Comput Syst Sci & Eng (2018) 2: 61–69
© 2018 CRL Publishing Ltd

International Journal of
Computer Systems
Science & Engineering

An iteration-based differentially private social network data release

Tianqing Zhu[1], **Mengmeng Yang**[1], **Ping Xiong**[2], **Yang Xiang**[1] and **Wanlei Zhou**[1]

[1] *School of Information Techonolgy, Deakin University, Burwood, Australia*
E-mail: t.zhu@deakin.edu.au, ymengm@deakin.edu.au, yang.xiang@deakin.edu.au, wanlei.zhou@deakin.edu.au
[2] *School of Information and Security Engineering, Zhongnan University of Economics and Law, Wuhan, China*
E-mail: pingxiong@znufe.edu.cn

Online social networks provide an unprecedented opportunity for researchers to analysis various social phenomena. These network data is normally represented as graphs, which contain many sensitive individual information. Publish these graph data will violate users' privacy. Differential privacy is one of the most influential privacy models that provides a rigorous privacy guarantee for data release. However, existing works on graph data publishing cannot provide accurate results when releasing a large number of queries. In this paper, we propose a graph update method transferring the query release problem to an iteration process, in which a large set of queries are used as update criteria. Compared with existing works, the proposed method enhances the accuracy of query results. The extensive experiment proves that the proposed solution outperforms two state-of-the-art methods, the Laplace method and the correlated method, in terms of *Mean Absolute Value*. It means our methods can retain more utility of the queries while preserving the privacy

1. INTRODUCTION

With the significant growth of *Online Social Networks* (OSNs), the increasing volumes of data collected in those OSNs have become a rich source of insight into fundamental societal phenomena, such as epidemiology, information dissemination, marketing, etc. Much of this OSN data is in the form of graphs, which represent information such as the relationships between individuals. Releasing those graph data has enormous potential social benefits. However, the graph data infer sensitive information about a particular individual [1] has raised concern among social network participants.

To deal with the problem, lots of privacy models and related algorithms have been proposed to preserve the privacy of graph data. *Differential privacy* is the most prevalent one due to its rigorous privacy guarantee. If the differential privacy mechanism is adopted in graph data, the research problem is then to design efficient algorithms to release statistics about the graph while satisfying the definition of differential privacy. Two concepts for graph have been proposed: *Node Differential Privacy* and *Edge Differential Privacy*. The former protects the node of the graph and the latter protects the edge in the graph.

Previous works have successfully achieved both node differential privacy and edge differential privacy when the number of queries is limited. For example, Hay et al. [9] implemented node differential privacy, and pointed out the difficulties to achieve the node differential privacy. Paper [17, 18] proposed to publish graph dataset using a *dK-graph* model. Chen et al. [4] considered the correlation between nodes and proposed a correlated release method for sparse graphes. However, these works suffer from a serious problem: when the number of queries is increasing, a large volume of noise will be introduced. In real world, we have to release large number of queries for data mining, recommendation or other purposes. The difficulty lies on the problem is that the privacy budget should be divided into tiny pieces when the query set are large. Large amount of noise will be introduce to the published query answers in this scenario.

This paper focuses on releasing a large set of queries for graph data. Given a set of queries, we apply an iteration method to generate a synthetic graph to answer these queries accurately. We can consider the iteration process as a training procedure, in which queries are training samples and the synthetic graph is an output learning model. Finally, we will adopt the synthetic graph to answer this set of queries. As the training process consumes less privacy budget than the state-of-the-art methods, the total noise will be diminished. In these procedures, the major research issue becomes how to design the iteration process to generate the synthetic graph.

Our major contribution of this paper is to transfer the query release problem to an iteration based training process. Specifically, we propose an iteration method, called Graph Update to generate a synthetic graph, which can answer a large amount of queries accurately. Compared with state-of-the-art methods, the Laplace method and the correlated method, it can decrease the total amount of noise significantly.

The rest of the paper is organized as follows: We present the preliminaries in Section 2. Section 3 discusses the *Graph Update* method and the experimental result is presented in Section 5, which is followed by the conclusion in Section 6.

2. PRELIMINARIES

2.1 Notation

We consider a finite *data universe* $\mathcal{X}$ and a *dataset* D is an unordered set of n records from $\mathcal{X}$. Let r be a *record* with d attributes sampled from $\mathcal{X}$, Two datasets D and D^* are *neighboring* datasets if they differ in only one record. A *query* f is a function that maps dataset D to an abstract range $\mathbb{R}$: $f : D \to \mathbb{R}$. A group of queries is denoted as $F = \{f_1, ..., f_m\}$, and $F(D)$ denotes $\{f_1(D), ..., f_m(D)\}$. We use symbol m to denote the number of queries in F.

The maximal difference on the results of query f is defined as the *sensitivity* s, which determines how much perturbation is required for the private-preserving answer. To achieve the target, differential privacy provides a mechanism $\mathcal{M}$, which is a randomized algorithm that accesses the database. The randomized output is denoted by a circumflex over the notation. For example, $\widehat{f}(D)$ denotes the randomized answer of querying f on D.

