal{M} denote a randomized mechanism that provides answers to all-pair distance queries while preserving privacy. Designing \mathcal{M} , which can answer all-pair distances with low error while providing differential guarantee for edges, is the main purpose of this paper.

B. Differential Privacy

Differential privacy [1] is a formal privacy standard that ensures privacy by minimizing the influence of any individual's record on the outcome of a query. Any adversary cannot obtain additional knowledge by observing the query outputs from two databases: one that includes and one that excludes the record of a target individual.

Let us consider the form of DP on the graph. Suppose G is a simple graph with n vertices and m edges. The privacy-sensitive information we aim to protect is the presence of any edge (or relationship) within G during the execution of distance queries on G .

Definition II.1. (Neighboring Datasets[1]) Let $G, G' \in \mathcal{G}$ be two undirected graphs. We say $G = (V, E)$ and $G' = (V, E')$ are neighboring datasets, denoted as $G \sim G'$, if they have the same vertex sets but edge sets differ in one edge: $|E - E'| = 1$.

Definition II.2. (Edge Differential Privacy[14]) Let $\mathcal{M} : \mathcal{G} \rightarrow \mathcal{S}^n$ be a randomized mechanism. Given any neighboring

databases G and G' , which differ in one edge, then \mathcal{M} satisfies ε -DP if, for all $S \in \text{Range}(\mathcal{M})$, it holds that

$$\Pr[\mathcal{M}(G) \in S] \leq \exp(\varepsilon) \times \Pr[\mathcal{M}(G') \in S]. \quad (1)$$

The parameters ε quantify the privacy loss of \mathcal{M} , with smaller value indicating stronger privacy guarantees. Specifically, ε constrains the probability ratio between the outputs of \mathcal{M} on database G and G' . DP can be achieved by adding Laplace noise, if the query f is numeric.

Definition II.3. (Laplace Mechanism[1]) Let $f : \mathcal{G} \rightarrow \mathcal{S}^n$ be a distance query function. $\mathcal{M} : \mathcal{G} \rightarrow \mathcal{S}^n$ satisfies ε -DP, if

$$\mathcal{M}(G) = f(G) + \frac{\Delta f}{\varepsilon} X^n, \quad (2)$$

where X^n represents n independent and identically distributed (i.i.d.) Laplace random variables and Δf is the sensitivity of f , which is

$$\Delta f = \max_{G \sim G'} \|f(G) - f(G')\|_1. \quad (3)$$

Lemma II.1. (Post-Processing Immunity[15]) Let $\mathcal{F} : \mathcal{S} \rightarrow \mathcal{Y}$ be a randomized mapping and \mathcal{M} satisfy ε -DP. For the mechanism $\mathcal{F} \circ \mathcal{M}$, it also satisfies ε -DP.

Post-processing immunity is an inherent feature of any DP mechanism. It maintains the established privacy guarantees, even when arbitrary analysis is performed on the results of private queries.

Lemma II.2. (Sequential Composition[15]) Let $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_k\}$ where \mathcal{M}_i satisfies ε_i -DP. \mathcal{M} satisfies $\sum_{i=1}^k \varepsilon_i$ -DP, if every \mathcal{M}_i is performed on the same database.

C. Local Differential Privacy

In local differential privacy, there is no trustworthy curator when executing a query on a database. In this model, each record is maintained independently by its holder. Without additional trust expenditure, every data holder transmits a perturbed version of their data to an untrusted curator for analyzing. For a distributed graph G , each vertex independently manages its neighbor list. We use $\mathbf{N}_u \in \{0, 1\}^n$ to denote the view (its neighbors) of the vertex u . For example, for $i \in [n]$, if $\mathbf{N}_u[i] = 1$ or 0 , indicates that the edge $e(u, i)$ exists or not, respectively. Then, the adjacency matrix \mathbf{A} of G can be denoted as $\{\mathbf{N}_1, \dots, \mathbf{N}_n\}$. We can define LDP under distributed graph.

Definition II.4. (Local Edge Differential Privacy) Suppose G and G' differ in one edge. Let $\mathbf{N}_u, \mathbf{N}'_u \in \mathcal{X}^n$ be two neighbor lists of any vertex u in G, G' , respectively. We can say $\mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{X}^n$ satisfies ε -LDP, if for any possible $S \in \text{range}(\mathcal{M})$, have

$$\Pr[\mathcal{M}(\mathbf{N}_u) \in S] \leq \exp(\varepsilon) \times \Pr[\mathcal{M}(\mathbf{N}'_u) \in S]. \quad (4)$$

In addition to Laplace noise, the randomized response [16] can also be used to provide the LDP guarantee. Let \mathcal{R} be a randomized response mechanism, for a binary value b_i

$$\mathcal{R}(b_i) = \begin{cases} b_i & \text{with probability } 1 - p \\ 1 - b_i & \text{with probability } p \end{cases}. \quad (5)$$

If $p = \frac{1}{1+\exp(\varepsilon)}$, $\mathcal{M} = \{\mathcal{R}_1, \dots, \mathcal{R}_n\}$ satisfies ε -LDP.

With randomized response, LDP can provide ε -LDP for all vertices if their neighbor lists are independent. However, this independence may not be established since two vertices share one edge. Specifically, if $N_u[i] = 1$ or 0, there must be $N_i[u] = 1$ or 0, respectively. The privacy guarantee for a specific edge is influenced by two vertices, which increases the associated privacy risk. Previous studies have suggested that each vertex can report a subset of its neighbors to preserve independence [13], [17], [18]. For instance, given an undirected graph, its adjacency matrix \mathbf{A} is symmetric. For vertex $u \in [n]$, it only reports neighbors $N_u[u+1:]$ with \mathcal{R} . The ε -LDP privacy guarantee can be preserved for each edge.

Intuitively, if each vertex chooses to report all its neighbors, and with \mathcal{R} applied to each edge, the 2ε -LDP can be held for each edge.

III. GRAPH AGGREGATION APPROACH

In this section, we introduce a graph aggregation approach designed to support distance queries on synthetic graphs. This approach addresses a challenge identified by Qin et al. [13], specifically that direct application of randomized responses tends to produce a denser graph than the original. To mitigate this effect, our approach utilizes bitwise operations to refine the graph structure.

A. The Motivation

Let us review the issue highlighted by Qin et al. [13]. Each vertex only owns a limited view (its neighbors) of the whole graph. An untrusted curator, tasked with aggregating this local information to create a synthetic graph, requests each vertex to submit their neighbor list. With privacy concerns, each vertex employs a randomized response technique to perturb its entire list of neighbors. For each neighbor relationship of u

$$\mathcal{R}(N_u[i]) = \begin{cases} N_u[i] & \text{with probability } 1-p \\ 1 - N_u[i] & \text{with probability } p \end{cases}. \quad (6)$$

By aggregating all neighbor lists from all local vertices, the curator can obtain a masked graph. However, the expected density of this graph is $\gamma(1-p) + (1-\gamma)p$ if the original density is γ . For example, if $\gamma = 0.1$ and $p = 0.1$, the expected density is 0.18, which is approximately twice the original value. These extra edges will severely destroy the graph structure, compared with the local graph. With this naive method, the density remains constant only under conditions where $p = 0$ or $\gamma = 1/2$. However, $p = 0$ means serious privacy leakage, and $\gamma = 1/2$ is too strict for real-world graphs.

