ℙ𝕊𝔾: Local Privacy Preserving Synthetic Social Graph Generation

Hongyu Huang, Yao Yang, and Yantao Li(✉)

College of Computer Science, Chongqing University, Chongqing 400044, China
yantaoli@cqu.edu.cn

Abstract. Social graph, as a representation of the network topology, contains users' social relationship. In order to obtain a social graph, a server requires users to submit their relationships. As we know, using or publishing social graph will cause privacy leakage to users. For this sake, it is necessary to generate synthetic social graph for various usages. In this paper, we propose ℙ𝕊𝔾, a local Privacy Preserving Synthetic Social Graph Generation method. In order to protect users' privacy, we utilize the local differential privacy model and a truncated Laplace mechanism to allow users to perturb their own data before submission. We then model the graph generation as a combinatorial optimization problem and design a greedy algorithm to maximize the utility of the generated graph. Through theoretical analysis and extensive experiments, we show that our method satisfies local differential privacy as well as maintains attributes of the original social graph.

Keywords: Social networks · Local differential privacy · Synthetic graph generation

1 Introduction

Social graph is often used to represent correlations among all users. By analyzing the social graph, we can obtain much valuable knowledge about the network which can help to improve the performance or quality of services. When generating a social graph, the server requires every user in the network to submit the social connections. Obviously, the information is sensitive which brings serious risk to users' privacy if it is misused by the server or exposed to the third party. Therefore, how to protect the individual privacy of users in social networks by a privacy-preserving way is an emerging research topic. A reasonable solution is to allow each user to add noise in the submission and then the server generates a synthetic social graph. Local differential privacy, as a privacy protection method, has been widely used to protect users' privacy. According to the local differential privacy, the probability that the server infers the real data from the noisy submission can be controlled by the user, which permits users to control their privacy leakage.

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 406, pp. 389–404, 2021.
https://doi.org/10.1007/978-3-030-92635-9_23

Since the server only has noisy data, it is challenging to guarantee that the synthetic graph has similar attributes to the original social network. However, most of the existing works generate the synthetic graph in a randomized way. That is, the server first requires users to submit social network attributes, such as degree sequences, subgraph counts and degree distributions. Then, the server divides users into different groups based on the information, and calculates the probability of generating an edge between intra-cluster and inter-cluster. Finally, edges are randomly generated according to the probability. The above random model is similar to the BTER model [1]. Nevertheless, the attributes of the social network synthetic graph generated by the random model is indeterminate, and thus we cannot obtain values that are similar to the attributes of the original social network graph.

In this paper, we propose $\mathbb{PSG}$, a local Privacy Preserving Synthetic Social Graph Generation mechanism that generates synthetic graphs under local differential privacy. In $\mathbb{PSG}$, the server first divides users into different groups and requires users to submit the number of friends in each group. Then, users form their degree vectors based on groups, add truncated Laplace based on privacy budget and normalization factor, generate a noise degree vector, and send it to the server. Next, the server uses the noise degree vector to calculate the utility of an edge generation, establishes a utility matrix, and constructs a social network synthetic graph, which aims to maximize the utility. To validate the effectiveness of $\mathbb{PSG}$, we conduct extensive experiments using real social network datasets to verify that the synthetic graph generated by $\mathbb{PSG}$ can maintain attributes of the original social network under the desired privacy level.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem of synthetic social graph generation under local differential privacy. Section 4 presents the proposed solution, $\mathbb{PSG}$, and proves that it satisfies local differential privacy. Section 5 evaluates the performance of $\mathbb{PSG}$ by extensive experiments. Last, Sect. 6 concludes this work.

2 Related Work

There are a large number of methods proposed to protect social networks and generate social network graphs, which have been investigated for over decades with flourished results.

2.1 Social Network Privacy Protection

Early works on privacy-preserving social network mainly focus on data mining of social network. Chen *et al.* proposed a privacy policy recommendation model, which aimed to recommend for text-based posts privacy policies to users [2]. Sweeney *et al.* proposed a k-anonymity model, which made each record have at least the same attribute value as the other $k-1$ records in the data, thereby reducing privacy leakage caused by link attacks [3]. In order to overcome the shortcomings of the k-anonymity model, Machanavajjhala *et al.* proposed a l-diversity privacy model, where attackers inferred the probability of each k anony-

mous dataset of private information in social networks was less than $\frac{1}{l}$ [4]. Unfortunately, the aforementioned models are vulnerable to attackers with stronger background knowledge, which has stimulated the use of differential privacy to obtain more rigorous privacy guarantees.