2.2 Graph Notations

We model social network as a simple undirected graph $G\langle V, E\rangle$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices (or Node) representing individuals in the social network and $E \subseteq \{(u, v)|u, v \in V\}$ is a set of edges representing relationships between individuals. Fig. 1 shows an example of a social network graph. The nodes are represented by circles and connected with each other by edges represented by lines. The degree of a node refers to the number of its neighbourhoods. Formally, we define degree as follows,
Neighbourhood:

$$N(v) = \{u|(u, v) \in E, u \neq v\} \tag{1}$$

Degree:

$$D(v) = |N(v)| \tag{2}$$

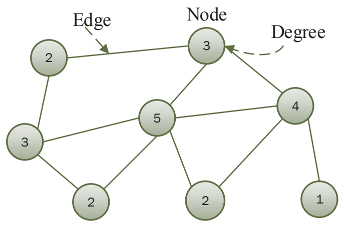


Figure 1: Graph Data

2.3 Differential Privacy

The target of *differential privacy* is to mask the difference in the answer of query f between the *neighboring datasets* [7]. In ϵ-differential privacy, parameter ϵ is defined as the *privacy budget* [7], which controls the privacy guarantee level of mechanism $\mathcal{M}$. A smaller ϵ represents a stronger privacy. The formal definition of differential privacy is presented as follows:

Definition 1 (ϵ-Differential Privacy) *A randomized algorithm $\mathcal{M}$ gives ϵ-differential privacy for any pair of* neighboring datasets *D and D^*, and for every set of outcomes , $\mathcal{M}$ satisfies:*

$$Pr[\mathcal{M}(D) \in \,] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D^*) \in \,] \tag{3}$$

Sensitivity is a parameter determining how much perturbation is required in the mechanism with a given privacy level.

Definition 2 (Sensitivity) *[7] For a query $f : D \to \mathbb{R}$, the sensitivity of f is defined as*

$$s = \max_{D,D'} ||f(D) - f(D^*)||_1 \tag{4}$$

The *Laplace* mechanism adds *Laplace* noise to the true answer. The mechanism is defined as follows:

Definition 3 (*Laplace* mechanism) *[7] Given a function $f : D \to \mathbb{R}$ over a dataset D, the Eq. 3 provides the ϵ-differential privacy.*

$$\widehat{f}(D) = f(D) + Laplace(\frac{s}{\epsilon}) \tag{5}$$

In graph data, we use G to represent D.

2.4 Related Work

2.4.1 Node Differential Privacy

Node differential privacy ensures the privacy of a query over two neighbouring graphs where two neighbouring graphs can differ up to all edges connected to one node.

Hay et al. [9] first proposed the notion of node differential privacy and pointed out the difficulties to achieve it, even it can provide strong privacy guarantee. Hay et al. [10] showed that the result of query was highly inaccurate for analysing graph due to the large noise.

Recently, there are few works [5, 11] contribute to reduce sensitivity and return accurate answers under node differential privacy. Although this is a good progress, these algorithms still hard to be applied in real world, the most prevalent algorithms are focusing on the *Edge differential privacy*.

2.4.2 Edge Differential Privacy

Edge differential privacy means adding or deleting a single edge between two nodes in the graph makes negligible difference to the result of the query. The first differential private computation over graph dataset with edge differential privacy appeared in paper [16], in which Nissim et al. tried to count the number of triangles in the graph. They provided the concept smooth sensitivity to calibrate the noise to a more local variant of sensitivity.

A work presented in [15] shared the differential private graph topology based on Stochastic Kronecker graph generation model by perturbing model parameters. While the Stochastic Konecker generation model cannot capture the properties of graph accurately due to simple generation process.

Paper [17, 18] published graph dataset using a *dK-graph* model. They applied *dK-series* as query function and added controllable noise based on sensitivity parameter. Wang et al. [18] proved that privacy *dK-graph* model can more precisely capture most of the graph properties and achieve better utility preservation. In order to reduce the noise added to the *dK-series*, Sala et al.[17] provided an algorithm partitioning the data of *dK-series* into clusters with similar degree. It significantly reduced the sensitivity for each sub-series. But it used local sensitivity which can reveal information that cannot achieve strict privacy preserving [16].

A different approach was proposed in paper [19]. Inferring the network's structure via connection probabilities. They encoded the structure information of the social network by the connection probabilities between nodes instead of the presence or absence of the edges. Which reduced the impact of a single edge. Another work in paper[2] provided a reasonable hypothesis about the structure of the dataset to restrict the sensitivity of the query. However, those methods would generate a large dense matrix which are computationally infeasible for large social network.

The most similar work to ours is from Chen et al. [4], which shared the same target of this paper: releasing a synthetic graph to publish a large set of queries. However, they focused on the correlated queries on the sparse graph. When dealing with large amount of queries, the performance is not optimal.

3. GRAPH UPDATE METHOD

3.1 Overview of Graph Update

The release method is an iteration-based algorithm, which is a prevalent release scenario of many applications [8]. Our proposed method is called *Graph Update* method as the key idea is to update a synthetic graph until all queries have been answered.