B. Graph Aggregation

Algorithm 1 outlines our two-round graph aggregation process. In the first round (Steps 1 to 6), each local vertex sends their degree, adjusted with Laplace noise $\frac{2}{\varepsilon_1} \times \text{Lap}(1)$, to the third curator. The curator aggregates these to estimate the density as $\hat{\gamma} \leftarrow \frac{\sum_{i=1}^n \hat{d}_i}{n(n-1)}$ and then broadcasts this estimate to each vertex.

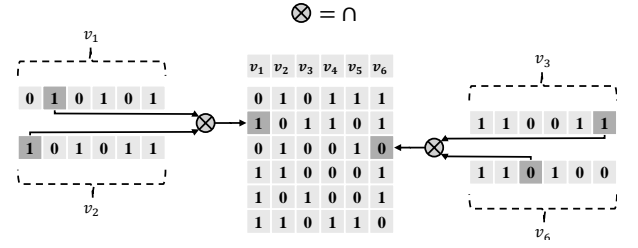


Fig. 1. An example of using AND operation to tune the edges

In the second round (Steps 7 to 17), each vertex perturbs all their neighbor relationships through a randomized response \mathcal{R} . After randomization, each vertex transmits their noisy neighbor list \hat{N}_i to the curator. Since each relationship is reported twice, the curator confirms the existence of an edge using the AND operator. That is, an edge $e(u, v)$ exists if both corresponding vertices u and v report it (i.e., $\hat{N}_u[v]$ and $\hat{N}_v[u]$ both equal to 1). In this case, the curator records the edge's existence as $\hat{A}[u][v] = 1$.

Algorithm 1 Graph Aggregation

Input: Privacy parameters ε_1 and ε_2

Output: Noisy graph $\hat{\mathbf{A}}$

```

1: for each vertex  $i = 1$  to  $n$  do
2:    $\hat{d}_i \leftarrow d_i + \frac{2}{\varepsilon_1} \times \text{Lap}(1)$ 
3:   send  $\hat{d}_i$  to curator
4: end for
5:  $\hat{\gamma} \leftarrow \frac{\sum_{i=1}^n \hat{d}_i}{n(n-1)}$ 
6: send  $\hat{\gamma}$  to vertices
7:  $p \leftarrow \frac{1}{\exp(\varepsilon_2)+1}$ 
8: for  $i = 1$  to  $n$  do
9:    $\hat{N}_i \leftarrow (\mathcal{R}(N_i[1]), \dots, \mathcal{R}(N_i[n]))$ 
10:  send  $\hat{N}_i$  to curator
11: end for
12:
13: for  $i = 1$  to  $n$  do
14:   for  $j = i+1$  to  $n$  do
15:      $\hat{A}[i][j] \leftarrow \hat{N}_i[j] \cap \hat{N}_j[i]$ 
16:      $\hat{A}[j][i] \leftarrow \hat{N}_i[j] \cap \hat{N}_j[i]$ 
17:   end for
18: end for
19:
20: return  $\hat{\mathbf{A}}$ 

```

Annotations for Algorithm 1:

- Steps 1-3: by vertices.
- Steps 4-6: by curator.
- Steps 8-11: by vertices.
- Steps 13-17: by curator.

Example III.1. Figure 1 illustrates the use of the AND operation to tune the existence of edges. In the second round, vertices v_1, v_2, v_3 , and v_4 report their noisy neighbor lists. Using the AND operation, the curator concludes that the edge $e(v_1, v_2)$ exists, as $\hat{N}_1[2] \cap \hat{N}_2[1] = 1$. Conversely, the edge $e(v_3, v_6)$ is determined to not exist, since $\hat{N}_3[6] \cap \hat{N}_6[3] = 0$.

Theorem III.1. Algorithm 1 satisfies $2(\varepsilon_1 + \varepsilon_2)$ -LDP for every edge.

Proof: the first round provides $2\varepsilon_1$ -LDP for edges, since

two vertices share each edge. In the second round, each edge is reported with \mathcal{R} twice. Then, the **AND** operation executed by the curator qualifies as post-processing. With Lemma II.2, Algorithm 1 satisfies $2(\varepsilon_1 + \varepsilon_2)$ -LDP for all edges. ■

Theorem III.2. When $\varepsilon_2 = \ln\left(\frac{1}{2\hat{\gamma}} - 1\right)$, the expectation of the density error $\mathbf{E}[\bar{\gamma} - \gamma] = O\left(\frac{1}{\varepsilon_1^2 n(n-1)^2}\right)$, where $\bar{\gamma}$ is the density of graph $\hat{\mathbf{A}}$.

Proof: with the **AND** operation, we have

$$\mathbf{P}\left(\hat{\mathbf{A}}[i][j] = 1 \mid \mathbf{N}_i[j] = 1\right) = (1-p)^2, \quad (7)$$

$$\mathbf{P}\left(\hat{\mathbf{A}}[i][j] = 1 \mid \mathbf{N}_i[j] = 0\right) = p^2. \quad (8)$$

Thus, we have the expected $\bar{\gamma}$

$$\bar{\gamma} = (1-p)^2 \cdot \gamma + (1-\gamma) \cdot p^2, \quad (9)$$

$$\mathbf{E}[\bar{\gamma}] = \mathbf{E}[\gamma + p^2 - 2p\gamma] \quad (10)$$

$$= \mathbf{E}[\gamma] + \mathbf{E}[p^2] - \mathbf{E}[2p\gamma]. \quad (11)$$

To make sure $\bar{\gamma}$ is similar to γ , we assume $p = a\hat{\gamma} + b$ where a and b are constants. Since

$$\hat{\gamma} = \frac{\sum_{i=1}^n d_i}{n(n-1)} + \frac{2 \sum_{i=1}^n X_i}{\varepsilon_1 n(n-1)} = \gamma + \frac{2 \sum_{i=1}^n X_i}{\varepsilon_1 n(n-1)}, \quad (12)$$

where X_i is a Laplace random variable with mean 0 and variance 2. Thus, we have

$$\text{var}[p] = a^2 \cdot \text{var}[\hat{\gamma}] = \frac{4a^2}{\varepsilon_1^2 n(n-1)^2}, \quad (13)$$

$$\mathbf{E}[p^2] = \text{var}[p] + (\mathbf{E}[p])^2 \quad (14)$$

$$= \frac{4a^2}{\varepsilon_1^2 n(n-1)^2} + (a\gamma + b)^2. \quad (15)$$

With above results, we can have

$$\mathbf{E}[\bar{\gamma}] = \gamma + \frac{4a^2}{\varepsilon_1^2 n(n-1)^2} + (a\gamma + b)^2 - 2\gamma(a\gamma + b). \quad (16)$$

For $a = 2, b = 0$, we can get small error

$$\mathbf{E}[\bar{\gamma} - \gamma] = \frac{16}{\varepsilon_1^2 n(n-1)^2}. \quad (17)$$

At this moment, $p = 2\hat{\gamma}$, and thus $\varepsilon_2 = \ln\left(\frac{1}{2\hat{\gamma}} - 1\right)$ holds. ■

When ε_2 is fixed, setting $p = 2\hat{\gamma}$ allows us to achieve $\hat{\mathbf{A}}$ with a minimal density error. However, under these conditions, the privacy level also depends on the local graph density γ . Graphs with low density offer weak privacy guarantees, while dense graphs provide strong protection. It is important to note that for graphs where $\gamma > 1/2$, we can opt to aggregate their complement graphs instead.