The latest research on applying differential privacy in social networks mainly focuses on: 1) the release of social network statistics, such as degree histograms and subgraph counts [5–7], and 2) the publication of a synthetic graph of a social network, such as BTER [1].

2.2 Synthetic Graph Generation

There are some contributions on formal models for synthetic graph generation. The earliest model is the Erdos-Renyi (ER) model proposed by Paul *et al.*, which assumes that an edge is randomly generated in the network based on the same probability [8]. Aiello *et al.* proposed the CL model, which was similar to the ER model, but each edge has different probability that satisfies the node degree distributions [9].

The aforementioned graph models ignore to take the protection privacy of social network users in the generation of synthetic graph into account. However, differential privacy is widely used to protect users privacy in social synthetic graph. Qin *et al.* proposed LDPGen model which was a multi-phase approach to generate synthetic decentralized social graph under local differential privacy [10]. Zhu *et al.* designed a degree-differential privacy mechanism, and proposed a graph generation approach with field theory [11].

According to previous literature, we find that when the existing social network graph models generate edge relationships between nodes, most of them link nodes by utilizing probability-based methods. However, our method is able to calculate the probability of social network edge relationships for social network graph generation, which is rarely studied in previous works.

3 Preliminaries

3.1 System Overview

We consider a social network with N users. We model the network as an undirected graph $G = (V, E)$, where V and E indicate sets of users and their connections, respectively. We assume that $|V| = N$ and $|E| = M$. We denote $v_i \in V$ in the graph as the $i-$th user in the network. We say v_i and v_j are neighbors in the graph, i.e., an edge $e_{i,j} \in E$ connects v_i and v_j, iff the two users are friends in the real world. An untrusted data curator, which we call a server in the rest of this paper for simplicity, collects neighbor lists from users to generate a synthetic network. With all neighbor lists, the server uses a three-stage process to generate the network, inspired by [10].

In the first stage, the server randomly divides all users into k disjoint groups, denoted as $R = \{R_1, R_2, ..., R_k\}$. Then, the server distributes these groups to

all users. Each user computes friends in each group and then obtains a degree vector $\gamma^i = (\gamma_1^i, \gamma_2^i, ..., \gamma_k^i)$, where γ_k^i denotes the number of friends that user v_i has in the $k-$th group. In order to protect the local privacy, each user perturbs the vector γ^i before submitting it to the server.

In the second stage, the server re-constructs the group R and distributes it to all users again. At this stage, the number of groups and members in each group are different from the first stage. When users obtain R, they do the same as the first stage that compute the degree vector based on the groups, then perturb the vector, and submit it to the server.

In the last stage, the server computes the degree of each user and assigns some other users as his friends. This assignment runs iteratively until all users have enough friends. To this end, the server builds the synthetic social graph.

3.2 Problem Statement

In order to generate a synthetic network in a privacy-preserving way, we need to solve two problems in the stages in previously introduced procedure. The first one is how to find a method to add the noise to the degree vector for the user privacy protection. Specifically, we use the differential privacy [12] as our privacy model. For randomized mechanism $\mathcal{M}$ on a graph, it satisfies $\epsilon-$differential privacy iff for any two neighboring graphs G and G' which only differ in one edge, and any possible $s \in range(\mathcal{M})$, we have $\frac{Pr[\mathcal{M}(G)=s]}{Pr[\mathcal{M}(G')=s]} \leq e^{\epsilon}$. Generally, we need to add Laplace noise to each entry of the degree vector to satisfy the differential privacy. As we know, however, the degree of each node on the graph must be non-negative and less than the number of nodes. Thus, simply using Laplace distribution may generate out-of-bounds noise. Hence, we design a randomized mechanism that satisfies differential privacy and outputs degrees in the range $[0, N-1]$ to address the first problem.

Our second problem is to generate a synthetic social network graph based on the degree vector submitted by users. Most of the existing works adopt randomized methods to connect nodes with edges. For fixed input parameters, the output network graph of a randomized method may be different. Hence, the performance of synthetic networks cannot be guaranteed. In order to overcome this defect, we try to find a deterministic method to improve the randomness. Suppose we have a metric to measure the utility of connecting v_i and v_j by edge, and then we can generate the synthetic social network graph by solving the following combinatorial optimization problem:

$$max \sum_{i=1}^{N} \sum_{j=1}^{N} u_{i,j} \cdot x_{i,j} \tag{1}$$

$$s.t. \sum_{j=1}^{M} x_{i,j} = d_i,\ i = 1, 2, 3, ..., N \tag{2}$$

$$x_{i,j} = x_{j,i} \tag{3}$$

$$x_{i,j} \in \{0,1\}, i = 1,2,...,N, j = 1,2,....,M \tag{4}$$

where $u_{i,j}$ is the utility of $e_{i,j}$, $x_{i,j} = 1$ indicates that $e_{i,j}$ exists and $x_{i,j} = 0$ for otherwise. The d_i is the degree of v_i which can be easily obtained by adding all entries of γ^i. Last, since the graph is undirected, we have $x_{i,j} = x_{j,i}$. Therefore, we propose SNGCO, a Synthetic Network Generation by Combinatorial Optimization to address the second problem.