For a social network graph G and a set of queries $F = \{f_1, ..., f_m\}$. Our goal is to release a set of query results $\widehat{F}$ and a synthetic graph $\widehat{G}$ to the public. Our general idea is to define an initial graph $\widehat{G}_0$ and update it to $\widehat{G}_{m-1}$ in m round according to m queries in F. Release answers $\widehat{F}$ and the synthetic graph $\widehat{G}$ are generated during the iteration. During the process, four different types of query answer involve in the iteration:

- *True answer* a_t: this is the real answer that a graph response to a query. We cannot release it directly as it will arise privacy concern. The true answer is normally used as the baseline to measure the utility loss of a privacy-preserving algorithm. In this paper, we use $a_t = f(G)$ to represent the true answer for a single query f, and $A_t = F(G) = \{a_{t1}, ..., a_{tm}\}$ to represent an answer set for a query set F.

- *Noise answer* a_n: when we add *Laplace* noise to a true answer, the result will be the noise answer. Traditional *Laplace* method will release the noise answer directly. However, as we mentioned in Section 1, it will introduce large amount of noise to the release result. We use $a_n = \widehat{f}(G) = f(G) + Lap(s/\epsilon)$ to represent a single query answer and $A_n = \widehat{F}(G) = \{a_{n1}, ..., a_{nm}\}$ to represent an answer set.

- *Synthetic answer* a_s: this is the answer generated by a synthetic graph $\widehat{G}$. We use $a_s = f(\widehat{G})$ to represent a single query and $A_s = F(\widehat{G}) = \{a_{s1}, ..., a_{sm}\}$ to represent an answer set.

- *Release answer* a_r: this is the answer finally released after the iteration. In Graph Update method, the release answer set will consist of noise answers and synthetic answers. We apply $a_r = \widehat{f}$ and $A_r = \widehat{F} = \{a_{r1}, ..., a_{rm}\}$ to represent the single answer of a query and the answer set, respectively.

These four different query answers control the graph update process. The overview of method is presented in Figure 2. On the left side of the figure, the query set F performs on the G to obtain a true answer set A_t. *Laplace* noise is then added to A_t to get a set of noise answer $A_s = \{a_{s1}, ...a_{sm}\}$. Each noise answer a_{si} helps to update the initial $\widehat{G}_0$ and produce a release answer a_{ri}. The method eventually outputs $A_r = \{a_{r1}, ..., a_{rm}\}$ and the $\widehat{G}_m$ as final results.

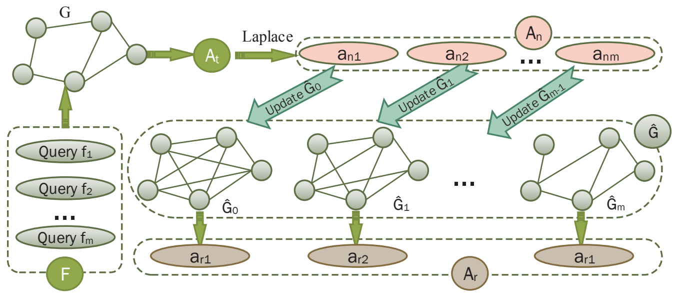


Figure 2: Overview of *Graph Update* Method

Comparing with the traditional *Laplace* method, the proposed *Graph Update* method adds less noise. As some queries are answered by the synthetic graph, these query answers will not consume any privacy budget. Moreover, the synthetic graph can be applied to predict new queries without any privacy budget. Eventually, the *Graph Update* method can outperform the tractional *Laplace* method.

3.2 Graph Update Method

The *Graph Update* method works in three steps:

- *initial the synthetic graph*: As we only preserve the edge privacy, we assume that the number and the labels of nodes

Algorithm 1 Graph Update method

```
Require: $G$, $F = \{f_1, ..., f_m\}$, $\epsilon$, $T$
Ensure: $A_r = \{a_{r1}, ..., a_{rm}\}$.
    1. $\epsilon' = \epsilon/m$
    2. initial graph $\widehat{G}_0$;
    for each query $f_i \in F$ do
        3. Compute true answer $a_{ti}$;
        4. Add Laplace Noise to true answer $a_{ni} = \widehat{f_i} = f_i(G) + Lap(S/\epsilon')$;
        5. Compute synthetic answer $a_{si} = f_i(\widehat{G})$;
        6. $d_i = a_{ni} - a_{si}$;
        if $|d_i| > T$ then
            7. $a_{ri} = a_{ni}$;
            8. update the $\widehat{G}_{i-1}$ to $\widehat{G}_i$;
        else
            9. $a_{ri} = a_{si}$;
            10. $\widehat{G}_m = \widehat{G}_{m-1}$
        end if
    end for
    11. Make all degrees in $G$ round numbers.
    12. Output $A_r = \{a_{r1}, ..., a_{rm}\}$, and $\widehat{G}$;
```

are fixed. The synthetic graph is initialed as a fully connected graph with fixed nodes.

- *update the synthetic graph*: the initial graph will be updated according to result of each query in F, until all queries in F have been used.
- *release query answers and synthetic graph*: Two types of answers, noise answers and synthetic answers that have potential to be released. Synthetic graph is also released to the public.