To reduce the strong dependency of ε_2 on $\hat{\gamma}$, we have enhanced our aggregation method with Algorithm 2. This algorithm details the curator's actions in the second round. Since the other steps are identical to those in Algorithm 1, they are omitted for simplicity. The key modification in Algorithm 2 involves the introduction of a Bernoulli random variable b , which determines whether to use **AND** or **OR**. The curator aggregates $\hat{\mathbf{N}}_i[j]$ and $\hat{\mathbf{N}}_j[i]$ using **AND** with probability α , and **OR** with probability $1 - \alpha$.

Algorithm 2 Improved Graph Aggregation (Partial)

Input: Privacy parameters ε_1 and ε_2

Output: Noisy graph $\hat{\mathbf{A}}$

```

1:  $\alpha = \frac{2\hat{\gamma} + p - 2}{2p - 2}$ 
2: for  $i = 1$  to  $n$  do
3:   for  $j = i + 1$  to  $n$  do
4:      $b \leftarrow \text{Bernoulli}(\alpha)$ 
5:     if  $b = 1$  then
6:        $\hat{\mathbf{A}}[i][j] \leftarrow \hat{\mathbf{N}}_i[j] \cap \hat{\mathbf{N}}_j[i]$ 
7:     else
8:        $\hat{\mathbf{A}}[i][j] \leftarrow \hat{\mathbf{N}}_i[j] \cup \hat{\mathbf{N}}_j[i]$ 
9:     end if
10:  end for
11: end for
12: return  $\hat{\mathbf{A}}$ 

```

} by curator.

Theorem III.3. With $\alpha = \frac{2\hat{\gamma} + p - 2}{2p - 2}$ and $\varepsilon_2 \geq \ln\left(\frac{1}{2\hat{\gamma}} - 1\right)$ if $\gamma \leq \frac{1}{2}$ or $\varepsilon_2 \geq \ln\left(\frac{1}{2(1-\hat{\gamma})} - 1\right)$ if $\frac{1}{2} \leq \hat{\gamma} \leq 1$, the expectation of the density error $\mathbf{E}[\bar{\gamma} - \gamma] = O\left(\frac{1}{\varepsilon_1^2 n(n-1)^2}\right)$.

Proof: similar to Theorem III.2, we have the following facts:

$$\mathbf{P}\left(\hat{\mathbf{A}}[i][j] = 1 \mid \mathbf{N}_i[j] = 1\right) = 2\alpha p^2 - 2\alpha p + 1 - p^2, \quad (18)$$

$$\mathbf{P}\left(\hat{\mathbf{A}}[i][j] = 1 \mid \mathbf{N}_i[j] = 0\right) = 2\alpha p^2 - 2\alpha p - p^2 + 2p. \quad (19)$$

Let $\alpha = \frac{2\hat{\gamma} + p - 2}{2p - 2}$, we have

$$\bar{\gamma} = \gamma + 2\alpha p^2 - 2\alpha p - p^2 + 2p - 2\gamma p \quad (20)$$

$$= 2p(\hat{\gamma} - \gamma). \quad (21)$$

Since p should be related to γ , we assume that $p = a\hat{\gamma} + b$. Similar to the proof of Theorem III.2, we have

$$\mathbf{E}[\bar{\gamma} - \gamma] = \frac{8a}{\varepsilon_1^2 n(n-1)^2}. \quad (22)$$

With $0 \leq \alpha \leq 1$, we can obtain $p \leq 2\hat{\gamma}$ if $\hat{\gamma} \leq \frac{1}{2}$, or $p \leq 2(1 - \hat{\gamma})$ if $\frac{1}{2} \leq \hat{\gamma} \leq 1$.

Thus, for $p = \frac{1}{\exp(\varepsilon_2) - 1}$, we have

$$\varepsilon_2 \geq \ln\left(\frac{1}{2\hat{\gamma}} - 1\right), \quad (23)$$

if $\hat{\gamma} \leq \frac{1}{2}$, or

$$\varepsilon_2 \geq \ln\left(\frac{1}{2(1-\hat{\gamma})} - 1\right), \quad (24)$$

if $\frac{1}{2} \leq \hat{\gamma} \leq 1$. ■

Remark 1. Our graph aggregation approach is specifically designed for undirected graphs. If applied to a directed graph G , this approach would remove the directed nature of its edges, transforming G into an undirected graph.

C. Limitation on Distance Query

The main limitation of the above method, shared by many synthetic graph generation approaches, is the inherent noise introduced during the initial aggregation of neighbor lists. Methods such as RNL [13] and probability-based graph generation [12] introduce randomness when collecting or generating edges, which can unpredictably disrupt community structures. These structures are critical, as shortest paths between vertices often depend on them. Distance queries are highly sensitive to changes in graph structure, and this sensitivity amplifies the impact of noise originating from the graph generation process. As a result, it becomes difficult to explain or improve the expected utility of distance query results.

Instead of generating synthetic graphs for querying, some studies focus on specific tasks, such as clustering coefficient estimation [19], subgraph counting (especially for triangle) [20], perform subqueries that are then aggregated to achieve an overall estimation. Following a similar routine, aggregating the local distance subquery results from each vertex is a sensible strategy. This method helps avoid the unpredictability associated with local and global community structures in graphs.

IV. NEIGHBOR AGGREGATION APPROACH

In this section, we introduce a neighbor aggregation method designed to address distance queries while providing a level of interpretability.

A. The Intuition

Let f be a distance query. For vertices s and t , their distance $f(s, t)$ can be aggregated by subqueries

$$f(s, t) \leq f(s, u) + f(u, t), \quad (25)$$

where u is an intermediate vertex. If u is located on the shortest path between s and t , the equation is satisfied. By strategically selecting an appropriate u , we can accurately aggregate the results to determine the distance.

Note that we hold an assumption in this paper that third-party adversaries cannot track traffic to reveal relationships. Thus, we can focus on the privacy risk from the query results. And this assumption is feasible through the use of a virtual private network in real-world.

From vertex s 's perspective, communication is limited to its neighbors. If s wants to know the distance to t (where t is not a neighbor of s), the most effective method is to inquire about the distances from its neighbors to t . For any neighbor u , the relationship $f(s, t) \leq 1 + f(u, t)$ holds. Let us use $N(\cdot)$ to denote the neighbor set. Therefore, s can aggregate all answers from its neighbors $N(s)$

$$f(s, t) = 1 + \min_{u \in N(s)} \{f(u, t)\}. \quad (26)$$

Neighbors can recursively perform the process to obtain the final result. However, two potential problems complicate this approach.