4 Design Details

4.1 Privacy Protection Mechanism Design

As a de facto standard, local differential privacy has been widely used in social networks for user sensitive data protection. It mainly focuses on publishing social network synthetic graph and various types of graph statistics, including social network degree distributions [13], degree sequences [14], subgraph counts [7], and so on. In this context, differential privacy is proposed to avoid the leakage of individual private information in social networks.

Specifically, differential privacy mechanism allows users to submit noisy data. In social networks, users with their own privacy considerations individually utilize traditional Laplace algorithm that satisfies differential privacy to perturb their personal information, and then submit the noisy data to a server. The server use these data to infer the overall statistic information of social networks. However, since the perturbation range of the traditional Laplace algorithm is not restricted, the noise range can be $[-\infty, +\infty]$, which cannot be applied to all scenarios, and may generate erroneous data beyond this range. For example, after the user receives the group result from the server, the user submits the number of friends in each group according to the division result, i.e., the degree vector. This value is non-negative and must be less than the number of nodes in the group. Obviously, the traditional Laplace cannot meet such requirements, so we adopt the truncated Laplace algorithm to control the noisy value within the valid range. By briefly reviewing the algorithm, we notice that the range of the Laplace noise function on the left and right sides of the true value is equal, so we use the distance from the true value to the upper and lower limits to determine the function curves on the left and right sides to obtain the function distribution on the left and right sides. To better illustrate the local differential privacy, it can be defined as:

Definition 1 (*Local differential privacy* [15]). *A randomized mechanism M satisfies ε-local differential privacy, iff for any two input data t and* t^**, we have*

$$\frac{\Pr[M(t) = s]}{\Pr[M(\mathrm{t}^*) = s]} \leq e^{\varepsilon}. \tag{5}$$

Where $s \subseteq$ range (M), and then we say that M satisfies ε-local differential privacy.

Definition 2 (*Local sensitivity* [12]). *Given a graph G = (V, E) containing user nodes* $V = \{v_i | 1 \leq i \leq n\}$, *and any function f, the local sensitivity of f can be defined as:*

$$LS_{f(D)} = \max \| f(G) - f(G') \|_1, \tag{6}$$

where G and G' are neighboring graph of users.

Definition 3 (*Laplace algorithm* [12]). *Given a dataset D and function f, if privacy protection mechanism Y satisfies ε-local differential privacy, we have:*

$$Y = f(D) + laplace(\frac{\Delta f}{\varepsilon}), \tag{7}$$

where Δ f is the sensitivity.

As mentioned earlier, our goal is to generate a synthetic graph with attributes similar to the original social network graph under user privacy protection. Now we formally describe our solution (referred to $\mathbb{PSG}$), as illustrated in Algorithm 1 (based on the truncated Laplace algorithm). Specifically, the user calculates the number of friends in each group according to the received group results to form a degree vector, and then uses the truncated Laplace algorithm to add noise to the degree vector. Finally, the noise degree vector is returned to a server for data processing.

As described in Algorithm 1, the user mainly uses the truncation mechanism twice in the following two steps. The first step is to add noise to user's node degree (lines 1–4). Specifically, the user initially calculates the distance between himself and the left and right ends of the valid range (line 1). In order to ensure that the range of the probability density function of the truncated Laplace is 1, the user calculates the respective areas of the two ends of the function (line 2). Next, the user calculates the normalization factor based on the previous calculation results (line 3). Then, the truncated Laplace function is obtain though multiplying the normalization factor by the original Laplace function, and the user adds the noise by this function (line 4). In the second step, the user also performs privacy protection for the node degree vector. The user repeats truncated Laplace algorithm on the degree vector, where the difference from the first step is that the valid range varies with the size of each group. To ensure the consistency of user data, the cumulative sum of the user degree vector needs to be equal to the user degree. Therefore, the second step is to add noise to the user degree vector ensuring that the cumulative sum of the user degree vector is equal to user's node degree (lines 6–13). In order to improve the data accuracy, the user initializes the cumulative sum of the degree vector s, and performs in an ascending order processing on each element of the degree vector (lines 5–6). Similarly, as the same step as the first user degree noise addition, the user calculates the valid range in each group, obtains the normalization factor, and calculates the noise addition value in each group (lines 7–10). According to the constraint of the degree, when the cumulative sum of the degree vector is greater than the degree, the i-th element is equal to the absolute value of the difference between the cumulative sum of the previous $(i-1)$-th elements and the degree

(lines 11–13). Finally, according to the above operations, the user constructs and returns a noise degree vector (lines 14–15).