Algorithm 1 is a detailed description of the *Graph Update* method. In step 1, the privacy budget ϵ is divided by m and will be arranged to each query in the set. Step 2 initializes the graph to $\widehat{G}_0$ as a full connected one. Then for each query f_i in the query set F, the algorithm computes the true answer $f_i(G)$ at Step 3. After that, the noise answer and the synthetic answer of f_i are computed at Step 4 and 5, respectively. Step 6 measures the distance between the true answer and the synthetic answer. If the distance is larger than a threshold T, the Step 7 will release the noisy answer. Otherwise, the synthetic graph will be updated by an *Updated Function* in Step 8 and Step 9 will release the synthetic answer. This means the synthetic graph is applicable for answering question, so in Step 10, we put the current synthetic graph to the next round. This process is iterated until all queries in F are preceded. Finally, As the number of edges should be a integer, we round the number of degrees in Step 11. the algorithm generates A_r and $\widehat{G}$ as the output in Step 12.

The parameter T is a threshold controlling the distance between A_n and A_s. A larger T means less update of the graph and most of the answer in A_r are synthetic answers. It leads to less privacy budget consuming, however, when the synthetic graph is far away from the original graph, the performance may not optimal. A smaller T means the algorithm has more updates of the graph and most of the answer in A_r are noise answers.

Algorithm 2 Update Function

```
Require: $\widehat{G}$, $f$, $d$, $\theta$, $(0 < \theta < 1)$
Ensure: $\widehat{G}'$.
    1. Identify related nodes $V_f$ that $f$ involved;
    if $d > 0$ then
        2. $D(V_f) = (1 + \theta) * D(V_f)$;
    else
        3. $D(V_f) = \theta * D(V_f)$;
    end if
    4. $\widehat{G}' = G \cup D(V_f)$.
    5. Output $\widehat{G}'$.
```

More privacy budgets will be consumes in this configuration. Consequently, the choice of T will have impact on different scenarios. We will confirm the value of T in the experiment in Section 5

3.3 Update Function

Step 8 in Algorithm 1 involves with an *Update Function*, which updates the synthetic graph $\widehat{G}$ to graph $\widehat{G}'$ according to query answers. Specifically, *Update Function* is controlled by the distance d between the a_n and a_s of f. If a_n is smaller than a_s, it means that the synthetic graph has more edges than the original graph in the related nodes. *Update Function* has to delete some edges between the related nodes. Otherwise, *Update Function* will add some edges in the synthetic graph.

These related nodes is defined in the follow definition 4:

Definition 4 (Related Node) *For a query f and a graph G,* related nodes V_f *are all nodes that response to the query f, we use set $D(V_f)$ to denote degrees of those nodes.*

The number of edges for a node should be a integer. However, to adjust degree of those related nodes, we arrange weight θ $(0 \leq \theta \leq 1)$ for each edge. After the updating, these weights will be rounded to represent node edges.

Algorithm 2 illustrates the detail of *Update Function*. In the first step, the function identifies related nodes. If $d > 0$, which means the synthetic graph has less edges than the original one, the function will enhance the θ in Step 2. If $d \leq 0$, which means the synthetic graph has too many edges, the function will diminish those edges by θ in Step 3. Step 4 merges the edges to the original graph. Step 5 outputs the $\widehat{G}'$.

4. PRIVACY AND UTILITY ANALYSIS

4.1 Privacy Analysis

This section presents a comparison on the privacy between the tractional *Laplace* method and *Graph Update*. The *sequential composition* [14] of the privacy budget is applied, which is shown in Lemma 1 The *sequential composition* accumulates privacy budget ϵ of each step when a series of private steps is performed *sequentially* on a dataset.

Lemma 1 Sequential Composition*: Suppose a method $\mathcal{M} = \{\mathcal{M}_1, ...\mathcal{M}_m\}$ has m steps, and each step $\mathcal{M}_i$ provides ϵ privacy guarantee, the sequence of $\mathcal{M}$ will provide* $(m * \epsilon)$-differential privacy.

For traditional *Laplace* method, when answering F with m queries, ϵ will be divided into m pieces and arranged to each query $f_i \in F$. Specifically, we have $\epsilon' = \epsilon/m$ and for each query, the noise answer will be $a_{ni} = f_i + Lap(s/\epsilon')$. According to the *sequential composition*, the *Laplace* method preserve $(\epsilon' * m)$-differential privacy, which is equal to ϵ-differential privacy.

In *Graph Update* method, the release answer set A_r are the combination of noise answers A_n and synthetic answers A_s. Only A_n consume privacy budget, while A_s do not. In algorithm 1, even Step 4 adds *Laplace* noise to the true answer, the noise result does not release directly. Only when the algorithm processed to Step 7, in which a_n is released, the algorithm consumes the privacy budget. Suppose there are $j(0 \le j \le m)$ queries in F is released by synthetic answers, the algorithm preserves $((m-j)*\epsilon')$-differential privacy. As $(m-j)*\epsilon' \le m*\epsilon'$, the *Graph Update* method preserve more strict privacy than tractional *Laplace* method.