- **Infinite-loop trap.** Cycles are common in graphs, resulting in distance queries being cyclically forwarded among vertices, thereby making it difficult to terminate the process.

- **Uncertainty in waiting for answers.** When a vertex receives a distance query, it must wait for responses from all its neighbors, except the one that sent the query to this vertex. This recursive process can significantly increase the time overhead, especially in deeply structured graphs. Given each vertex's limited perspective, the waiting time is unpredictable. Furthermore, if messages are lost in transit, a vertex may end up waiting indefinitely.

B. Neighbor Aggregation

Algorithm 3 Neighbor Aggregation

Input: Initial noisy distance $\{\hat{\mathbf{D}}_1^{(0)}, \dots, \hat{\mathbf{D}}_n^{(0)}\}$, distance threshold T
Output: All-pair distances $\{\hat{\mathbf{D}}_1^{(T-1)}, \dots, \hat{\mathbf{D}}_n^{(T-1)}\}$
1: **for** $k = 1$ **to** $T - 1$ **do**
2: **for** each vertex $i = 1$ **to** n **do**
3: $\hat{\mathbf{D}}_i^{(k)} \leftarrow \text{Agg} \left(\{\hat{\mathbf{D}}_1^{(k-1)}, \dots, \hat{\mathbf{D}}_n^{(k-1)}\} \right)$
4: **end for**
5: **end for**
6: **return** $\{\hat{\mathbf{D}}_1^{(T-1)}, \dots, \hat{\mathbf{D}}_n^{(T-1)}\}$

Algorithm 3 outlines the neighbor aggregation method, where the central idea is to continuously update global information by aggregating local data from neighbors, analogous to the learning mechanism in GNN [21]. The key components of this method include the initial local distance design for each vertex and the implementation of the $\text{Agg}(\cdot)$ function.

Initialize Local Distance. For each vertex in a local graph, they only know their neighbors. From their perspective, the distances to the other vertices are uncertain, except the distances to the neighbors, which are 1. Let $\mathbf{D}_u^{(0)}$ denote the initial distances from the vertex u to all other vertices. For any neighbor $j \in N(u)$, $\mathbf{D}_u^{(0)}[u] = 1$; for non-neighbors, $\mathbf{D}_u^{(0)}[u] = \infty$. And for u itself, $\mathbf{D}_u^{(0)}[u] = 0$.

Subsequently, each vertex sends the distances to neighbors for aggregation. Since our privacy-preserving target is each edge in the graph, vertices must conceal their initial distance vector to prevent privacy leakage. Without such concealment, directly exposing distances of '1' could inadvertently reveal neighbor relationships. To perturb these vectors effectively, two common methods are employed: Laplace and Randomized Response (RR).

- **Laplace.** The vector \mathbf{X} is an n -dimensional random variable, with each variable being independent and following a Laplace distribution characterized by a mean of 0 and a scale parameter of 1. We can conceal edges by

$$\hat{\mathbf{D}}_u^{(0)} \leftarrow \mathbf{D}_u^{(0)} + \frac{T-1}{\varepsilon} \cdot \mathbf{X}, \quad (27)$$

where ε is the privacy parameter, and T is the distance threshold that replaces ∞ to avoid unbounded sensitivity.

- **RR.** Let \mathcal{R} be a RR mechanism and $\text{Unif}(\cdot)$ be a uniform sampling function. We can conceal edges by giving $p = \frac{T}{e\varepsilon + T - 1}$

$$\mathcal{R}(x) = \begin{cases} x & \text{with probability } 1 - p \\ \text{Unif}([T]) & \text{with probability } p \end{cases}, \quad (28)$$

$$\hat{\mathbf{D}}_u^{(0)} \leftarrow [\mathcal{R}(\mathbf{D}_u^{(0)}[1]), \dots, \mathcal{R}(\mathbf{D}_u^{(0)}[n])]. \quad (29)$$

We will discuss the differences between the two methods later in this subsection IV-D.

Aggregation Process. At first, each vertex just knows the exact distances to their neighbors; distances to all other vertices remain unknown. To manage these unknown distances, we use T as a constraint. To update their unknown distances, all vertices can aggregate the distance vectors from their neighbors. The process can be described by the following formula.

$$\hat{\mathbf{D}}_u^{(k)}[j] \leftarrow \min \left\{ \min_{i \in N(u)} \left\{ \hat{\mathbf{D}}_i^{(k-1)}[j] \right\} + 1, \hat{\mathbf{D}}_u^{(k-1)}[j] \right\}, \quad (30)$$

where $j \in [n]nN(u)$ and $k \geq 1$ denotes the k -th aggregation.

In the k -th aggregation, each vertex u receives $|N(u)|$ messages containing distance vectors from its neighbors. These vectors are the latest knowledge that each neighbor has regarding their distances to all other vertices. Therefore, the updated distance from u to another vertex j should be calculated as the minimum of the sum of u 's distance to its neighbors and their distances to j . $\hat{\mathbf{D}}_u^{(k-1)}$ contains the results from the previous round of aggregation, ensuring that the vertex u progressively acquires the optimal global distances to all vertices throughout the aggregation process.

Intuitively, this aggregation process resembles a breadth-first search. As each vertex continuously aggregates information from its neighbors, its perspective effectively expands outward by one hop with each aggregation. By the k -th aggregation, u can obtain knowledge (initial distance vector) from its k -hop neighbors. This knowledge allows u to identify its $(k+1)$ -hop neighbors. Conversely, during the k -th aggregation, a vertex that is k -hop away from u transmits its initial distance vector to u .

If no noise is injected, any vertex u is able to find vertices at distance $k+1$ at the k -th aggregation. Thus, each distance updated at an aggregation is optimal and will not change in later aggregations. However, with the introduction of random noise, the distance may be updated in each aggregation.

Theorem IV.1. *The Algorithm 3 satisfies 2ε -LDP for each edge.*

Proof: the privacy guarantee is provided by the noisy initial distance vector $\{\hat{\mathbf{D}}_1^{(0)}, \dots, \hat{\mathbf{D}}_n^{(0)}\}$. Each edge is disclosed twice by $\hat{\mathbf{D}}_i^{(0)}[j]$ and $\hat{\mathbf{D}}_j^{(0)}[i]$ where $i, j \in [n]$, it satisfies 2ε -LDP. The later aggregation process is post-processing. Thus, this privacy guarantee holds. ■

The trade-off for T . In our setting, T is the distance threshold that can limit our sensitivity to $T-1$. Large T will introduce more rounds of aggregation, while small T will cause excessive distance cropping, leading to larger errors. Diameter is an alternative option. We can predict the diameter by following diameter bound.

Lemma IV.2. (Diameter bound [22]) *Given G , a connected graph with n vertices. For its diameter d*

$$d \leq \frac{3(n-t)}{\delta+1} + O(1), \quad (31)$$

where δ is the minimum degree and t represents the count of unique values in the degree sequence.