Algorithm 1. Truncated Laplace Mechanism for Nodes Degree Vector

Input: *group*
Output: Noisy degree vector q

1: $\Delta L = \mu$, $\Delta R = (N-1) - \mu$
2: $L = \frac{e^{\frac{-\Delta R}{\sigma}}}{2}$, $R = \frac{e^{\frac{-\Delta L}{\sigma}}}{2}$;
3: $n = \frac{1}{1-(R+L)}$;
4: $\widetilde{d} = d + nLap(\frac{1}{\varepsilon})$;
5: $s = 0$;
6: for each d_i in q_i with ascending order :
7: $\Delta L_i = \mu_i$, $\Delta R_i = |q_i| - 1 - \mu_i$;
8: $L_{\mathrm{i}} = \frac{e^{\frac{-\Delta R_i}{\sigma}}}{2}$, $R_{\mathrm{i}} = \frac{e^{\frac{-\Delta L_i}{\sigma}}}{2}$;
9: $n_{\mathrm{i}} = \frac{1}{1-(L_{\mathrm{i}}+R_{\mathrm{i}})}$;
10: $\widetilde{d_i} = d_i + n_i Lap(\frac{1}{\varepsilon})$;
11: $s = s + \widetilde{d_i}$;
12: if $s > \widetilde{d}$:
13: $\widetilde{d_i} = |s_{i-1} - \widetilde{d}|$;
14: $q = < \widetilde{d_1}, \ldots, \widetilde{d_{|q_i|}} >$;
15: **return** q

4.2 Privacy Analysis

In this section, we demonstrate that PSG satisfies local differential privacy. Let $\{\widetilde{\gamma}_1, \widetilde{\gamma}_2, ..., \widetilde{\gamma}_k\}$ be user's noise degree vector based on k groups, where $\widetilde{\gamma}^{v_i} = \gamma_j^{v_i} + n_j^i Lap(\frac{\Delta F}{\varepsilon})$. Denote this mechanism as L.

Theorem 1. *The PSG satisfies ε-local differential privacy.*

Proof. Without loss of generality, any two users v_i and v_j send degree vectors of $\gamma^{v_i} = \{\gamma_1^{v_i}, \gamma_2^{v_i}, ..., \gamma_k^{v_i}\}$ and $\gamma^{v_j} = \{\gamma_1^{v_j}, \gamma_2^{v_j}, ..., \gamma_k^{v_j}\}$ to a server. If γ^{v_i} and γ^{v_j} differ in one element, we assume $\gamma_k^{v_i} \neq \gamma_k^{v_j}$. Then, we can obtain $|\gamma_k^{v_i} - \gamma_k^{v_j}| = 1$, and $\Delta F = 1$. Given an arbitrary vector $s = (s_1, \ldots, s_k)$, based on traditional Laplace algorithm, the privacy guarantee of differential privacy can be shown in Eq. (8):

$$\begin{aligned} \frac{\Pr[M(\gamma^{\mathrm{u}}) \in s]}{\Pr[M(\gamma^{v}) \in s]} &= \frac{\Pr[M(\gamma_1^u) = s_1] \ldots \Pr[M(\gamma_k^u) = s_k]}{\Pr[M(\gamma_1^v) = s_1] \ldots \Pr[M(\gamma_k^v) = s_k]} \\ &= \frac{\Pr[M(\gamma_k^u) = s_k]}{\Pr[M(\gamma_k^v) = s_k]} \\ &\leq e^{\varepsilon}. \end{aligned} \tag{8}$$