4.2 Utility Analysis

We applied *Mean Absolute Error* (MAE) as the utility of the query set on a graph. MAE_r of release answer A_r is defined as Eq. 6

$$\begin{aligned} MAE_r &= \frac{1}{m}|\widehat{F_i}(G) - F_i(G)| \\ &= \frac{1}{m}\sum_{f_i \in F}|\widehat{f_i}(G) - f_i(G)| \\ &= \frac{1}{m}\sum_{a_i \in A_r}|a_{ri} - a_{ti}| \\ &= \frac{1}{m}|A_r - A_t| \end{aligned} \tag{6}$$

Similarly, MAE_n of noise answers and MAE_s of synthetic answers are defined as Eq. 7 and Eq. 8, respectively.

$$MAE_n = \frac{1}{m}|A_n - A_t| \tag{7}$$

$$MAE_s = \frac{1}{m}|A_s - A_t| \tag{8}$$

It is obvious that for true answers A_t, the MAE is zero. MAE_n represents the performance of traditional *Laplace* method. A lower MAE implies a better performance.

The target of *Graph Update* method is to achieve a lower MAE_r in a fixed privacy budget. We apply a simulated figure 3 to illustrate the relationship between MAE values and the size of the query set m.

In Figure 3, x axis is the size of the query set and y axis is the value of MAE. For noise answer A_n, MAE_n is arising with the increasing of m. We apply a smooth line to represent the MAE_n in this simulated figure. In real case, the line is fluctuated as the noise is derived from *Laplace* distribution. The MAE_s is decreasing at the beginning with the increasing of m. When it reaches to its lowest point, the MAE_s begins to rise with the enhance of m. This is because with the update of the graph, the synthetic graph is getting more and more accurate, MAE_s is keeping decreasing. However, as the iteration procedure is controlled by the noise answer, it is impossible for synthetic graph to equal to the original graph, no matter how large m is. On the contrary, with the increasing of m, more noise will be introduced to iteration and the synthetic graph will be far away from the original graph.

As A_r is the combination of A_n and A_s, MAE_r of release answers can be reflected by synthetic answer MAE_s and noise answer MAE_n. Figure 3 shows that MAE_s will below MAE_n when the query size reaches to m_1. After reaching to a lowest point, it begins to increase. After reaching to m_2, the MAE_s is higher than MAE_n. Consequently, when m in the scale of $[0, m_1) \cup (m_2, m]$, the MAE_r is dominated by noise answer MAE_n. When m in the scale of $[m_1, m_2]$, the MAE_r is dominated by synthetic answer MAE_s. By this way, in the scale of $[0, m]$, the MAE_r of release answers is smaller than MAE_n, which means that the performance of the proposed *Graph Update* method is better than the traditional *Laplace* method.

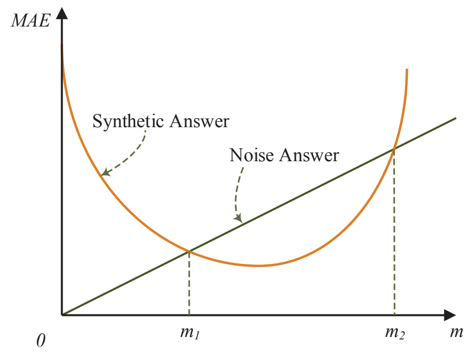


Figure 3: Utility of the Query set on a Graph

We will use experiment to confirm the optimal MAE in Section 5. As random noise is introduced to the method, points m_1 and m_2 can hardly be determined. In real case, they are ranges rather than exact points. In the *Graph Update* method, the parameter T is used to adjust the range.

5. EXPERIMENT AND ANALYSIS

This section evaluates the performance of the proposed *Graph Update* method by answering the following questions:

- *How do the parameter T impact on the performance of Graph Update*

  Graph Update contains an essential parameter T that controls releasing outputs. In the first part of the experiment, we will test the impact of T in terms of *Mean Absolute Error* (MAE).

- *What is the performance of Graph Update comparing with the traditional Laplace method and other related methods?*

  The proposed Graph Update method aims to effectively answer a large set of queries. We will investigate the performance of *Graph Update* on a set of queries and compare it with the traditional *Laplace* method and a *Correlated* method proposed by Chen et al. [4]. In addition, the performance will also be measured under different privacy budgets.

5.1 Datasets and Configuration

The experiment involve with four datasets, which are collected from Stanford Network Analysis Platform (SNAP) [13].

Table 1: Graph Datasets

Type	Name	Nodes	Edges
Social Networks	ego-Facebook	4,039	88,234
Social Networks	wiki-Vote	7,115	103,689
Internet peer-to-peer networks	p2p-Gnutella08	6,301	20,777
Collaboration networks	ca-GrQc	5,242	14,496

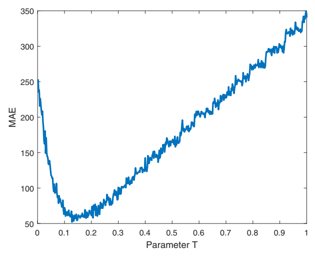
In the experiment, we consider the degree query on nodes, which is similar to the count query on relation dataset. To preserve the edge privacy, the degree query has the sensitivity of 1, which means deleting an edge will have maximum impact of 1 on the query result. The performance of results is measured by *Mean Absolute Error* (MAE) 6.