Thus, the diameter can be estimated by an additional round of degree collection, similar to the first round in Algorithm 1. If the privacy loss for degree collection is ε_1 and for neighbor aggregation is ε_2 , the total privacy loss is $2(\varepsilon_1 + \varepsilon_2)$. However, the bound given by Lemma IV.2 is not tight. Therefore, to facilitate experimental analysis, we empirically specify T in the experiments through six degrees of separation [23].

Remark 2. In this paper, our privacy-preserving target is any neighbor relationship for every vertex. Before the aggregation process, each vertex u has injected enough randomness into its neighbor relationships (the initial distance vector $\mathbf{D}_u^{(0)}$). Thus, each vertex cannot increase their confidence in their neighbors' privacy by receiving their neighbors' distance vectors. However, we cannot guarantee that the change in the distance vector will not expose indirect neighbor relationships (2-hop or more distant neighbors) in each aggregation. Fortunately, these indirect relationships, which are not the primary target of our privacy measures, generally hold less value than direct neighbor relationships.

C. Attack Resistance

The perturbation used in the graph aggregation approach is similar to the neighbor aggregation approach. The key difference is that graph aggregation shares neighbor lists with all neighbors rather than just with a single curator. In fact, for each vertex, the initial distance vector $\mathbf{D}_u^{(0)}$ is equal to its neighbor list (just replace the value T by 0). Hence, both approaches comply with LDP.

In analyses involving multiple parties, such as Multi-Party Computation or federated learning, the assumption of participants being "honest but curious" is practical and common. Following this assumption, we consider all vertices to behave honestly during the execution but still attempt to extract private information from the aggregation process. However, such attempts fail due to the added noise in each local distance vector $\mathbf{D}_u^{(0)}$. We now discuss scenarios involving malicious vertices:

Case 1: One Malicious Vertex. Suppose vertex m behaves maliciously with two possible goals: 1) extracting private edge information and 2) increasing errors in the aggregated results. Extracting private edge information is impossible since each vertex's neighbor relationships are concealed in the initial distance vectors $\mathbf{D}_v^{(0)}, v \in [n]$. Further aggregation does not disclose any additional neighbor relationship information. However, vertex m can negatively affect the accuracy by initializing $\mathbf{D}_m^{(0)}$ incorrectly (e.g., with all zeros or ones). This manipulation can cause other vertices to underestimate distances, thus increasing overall error.

Case 2: Multiple Malicious Vertices. The worst-case scenario involves vertex u having all neighbors except vertex

v being malicious. These malicious neighbors may try to determine if vertex v is also a neighbor of vertex u . Such collusion attacks fail because all malicious vertices only receive the same perturbed distance vector $\hat{\mathbf{D}}_u^{(0)}$ from u , adding no extra privacy loss. However, if one malicious neighbor t is also a neighbor of v , the privacy loss regarding the u - v relationship is doubled. This scenario aligns with the upper bound privacy loss indicated by Theorem IV.1. It is unnecessary to consider the scenario where all neighbors of u are malicious, as collusion would straightforwardly reveal u 's neighbor list.

In conclusion, while malicious vertices may reduce the utility of all-pairs distance measurements, they do not compromise overall privacy.

D. Utility Analysis

In Subsection IV-B we provide two perturbation methods: Laplace and RR. These two have similar utility for some common queries in the local model, such as summation or counting. However, due to the complexity of the aggregation process, it is difficult to analyze the error bounds theoretically. Thus, we propose two random variables, Y_1 and Y_2 , to support the empirical utility analysis. Section V presents numerical simulations based on the following guarantees. We use $\min\{\cdot\}_n$ to denote the minimum of n independent random variables.

Theorem IV.3. *For Laplace method, the distance $\hat{\mathbf{D}}_u^{(T-1)}[j]$, if j is not the neighbor of u , can be described by random variable $Y_1 = \min\{T, \min\{W + X\}_{n-2}\}$. Here, W is a discrete random variable that depends on the graph structure around the vertex u , and X is a Laplace random variable with mean 0 and scale $\frac{T-1}{\epsilon}$.*

Proof: Let t be the true distance from vertex u to vertex j . For k -th aggregation, u can aggregate the neighbor relationships from k -hop neighbors. Without loss of generality, we assume $t < T - 2$.

- For $k < t - 1$ or $k > t + 1$,

$$\hat{\mathbf{D}}_u^{(k)}[j] \leftarrow \min\{\hat{\mathbf{D}}_u^{(k-1)}[j], \min\{X + T + k\}_{m_k}\}, \quad (32)$$

where m_k is the number of k -hop neighbors of the vertex u .

- For $k = t - 1$ or $k = t + 1$, since there are some vertices that are neighbors of j , let $\omega = \min\{X + k + 1\}_{a_k}$,

$$\hat{\mathbf{D}}_u^{(k)}[j] \leftarrow \min\{\hat{\mathbf{D}}_u^{(k-1)}[j], \min\{X + T + k\}_{m_k - a_k}, \omega\}, \quad (33)$$

where a_k is the number of neighbors j has, for which the distance to vertex u is k .

- For $k = t$, there is an exception that the distance $\hat{\mathbf{D}}_j^{(0)}[j]$ will not propagate to u since j 's neighbors will ignore it when aggregating. At this aggregation, let $\omega = \min\{X + k + 1\}_{a_k}$,

$$\hat{\mathbf{D}}_u^{(k)}[j] \leftarrow \min\{\hat{\mathbf{D}}_u^{(k-1)}[j], \min\{X + T + k\}_{m_k - a_k - 1}, \omega\}. \quad (34)$$

For clarity, we denote a_{t-1}, a_t and a_{t+1} by a_1, a_2 and a_3 , respectively. Let W be a random variable drawn from the distribution that represents the above constants. Specifically, we consider the histogram, which captures the frequency of occurrence of all constants (such as $T + k$ or $k + 1$), as a probability distribution. This histogram is detailed in Table I. $\hat{\mathbf{D}}_u^{(T-1)}[j]$ can be represented by a random variable Y_1

$$Y_1 = \min\{T, \min\{W + X\}_{n-2}\}, \quad (35)$$

where the $n - 2$ means that u will aggregate distance vectors from all vertices except itself and j . ■

Theorem IV.4. *For RR method, the distance $\hat{\mathbf{D}}_u^{(T-1)}[j]$, is the value of $Y_2 = \min\{T, \min\{W_1 + X_1\}_m, \min\{W_2 + X_2\}_a\}$ if j is not the neighbor of u . Here, $a + m = n - 2$, W_1 and W_2 are two discrete random variables that depend on the graph structure around the vertex u , and X_1, X_2*

$$P(X_1 = x) = \begin{cases} 1 - \frac{T-2}{T-1}p & \text{if } x = 0, \\ \frac{T-2}{T-1}p & \text{if } x = \text{Unif}([1 - T, -1]), \end{cases} \quad (36)$$

$$P(X_2 = x) = \begin{cases} 1 - \frac{T-2}{T-1}p & \text{if } x = 0, \\ \frac{T-2}{T-1}p & \text{if } x = \text{Unif}([1, T - 1]). \end{cases} \quad (37)$$

Proof: let us reformulate RR using random variables. For $\mathbf{D}_u^{(0)}[j] = T$, $\mathcal{R}(T)$ has the same distribution as $X_1 + T$. And for $\mathbf{D}_u^{(0)}[j] = 1$, $\mathcal{R}(1)$ has the same distribution as $X_2 + 1$. The distributions of X_1 and X_2 are listed above. Most of the proof is similar to Theorem IV.3, but using X_1 and X_2 replace X .