Similarly, when using truncated Laplace mechanism, we have $\Pr[L(\gamma_k^v) \in s_k] = n_k^v \frac{\varepsilon}{2\Lambda F} e^{-\frac{\varepsilon|s-\gamma_k^v|}{\Lambda F}}$, $\Pr[L(\gamma_k^u) \in s_k] = n_k^u \frac{\varepsilon}{2\Lambda F} e^{-\frac{\varepsilon|s-\gamma_k^u|}{\Lambda F}}$. It can be shown in Eq. (9):

$$\begin{aligned} \frac{\Pr[L(\gamma^{\mathrm{u}}) \in s]}{\Pr[L(\gamma^v) \in s]} &= \frac{\Pr[L(\gamma_k^u) = s_k]}{\Pr[L(\gamma_k^u) = s_k]} \\ &= \frac{n_k^u \frac{\varepsilon}{2\Lambda F} e^{-\frac{\varepsilon|s-\gamma_{\mathrm{u}}|}{\Lambda F}}}{n_k^v \frac{\varepsilon}{2\Lambda F} e^{-\frac{\varepsilon|s-\gamma_v|}{\Lambda F}}} \\ &= \frac{n_k^u}{n_k^v} e^{\frac{\varepsilon|s-\gamma_{\mathrm{u}}|-\varepsilon|s-\gamma_v|}{\Lambda F}} \\ &\leq \frac{n_k^u}{n_k^v} e^{\frac{\varepsilon|\gamma_{\mathrm{u}}-\gamma_v|}{\Lambda F}} \\ &= \frac{n_k^u}{n_k^v} e^{\varepsilon}. \end{aligned} \tag{9}$$

Since the scaling parameters are all the same, and both of L_k^u (i.e., L_k^v) and R_k^u (i.e., R_k^v) have a range of $[0, 0.5)$, we substitute the L_k^v and R_k^v into $L_k^v{=}L_k^u \mathrm{e}^{\varepsilon}$, $R_k^v{=}R_k^u \mathrm{e}^{-\varepsilon}$ to obtain the form shown in Eq. (10):

$$\begin{aligned} \frac{n_k^u}{n_k^v} e^{\varepsilon} &= \frac{1 - (L_k^u \mathrm{e}^{\varepsilon} + R_k^u \mathrm{e}^{-\varepsilon})}{1 - (L_k^u + R_k^u)} e^{\varepsilon} \\ &\leq e^{\varepsilon}. \end{aligned} \tag{10}$$

According to the above proof, we obtain

$$\frac{\Pr[L(\gamma^{\mathrm{u}}) \in s]}{\Pr[L(\gamma^{\mathrm{v}}) \in s]} \leq e^{\varepsilon}. \tag{11}$$

In $\mathbb{PSG}$, users add noise to their degree vector applying the truncated Laplace $n_i Lap(\frac{\Delta F}{\varepsilon_1})$ and $n_i Lap(\frac{\Delta F}{\varepsilon_2})$ twice. According to the composability property of differential privacy $\varepsilon = \varepsilon_1 + \varepsilon_2$, $\mathbb{PSG}$ satisfies differential privacy.

4.3 Synthetic Network Generation

As we mentioned in Sect. 3.1, there are two stages of interactions between the server and users. The operations in each stage are the same: the server distributes the partition to all users and they return perturbed vectors to indicate how many friends they have in each group of the partition. Let k_1 and k_2 be the numbers of groups in the first and second stages, respectively.

Before we show how to generate the synthetic network graph, we informally prove that the SNGCO problem is NP-hard. In brief, we transform the SNGCO into another equivalent problem. We know that the degree of v_i must be d_i (see the constraint (2)). We replace d_i with v_i, which we call dummy nodes, denoted as $v_{i,j}, j = 1, 2, ..., d_i$. Now we have two kinds of nodes: the original nodes and the dummy nodes. Please note that the degree of each dummy node is just 1. Obviously, we have $\sum v_i$ dummy nodes in total. Then we can equivalently

transform the SNGCO problem into connecting edges between v_i and $v_{k,j}$ under the constraint that $i \neq k$ and at most one of $v_{k,j}, j = 1, 2, ..., d_k$ can connect to v_i. Formally, we add a new constraint, i.e., $\sum_{i=1, i\neq j}^{N} x_{i,j} = 1$, to the SNGCO problem. Next, we re-consider the assignment problem, which is a well known NP-hard problem, to the SNGCO. In the assignment problem, we have M tasks and N workers. Each worker can obtain different gain when he takes different task. And one task can only be taken by one worker. We can simply regard N nodes as workers, and $\sum v_i$ dummy nodes as M tasks. Obviously, this reduction only needs polynomial time. To this end, since the assignment problem is NP-hard, the SNGCO problem is also NP-hard.