5.2 Evaluation of Parameters

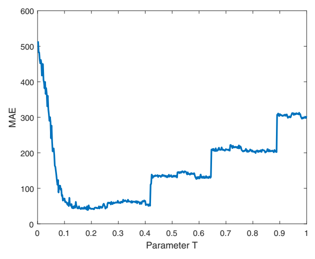
In *Graph Update*, T is a threshold that controls the releasing results and has a direct impact on the performance of the query result. To achieve a comprehensive investigation, we investigate the impact of T on the utility. The parameter T varied from 0.02 to 1 with a step of 0.02 with the size of query set equal to 10 and privacy budget ϵ fixing to 1.

Fig. 4 shows that at the beginning, it is apparent with an increasing of T, MAE drops quickly. But when T achieves a threshold, MAE reaches its minimum and keeps increasing. For example, as shown in Fig. 4a, MAE keeps decreasing until $T = 0.1100$, with $MAE = 50.37$ at its lowest point. After this, as T increases, MAE keeps rising. This trend can be observed in other data sets. in Fig. 4b, the MAE reaches its minimum when $T = 0.2100$ and remains stable until $T \geq 0.7100$. After that, T is keeping increasing. Fig. 4c and 4d show the same trend. This pattern shows the impact of T on the performance. At the beginning, when T is relatively small, the increasing value of T will decrease the update round, which means the privacy budget can be saved and less noise is added to query answers. Thus the MAE is keeping decreasing. However, when T reaches to a threshold, the decreasing number of update rounds leads an inaccurate synthetic graph. Consequently, we choose a suitable T for each dataset to achieve a minimal MAE. According to the results shown in Fig. 4, we can chose $T = 0.3100$ as the parameter in `ego-facebook` dataset; $T = 0.3600$ in `Wiki-Vote` dataset; $T = 0.2600$ in `p2p-Gnutella08` dataset and $T = 0.4100$ in `ca-GrQc` dataset.

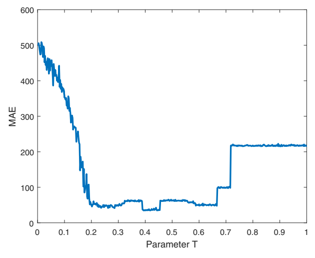
The parameter θ is another important parameter that affects *Graph Update*. To evaluate the impact of θ, we use 100 queries and vary it from 0.1 to 1.



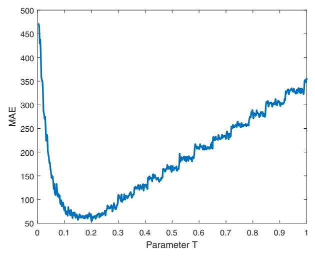
(a) `ego-Facebook`



(b) `Wiki-Vote`



(c) `p2p-Gnutella08`

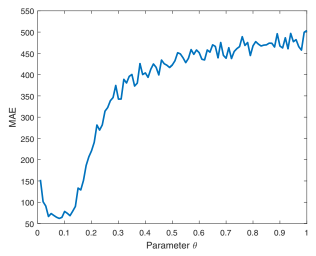


(d) `ca-GrQc`

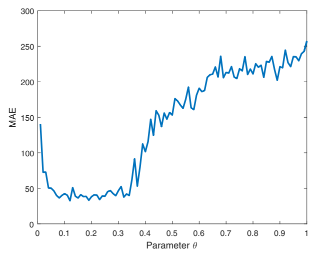
Figure 4: Impact of T on Update Round

Fig. 5 shows that in all datasets, when θ is increasing at the beginning, the MAE of *Graph Update* is decreasing. However, when MAE reaches to its lowest value, it begins to keep increasing with the enhancement of θ. This trend means that when θ is too small, the graph can not be fully updated within 100 queries. Consequently, MAE is keeping decreasing with the increasing of the θ. In this particular scale, a larger θ can help to update the graph in limited queries. But this decreasing of MAE cannot be lasted long, when θ is large enough, the MAE will be raised with the increasing of θ. During this process, we can choose a suitable θ that can minimize MAE.