- For $k < t - 1$ or $k > t + 1$,

$$\hat{\mathbf{D}}_u^{(k)}[j] \leftarrow \min\{\hat{\mathbf{D}}_u^{(k-1)}[j], \min\{X_1 + T + k\}_{m_k}\}. \quad (38)$$

- For $k = t - 1$ or $k = t + 1$, $\hat{\mathbf{D}}_u^{(k)}[j]$ is the minimum of $\hat{\mathbf{D}}_u^{(k-1)}[j]$ and

$$\min\{\min\{X_1 + T + k\}_{m_k - a_k}, \min\{X_2 + k + 1\}_{a_k}\}. \quad (39)$$

- For $k = t$, $\hat{\mathbf{D}}_u^{(k)}[j]$ is the minimum of $\hat{\mathbf{D}}_u^{(k-1)}[j]$ and

$$\min\{\min\{X_1 + T + k\}_{m_k - a_k - 1}, \min\{X_2 + k + 1\}_{a_k}\}. \quad (40)$$

For clarity, we denote a_{t-1}, a_t and a_{t+1} by a_1, a_2 and a_3 , respectively. Let $a = \sum_{i=1}^3 a_i$ and $m = n - 2 - a$, we have

$$Y_2 = \min\{T, \min\{W_1 + X_1\}_m, \min\{W_2 + X_2\}_a\}, \quad (41)$$

where $W = W_1 \cup W_2$. ■

V. EXPERIMENTS

In this section, we evaluate the utility of the Laplace and RR methods through numerical simulations. Subsequently, we compare the performance of our two approaches with LDPGen and RNL [13] in three real-world datasets.

TABLE I
THE HISTOGRAM OF ALL CONSTANTS.

Constant	W	W_1	W_2
t	a_1	0	a_1
$t+1$	a_2	0	a_2
$t+2$	a_3	0	a_3
$T+1$	m_1	m_1	0
$T+2$	m_2	m_2	0
...			
$T+t-1$	$m_{t-1}-a_1$	$m_{t-1}-a_1$	0
$T+t$	m_t-a_2-1	m_t-a_2-1	0
$T+t+1$	$m_{t+1}-a_3$	$m_{t+1}-a_3$	0
$T+t+2$	m_{t+2}	m_{t+2}	0
...			
$T+T-1$	m_{T-1}	m_{T-1}	0

A. Datasets and Metric

We utilize three real-world datasets of varying sizes, described as follows:

- EIES [24]. This dataset represents a personal relationship graph among researchers. For our analysis, we ignore the weight information and utilize its complement graph, which comprises 34 vertices and 87 edges.
- Twitter [25]. This is a directed Twitter interaction graph about the United States Congress. It contains 475 vertices and 10,222 edges.
- Facebook [26]. Originally an undirected social graph from the Facebook app, this dataset includes 10 networks. We use the largest network that contains 1,034 vertices and 26,749 edges.

To evaluate the performance of our approaches on distance queries, we employ the Relative Absolute Mean Error (RAME) and the Mean Relative Error (MRE). RMAE captures the errors across all elements and is sensitive to fine-grained error distributions, whereas MRE is insensitive to individual element errors and reflects only the overall trend.

Definition V.1. (RAME and MRE) Let $d_{u,v}$ be the distance from the vertex u to v and $d'_{u,v}$ be the noisy distance. We have RAME η_1 and MRE η_2

$$\eta_1 = \frac{1}{n^2 - n} \sum_{\substack{u,v \in G \\ u \neq v}} \frac{|d'_{u,v} - d_{u,v}|}{|d_{u,v}|}, \eta_2 = \frac{|\bar{d}' - \bar{d}|}{|\bar{d}|}, \quad (42)$$

where \bar{d} and \bar{d}' are the mean of all distances and all noisy distances, respectively.

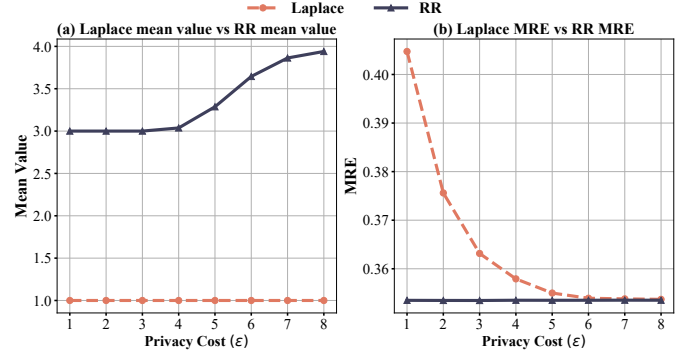


Fig. 2. The simulation results for Laplace and RR in privacy cost $[1, 8]$.

B. Simulation Results

We conduct numerical simulations based on Theorems IV.3 and IV.4 to compare the utility of the Laplace and RR mechanisms. Given that the distributions of X , X_1 , and X_2 are already defined, the primary focus shifts to determining the distributions of W , W_1 , and W_2 . As depicted in Table I, these variables form histograms where $W = W_1 \cup W_2$. We approximate the histogram frequencies as probabilities and, for simplification, assume that they follow a uniform distribution.

We empirically set $T = 6$ for our experiments, inspired by the concept of six degrees of separation. Assuming, without loss of generality, that the true distance between vertices u and v is 4. To reduce randomness in our results, we set $n = 10,000$ and repeat the experiments 1,000 times. Finally, we set $\varepsilon \in [1, 8]$. This range of ε does not represent a serious privacy risk but is chosen to better illustrate trends.

In Fig. 2(a), as ε increases, the mean value of the RR method approaches 4, reflecting an increase in accuracy. In contrast, the mean value of the Laplace method remains consistently close to 1. This consistency occurs because the sample values from the Laplace method are predominantly negative, leading us to adjust for utility by setting any sample value less than 1 equal to 1. Furthermore, Fig. 2(b) illustrates the MRE of both the Laplace and RR methods when applied to the Facebook dataset. It is evident that the RR method consistently yields smaller errors compared to the Laplace method as ε increases.

The underperformance of the Laplace method can be ascribed to its unbounded sample space. The min causes the expectation to be biased in a negative direction. Let $Y = \min\{X\}_n$, X be an independent Laplace random variable with mean 0 and variance $2b^2$. Using the properties of cumulative distribution functions and order statistics, we can get the expectation of Y

$$\mathbf{E}[Y] = b \ln \left(\frac{1}{2} - \frac{1}{n+1} \right), \quad (43)$$

if $n > 1$.