Algorithm 2. Utility-based Synthetic Graph Generation

Input: N, $Q = \{q_1, \ldots, q_{|N|}\}$
Output: Edge set E_T

Initialize m=a, index=b
for any v_i and v_j do :
  $u[v_i][v_j] = \frac{\sum_{t=1}^{k} (q_t^i \times q_t^j)}{\sqrt{\sum_{t=1}^{k} (q_t^i)^2} \times \sqrt{\sum_{t=1}^{k} (q_t^j)^2}}$;
Sort(v_i) by $\widetilde{d}_i$ in descending order;
for each v_i do;
  s=0;
  while($s \leq \widetilde{d}_i$) do;
    for j=0 to N-1 do;
      if $u[v_i][j] \geq m$ and $i \neq j$;
        index=j, $m = u[v_i][j]$;
    if e_{ij} not in E_T then;
      $E_T = E_T \cup e_{ij}$, $s = s + 1$,
      $u[v_i][j] = min(u[v_i])$;
return E_T

Comparing our problems with the assignment problem, we observe that our problem is more complicated than the traditional assignment problem. We aim to turn our problem into an assignment-like combinatorial optimization problem to solve. For the assignment problem and its variants, a large number of algorithms are proposed to obtain feasible solution close to the optimal algorithm, and the time complexity is low. Among them, the greedy algorithm is the simplest and most intuitive one.

In this section, we formally describes how to use greedy solution based on utility matrix to generate social graph, i.e., ℙ𝕊𝔾. According to Algorithm 2, the server has two stages: 1) the server calculates a utility matrix based on cosine similarity between nodes, and 2) the server uses a greedy algorithm to generate a social network synthetic graph. Specifically, in the user division phase, the server randomly divides users twice. First, it randomizes the

users into k_0 groups. Next, the server collects the user noise degree vector, and then uses k-means to classify users. The optimal k_2 value is calculated by $k_2 = \left\lceil \sum_{\eta=1}^{\eta_{\max}} p_\eta \left(\frac{d}{2} + \frac{(\frac{d}{2})^2 - 2(1+\sqrt{5})(\frac{d}{2})+1}{\varepsilon_2} \right) \right\rceil$ to obtain the latest user noise degree vector. In real life, we realize that if two people have more friends in common, then they are more likely to become friends. In the same way, if the degree vectors of two nodes are more similar, then they are more likely to be connected. In Algorithm 2, after server collects noise degree vector of users, using correlation with cosine similarity between nodes as the metric, an utility matrix is constructed to express the possibility of connecting between users (lines 2–3). In order to improve the efficiency of data processing, we sort node degree of users as a pre-processing (line 4), and then generate edges in synthetic graph though selecting the maximum value of utility matrix until the constraint conditions are met. Specifically, first, the sum of user degrees is initialized to 0. When the user degree does not satisfy the constraint, the maximum value is continuously selected from the user row corresponding to the utility matrix (lines 5–10). Next, if any two users corresponding to the maximum value do not have an edge, then add an edge to the synthetic graph, and the value of s is increased by 1. Otherwise, the value of s remains. The above operations are continued until the degree of constraint is satisfied (lines 11–13). Finally, we return to the edge set of the synthetic graph of the social network (line 15).

5 Performance Evaluation

In this section, we evaluate the performance of our proposed generating social network graph method. We first describe a large number of experiments to verify that $\mathbb{PSG}$ can maintain the attributes of the original social network graph under the same privacy level. Then, we compare our method with several graph generation models based on real social network datasets to verify that $\mathbb{PSG}$ is more efficient.

5.1 Datasets and Models

We apply our algorithm on real datasets from Stanford Large Network Dataset Collection [16]. In order to facilitate the comparison and versatility of our experiments, we convert the datasets into an undirected graph. The experiments involve two real social graphs and the details are illustrated as follows:

- Facebook: It is an undirected social graph consisting of 4,039 nodes and 88,234 edges.
- Lastfm: It contains 7,624 nodes and 27,806 edges in a social network.

To demonstrate the usability of our proposed algorithm compared with the traditional graph generation models, we focus on three aspects of utility including the mean absolute error (MAE) of clustering coefficient [17], modularity [18],

and Adjusted Rand Index (ARI) [19]. The importance of nodes is also an evaluation metric, since different users have different influences in networks. Nodes with high influence are more likely to affect other nodes. For example, a teacher is more influential than students in a teacher-student network. Therefore, the importance of different nodes in the network is not the same. So, we utilize the mean absolute error of eigenvector centrality (EVC) [18] to evaluate influence of users.

In this paper, We compare above metrics in our model with three graph generation models (FCL [20], BTER [1] and LDPGen [10]) based on differential privacy.