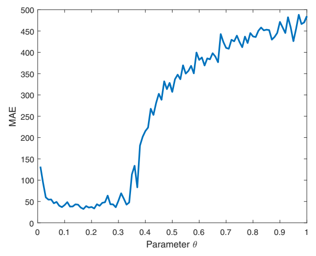
The `ego-Facebook` dataset in Fig. 5a shows that when θ is reached to 0.0800, the minimum MAE is 70.100. This means for this datset, a proper θ could be 0.0800 when answering 100 queries. When θ reaches to 0.1300, MAE is increasing sharply. Fig. 5b shows that for `Wiki-Vote` dataset, MAE keeps in a low level when θ is in the scale of 0.1000 to 0.3500. Similarly, Datasets `p2p-Gnutella08` and `ca-GrQc` are showing the



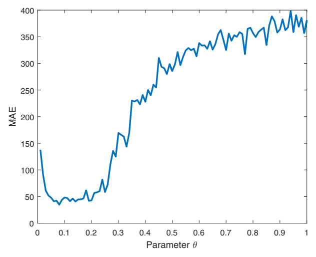
(a) `ego-Facebook`



(b) `Wiki-Vote`

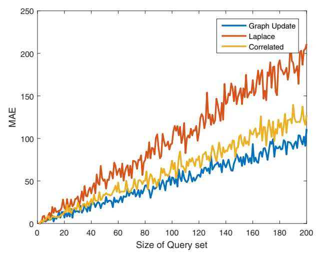


(c) `p2p-Gnutella08`

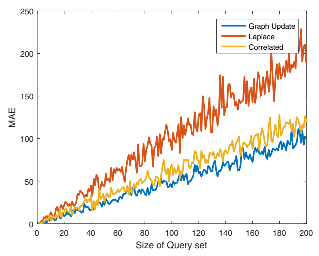


(d) `ca-GrQc`

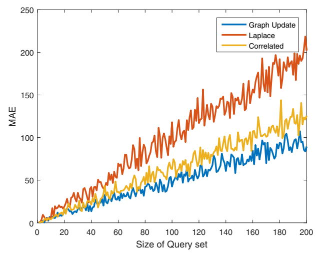
Figure 5: Impact of θ on Update Graph



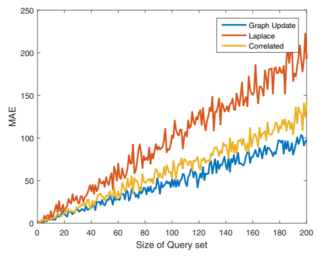
(a) `ego-Facebook`



(b) `Wiki-Vote`



(c) `p2p-Gnutella08`



(d) `ca-GrQc`

Figure 6: Performance of different methods

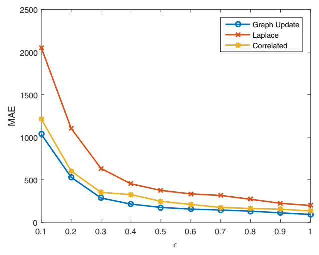
same trend. In Fig. 5c and Fig. 5d, we can observe that θ can be $0.1 - 0.3$ and $0.1 - 0.25$ for those two datasets, respectively.

5.3 Performance Evaluation on Diverse Size of Query Sets

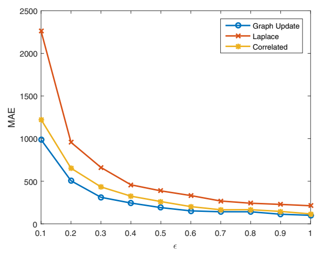
The performance of the *Update Graph* is examined through comparison with the state-of-the-art *Laplace* method [6] and Correlated method [4]. We set the size of query sets from 1 to 200, in which each query is independent to each other. Parameters T and θ as optimal one for each dataset and the ϵ is fixed at 1 for all methods.

According to the Figure 6, we can generally get the performance of the *Graph Update* comparing with other methods. First, we observe that with the increasing of the size of the query sets, MAEs of all methods are increasing approximately in linear. This is because the queries are independent to each other and the privacy budget is arranged equally to each query. With the linear increasing of the query number, the noise added to each query answer is enhanced linearly.

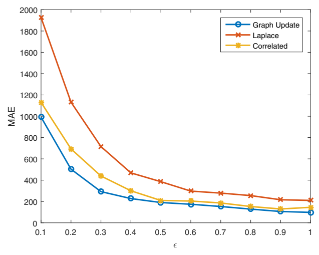
Second, Figure 6 shows that *Update Graph* has lower MAE comparing with other two methods, especially when the size of the query set is large. As shown in Figure 6a, when the size of query set is 200, the MAE of *Graph Update* is 99.8500 while the *Laplace* method has MAE of 210.0020, and the *Correlated* method has MAE of 135.2078 which is 52.45% and 26.15% higher than the proposed *Update Graph*. This trend can be observed in Figure 6b, 6c and 6d. *Graph Update* has better performance because part of query answers does not consume any privacy budget, while noise is only added in the updated procedure. Other methods, including *Laplace* method consume the privacy budget when answering every query. The result shows the effectiveness of *Graph Update* in answering a large set of queries.



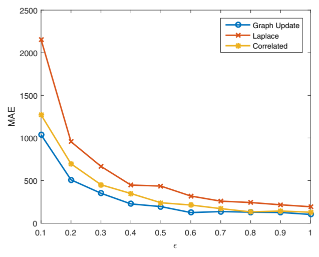
(a) `ego-Facebook`



(b) `Wiki-Vote`



(c) `p2p-Gnutella08`



(d) `ca-GrQc`

Figure 7: Impact of ϵ on the performance

Third, it is worth to mention that when the size of the query set is limited, the proposed *Graph Update* may not necessary outperform the *Correlated* method. Figure 6a shows that when the size is less than 20, MAEs of *Graph Update* and the *Correlated* method are mixed together. This is because when the query set is limited, the synthetic graph can not be fully updated and may differ from the original graph largely. Therefore, the performance may not necessary outperform other methods significantly. This result shows that *Graph Update* is more suitable in scenarios that need to answer a large amount of queries.