We observe that $\mathbf{E}[Y] < 0$, a phenomenon that is likely to persist for Y_1 as well, despite the bias introduced by W that tends to shift its expectation towards positive values. Consequently, we will employ the RR method in our subsequent experiments.

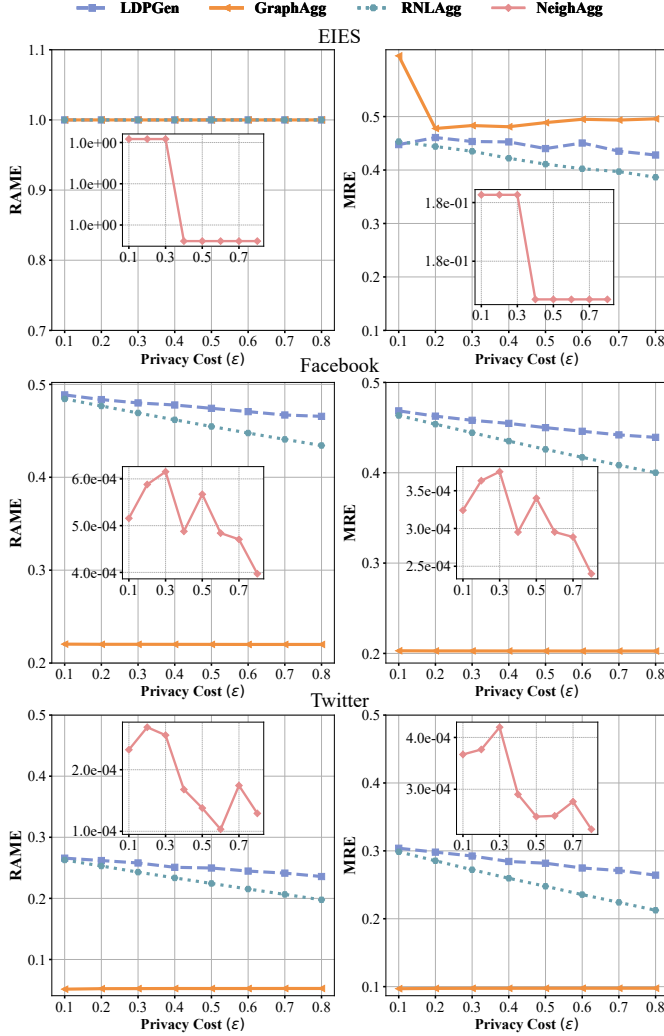


Fig. 3. The RAME and MRE results for four approaches: LDPGen, GraphAgg, RNLA and NeighAgg under three real-world datasets: EIES, Facebook and Twitter when ϵ changes from 0.1 to 0.8. The inset plot in each plot shows the trend of NeighAgg for clear visibility.

C. Performance Results

We perform our experiments on three real-world datasets using two our approaches: graph aggregation (GraphAgg) and neighbor aggregation (NeighAgg). Due to the lack of similar methods for distance queries, we select the classical LDPGen and RNLA [13] as our baselines for comparative analysis. The initial parameters for LDPGen are set according to its recommended settings: $K_0 = 2$, $\epsilon_1 = \epsilon/2$, and $\epsilon_2 = \epsilon/2$.

For GraphAgg, we use Algorithm 1. The privacy budget for the first round is ϵ . And for NeighAgg, with a fixed $T = 6$, $\epsilon/2$ directly represents the privacy budget utilized to initialize each distance vector. Since LDPGen, RNLA, and GraphAgg can generate synthetic graphs where certain vertex pairs become unreachable, we uniformly assign $T = 6$ to represent the distance between such unreachable vertices.

Fig. 3 demonstrates the comparative performance of our two methodologies against established baselines. Across all graphs, there is a noticeable decreasing trend in errors as ϵ increases. Notably, except for EIES, all methods adhere to the following

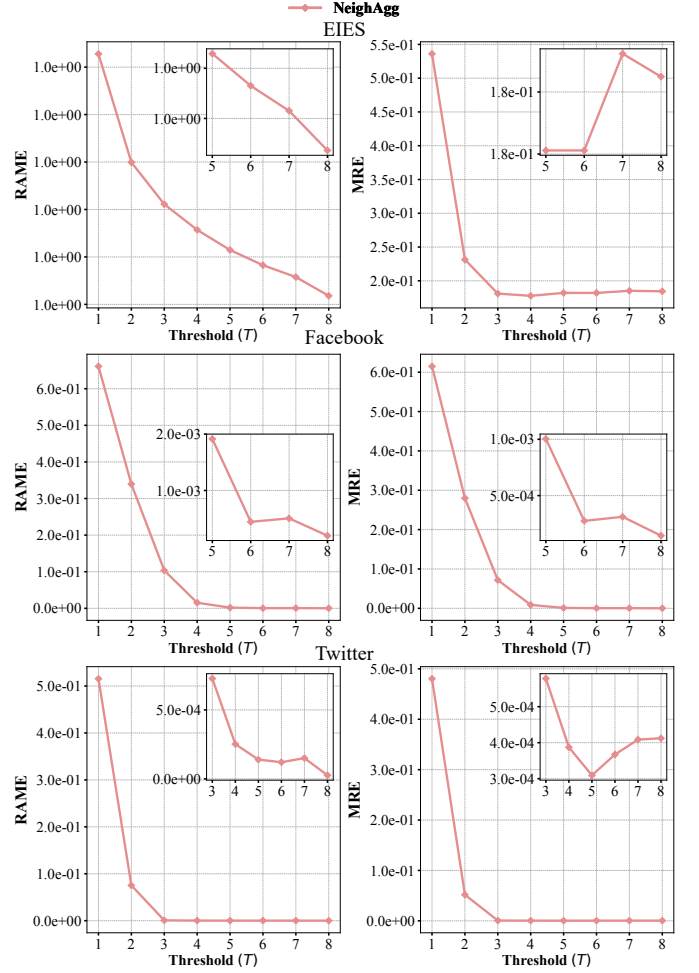


Fig. 4. The RAME and MRE results for NeighAgg under three real-world datasets: EIES, Facebook and Twitter when fixing $\epsilon = 0.1$ and T changes from 1 to 8. The inset plot in each plot shows local trend of NeighAgg for clear visibility.

pattern: $\text{NeighAgg} < \text{GraphAgg} < \text{RNLA} < \text{LDPGen}$, with NeighAgg exhibiting significantly lower errors, approximately 10^{-4} . It is important to mention that for GraphAgg, errors consistently reduce with higher values of ϵ . However, due to space constraints, we are unable to present these details as comprehensively as for NeighAgg. In the case of EIES, its atypical variations can be attributed to having fewer vertices and a higher susceptibility to noise interference. Nonetheless, NeighAgg maintains the lowest error rate among the compared methods in EIES.

Fig. 4 illustrates the impact of varying T values on the error. Across all plots, the error decreases as T increases. However, a larger T also results in higher computational costs, as more aggregation steps are required in NeighAgg, leading to a linear increase in runtime. Therefore, we identify $T = 6$ as the optimal choice, striking a balance between low error and acceptable aggregation overhead.