5.2 Evaluation Metrics

Clustering Coefficient. Local clustering coefficient, which represents a measure of the tendency of nodes in the graph to cluster together. The local clustering coefficient is defined as follows:

$$C_{\mathrm{i}} = \frac{2\Delta_i}{d_i(d_i-1)} = \frac{a_{ii}^{(3)}}{a_{ii}^{(2)}(a_{ii}^{(2)}-1)} \tag{12}$$

Where Δ_i is the number of triangles connected to user v_i in the social network, d_i is v_i degree, $a_{ii}^{(2)}$ represents the second power diagonal element of the neighboring matrix, and $a_{ii}^{(3)}$ is the third power diagonal element. The smaller the MAE value, the higher the data usefulness.

Modularity. It is used to measure the effect of community division results. Generally, the community satisfies that the nodes within the community have high similarity, while the similarity of external nodes is low. Therefore, modularity is proposed to measure community structure.

$$Q = \sum_{c=1}^{N} [\frac{M_c}{n} - (\frac{d_c}{2n})^2] \tag{13}$$

Where n represents the total number of edges in social network graph, M denotes the number of communities, M_c is the sum of edges within the community c, and d_c represents the sum of degrees in c.

Adjusted Rand Index. ARI is also a metric to measure the clustering effect of social networks. The larger the value of the ARI, the more similar the clustering result is with the real original social networks. Given a set of n elements $S = \{O_1, O_2, ..., O_n\}$, if $U = \{u_1, u_2, ..., u_Z\}$ and $Y = \{y_1, y_2, ..., y_R\}$ represent two different divisions of S satisfying $\cup_{i=1}^{Z} u_i = S = \cup_{j=1}^{R} y_j$, then ARI can be denoted:

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)} \tag{14}$$

$$E(RI) = E(\sum_{i,j} \binom{n_{ij}}{2}) = \frac{[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}]}{\binom{n}{2}} \tag{15}$$

$$\max(RI) = \frac{1}{2}[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}] \tag{16}$$

$$RI = \frac{\mathrm{a} + d}{a + b + c + d} \tag{17}$$

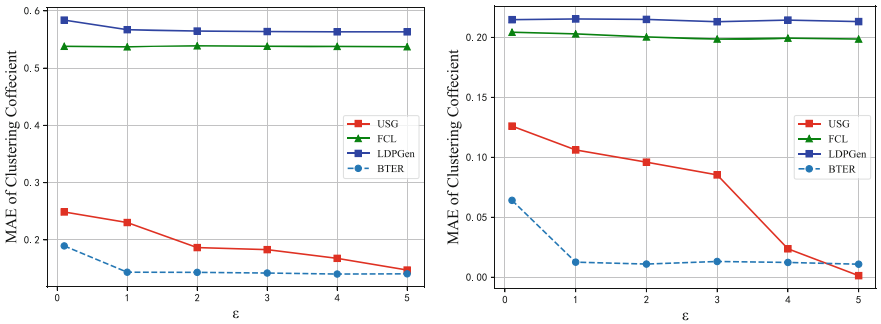
Where a (opposite to c) is the logarithm of nodes of the same group in U and the same group in Y. In the same way, b (opposite to d) is the logarithm of nodes that belong to the same group in U but belong to different group in Y. n_{ij} represents the number of nodes in the same group u_i and group y_j, and $n_{i\cdot}$ (resp. $n_{\cdot j}$) is the number of nodes of group u_i (resp. y_j).

Eigenvector Centrality. It measures the influence of nodes in the network, which means that the importance of a node depends not only on the number of its neighboring, but also on the importance of its neighboring nodes. It can be denoted as:

$$\mathrm{x_i} = \frac{1}{\lambda} \sum_{j=1}^{N} a_{ij} x_j \tag{18}$$

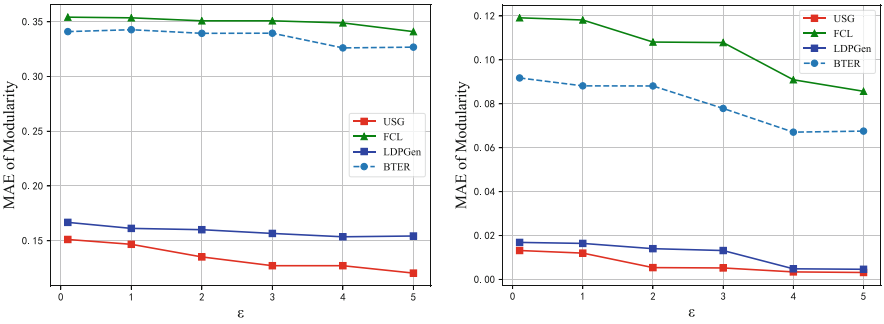
Where λ is a constant, and $\mathrm{x_i}$ is the importance metric value of node v_i.