5.4 Performance Evaluation on Diverse Privacy Budgets

In addition, we test the performance of *Graph Update* with varying privacy budgets ϵ from 0.1 to 1 with 0.1 step, and a query set with 100 queries.

It is observed that as ϵ increases, the MAE evaluation becomes better, which means that the lower the privacy preservation level, the better the utility. In Fig. 7a, the MAE of *Graph Update* is 1035.40 when $\epsilon = 0.1$. Even though it preserves a strict privacy guarantee, the query answer is inaccurate and can not be used in real world. When $\epsilon = 0.7$, the MAE drops to 144.0774, retaining an acceptable utility in the result. The same trend can be observed on other datasets. For example, when $\epsilon = 0.7$, the MAE is 141.7209 in Fig. 7b, and is 153.0225 in Fig. 7c. Both show great improvement compared to $\epsilon = 0.1$. These results confirm that the utility is enhanced as the privacy budget increases.

We observe that the MAE decreases faster when ϵ ascends from 0.1 to 0.4, than when ϵ ascends from 0.4 to 1. This indicates that a larger utility cost is needed to achieve a higher privacy level ($\epsilon = 0.1$). We also observe that *Graph Update* and other methods perform stably when $\epsilon \geq 0.7$. This indicates that *Graph Update* is capable of retaining the utility for data release while satisfying a suitable privacy preservation requirement.

Figure 7 also shows that *Graph Update* has a lower MAE on all values of ϵ in all datasets. In Figure 7a when $\epsilon = 0.3$, *Graph Update* has an MAE of 285.2505 while the *Laplace* method has 633.0361, with an improvement of 55.41%. When $\epsilon = 1$, *Graph Update* achieves an MAE of 91.8746 and outperforms *Laplace* which has 197.0926. These results imply that *Graph Update* outperforms *Laplace* when answering a large set of queries. When compared with other methods, *Graph Update* still has lower MAE. When $\epsilon = 0.3$, the *Correlated* has a MAE of 351.5872, which is higher than *Graph Update*. This trend is consistent with the increase in ϵ. When ϵ reaches 1, *Correlated* is 133.1599, which is higher than that of *Graph Update* with $MAE = 91.8746$.

The evaluation shows that the *Graph Update* method retains a higher accuracy compared to other methods when answering large sets of queries, and its performance is significantly enhanced with the increase in the privacy budget. We can select a suitable privacy budget to achieve a better trade-off.

6. CONCLUSIONS

Nowadays, the privacy problem has aroused people's attention [3, 12, 20]. Especially the online social network data, which contains a massive personal information. How to release social network data is a hot topic that attracts lots of attention. However, existing methods cannot provide accurate results when releasing large numbers of queries due to the huge noises added to query results. This paper proposed an interaction method that transfers the query release problem to an iteration based update process, so as to providing a practical solution for publishing a sequence of queries with high accuracy. We evaluate our methods on numerous graphs. Through extensive experiments on real datasets we have shown that our method is effective and outperforms the Laplace method and the correlated method. In the future, we will consider much more complied quires, such as cut queries and triangle queries, which can allow researchers get more information of the dataset while still can guarantee users' privacy.

7. ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of China under Grant No. 61502362 and the Natural Science Foundation of HuBei province (China) under Grant No. 2014CFB354.

REFERENCES

1. Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190. ACM, 2007.
2. Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.
3. Shuchih Ernest Chang and Anne Yenching Liu. Information security in practices: Exploring privacy and trust in computer and inernet surveillance. *Comput. Syst. Sci. Eng.*, 31, 2016.
4. Rui Chen, Benjamin CM Fung, S Yu Philip, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, 23(4):653–676, 2014.
5. Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664. ACM, 2013.
6. Cynthia Dwork. Differential privacy: a survey of results. In *TAMC'08: Proceedings of the 5th international conference on Theory and applications of models of computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer-Verlag.
7. Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.
8. Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 61–70, 2010.
9. Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 169–178, 2009.
10. Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 169–178. IEEE, 2009.
11. Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.
12. D Davide Lamanna, Giorgia Lodi, Flavio Bertini, and Roberto Baldoni. How to act without being observed: Progressive privacy in desktop-as-a-service. *Comput. Syst. Sci. Eng.*, 28, 2013.
13. Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `Mhttp://snap.stanford.edu/data`, June 2014.
14. Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 94–103, 2007.
15. Darakhshan Mir and Rebecca N Wright. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 167–176. ACM, 2012.
16. Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.
17. Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98. ACM, 2011.
18. Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy*, 6(2):127, 2013.
19. Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 911–920. ACM, 2014.
20. Yangyang Xu, Zhaobin Liu, and Zhonglian Hu. Ascending partition method for differentially private histogram publication. *Comput. Syst. Sci. Eng.*, 31, 2016.