VI. RELATED WORK

A. Synthetic Graph Generation

Qin et al. [13] proposed a synthetic social graph generation method under LDP by combining the Randomized Neighbor List (RNL) and Degree-Based Graph Generation (DGG). They achieved a superior partition of the vertices using the k -means several times. Ju et al. [27] explored graph correlations and developed a correlation-based method for synthetic graph generation to address privacy leaks. Wei et al. [28] achieved preservation for graph statistics and attribute distributions by finely tuning the noise added. Focusing on the dynamic update problem of synthetic graphs, Hou et al. [29] proposed a new graph publishing mechanism that reduces noise dependency on graph scale through a privacy-preference-specifying mechanism, facilitating the release of dynamic graphs.

B. Distance Query

Current approaches that support differentially private distance queries are mostly based on the weight private graph assumption. This assumption can limit the global sensitivity to 1 (or a unit). Using the graph assumption, Sealfon [7] pioneered the release of all-pair distances, with the global sensitivity considered negligible. This work showed that the upper bound of the additional error is $O(n \log n / \epsilon)$. Fan et al. [8] devised two strategies specifically for trees and grid graphs that limit errors to $O(\log^{1.5} n)$ and $\tilde{O}(n^{3/4})$, respectively. Furthermore, Fan et al. [9] developed a method for privacy-sensitive distance queries via synthetic graphs, resulting in an error of $\tilde{O}(n^{1/2})$. With the same assumption, Chen et al. [10] provided an algorithm for distance release, incurring an error $\tilde{O}(n^{2/3} / \epsilon)$ and noted that the minimum error should be $\Omega(n^{1/6})$. They further reduced the error for bounded weights to roughly $n^{(\sqrt{17}-3)/2+o(1)} / \epsilon$. Leaving that assumption, Cai et al. [30] examined scenarios involving more significantly varied neighboring weights, specifically where weights range from $[a, b]$ and differences can reach $b - a$. This variation introduces substantial global sensitivity. Consequently, their research focuses on mitigating the impact of noise on the shortest paths and distances.

VII. CONCLUSION

In this paper, we propose two novel approaches for distance query under LDP. The first approach, graph aggregation, effectively addresses issues related to graph density by generating synthetic graphs through the innovative use of bitwise operations on RNL. Our second approach, neighbor aggregation, addresses the low utility typically observed in synthetic graph methods by enabling the continuous aggregation of distance vectors from each vertex's neighbors. Both theoretical analysis and empirical evaluation confirm the robustness and effectiveness of the proposed approaches.

ACKNOWLEDGMENTS

This paper benefited from the use of OpenAI's ChatGPT in polishing the language and improving clarity. No part of the technical contributions or experimental results was generated by AI models.

REFERENCES

- [1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "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, 2006, pp. 265–284.
- [2] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting Telemetry Data Privately," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 3571–3580.
- [3] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in apple's implementation of differential privacy on macos 10.12," *arXiv preprint arXiv:1709.02753*, 2017.
- [4] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor - Randomized Aggregatable Privacy-Preserving Ordinal Response," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014.
- [5] D. Nguyen, M. Halappanavar, V. Srinivasan, and A. Vullikanti, "Faster approximate subgraph counts with privacy," *Advances in Neural Information Processing Systems*, vol. 36, pp. 70402–70432, 2023.
- [6] F. T. Brito, V. A. Farias, C. Flynn, S. Majumdar, J. C. Machado, and D. Srivastava, "Global and local differentially private release of count-weighted graphs," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–25, 2023.
- [7] A. Sealfon, "Shortest Paths and Distances with Differential Privacy," in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM, 2016.
- [8] C. Fan and P. Li, "Distances Release with Differential Privacy in Tree and Grid Graph," in *International Symposium on Information Theory (ISIT)*, 2022, pp. 2190–2195.
- [9] C. Fan, P. Li, and X. Li, "Private Graph All-Pairwise-Shortest-Path Distance Release with Improved Error Rate," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
- [10] J. Y. Chen, B. Ghazi, R. Kumar, P. Manurangsi, S. Narayanan, J. Nelson, and Y. Xu, "Differentially Private All-Pairs Shortest Path Distances: Improved Algorithms and Lower Bounds," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023, pp. 5040–5067.
- [11] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 171–180.
- [12] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of erdős-rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, 2012.
- [13] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating Synthetic Decentralized Social Graphs with Local Differential Privacy," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017.
- [14] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate Estimation of the Degree Distribution of Private Networks," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009.
- [15] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014. [Online]. Available: <http://dx.doi.org/10.1561/04000000042>
- [16] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.
- [17] J. Imola, T. Murakami, and K. Chaudhuri, "Locally Differentially Private Analysis of Graph Statistics," in *USENIX Security Symposium*, 2021, pp. 983–1000.
- [18] T. Eden, Q. C. Liu, S. Raskhodnikova, and A. Smith, "Triangle counting with local edge differential privacy," in *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.
- [19] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Lf-GDPR: A Framework for Estimating Graph Metrics With Local Differential Privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4905–4920, 2022.
- [20] J. Imola, T. Murakami, and K. Chaudhuri, "Communication-Efficient Triangle Counting under Local Differential Privacy," in *USENIX Security Symposium*, 2022, pp. 537–554.
- [21] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "{GAP}: Differentially private graph neural networks with aggregation perturbation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3223–3240.
- [22] S. Mukwembu, "A note on diameter and the degree sequence of a graph," *Applied mathematics letters*, vol. 25, no. 2, pp. 175–178, 2012.

- [23] E. Elmacioglu and D. Lee, "On six degrees of separation in dblp-db and more," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 33–40, 2005.
- [24] S. C. Freeman and L. C. Freeman, *The networkers network: A study of the impact of a new communications medium on sociometric structure*. School of Social Sciences University of Calif., 1979.
- [25] C. G. Fink, K. Fullin, G. Gutierrez, N. Omodt, S. Zinnecker, G. Sprint, and S. McCulloch, "A centrality measure for quantifying spread on weighted, directed networks," *Physica A*, 2023.
- [26] J. Leskovec and J. Mcauley, "Learning to discover social circles in ego networks," *Advances in neural information processing systems*, vol. 25, 2012.
- [27] X. Ju, X. Zhang, and W. K. Cheung, "Generating Synthetic Graphs for Large Sensitive and Correlated Social Networks," in *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2019.
- [28] C. Wei, S. Ji, C. Liu, W. Chen, and T. Wang, "Asgldp: Collecting and Generating Decentralized Attributed Graphs With Local Differential Privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3239–3254, 2020.
- [29] L. Hou, W. Ni, S. Zhang, N. Fu, and D. Zhang, "Ppdu: dynamic graph publication with local differential privacy," *Knowledge and Information Systems*, vol. 65, no. 7, pp. 2965–2989, 2023.
- [30] B. Cai, W. Sheng, J. Chen, C. Hu, and J. Yu, "Shortest Paths Publishing With Differential Privacy," *IEEE Transactions on Sustainable Computing*, vol. 9, no. 2, pp. 209–221, 2024.