5.3 Experimental Results



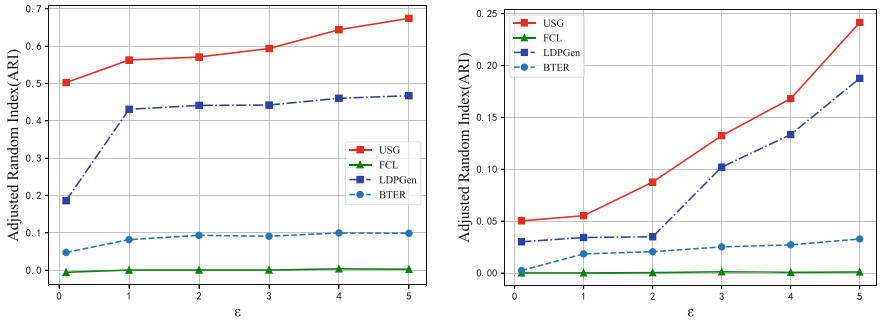
(a) Facebook (b) Lastfm

Fig. 1. Effect of ε on clustering coefficient



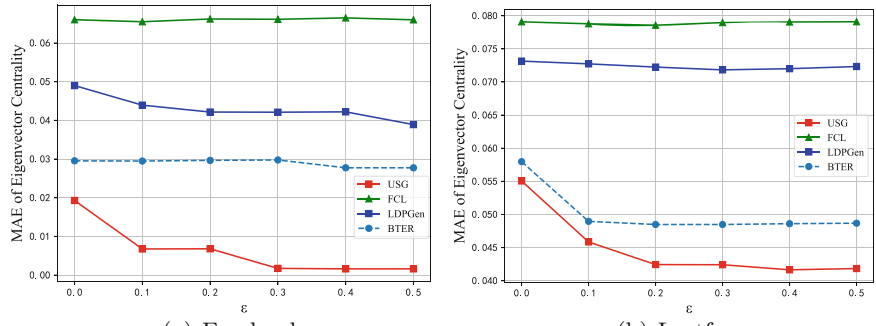
(a) Facebook (b) Lastfm

Fig. 2. Effect of ε on modularity



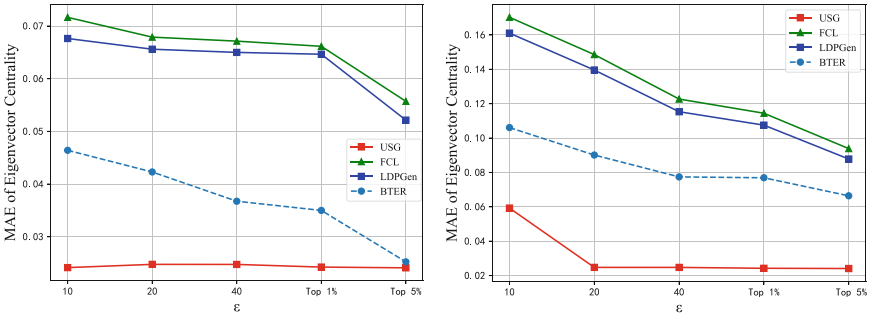
(a) Facebook (b) Lastfm

Fig. 3. Effect of ε on ARI



(a) Facebook (b) Lastfm

Fig. 4. Effect of ε on EVC



(a) Facebook (b) Lastfm

Fig. 5. MAE of top-k vertices

Figure 1(a) and 1(b) show that the MAE of clustering coefficient varies with privacy budget ε when using the Facebook and Lastfm datasets, respectively. We find that MAE of all methods decreases as ε increases, but BTER obtains the best result in all graph generation models. As expected, BTER is an algorithm proposed for optimizing the clustering coefficient. However, our model is very close to BTER compared to others, whereas FCL is far higher than others. This is because $\mathbb{PSG}$ considers the correlations between nodes in different groups to generate edges between nodes, and preferentially selects nodes with stronger correlation to link edges. Moreover, other graph generation models use probability-based methods to generate a synthetic graph, which makes the community structure unable to obtain relative stability under the same privacy level.

In Fig. 2 and Fig. 3, we use the two metrics of modularity and ARI to evaluate how well the graph models preserve the community structure using Facebook and Lastfm datasets, respectively. Similarly, if the MAE of the modularity is smaller and the value of ARI is more higher, the synthetic social graph is better to preserve community structure of the original social network graph. In Fig. 2 and Fig. 3, we shows $\mathbb{PSG}$ significantly outperforms other generation models. These are two reasons: 1) we use k-means to classify groups which preserves the community structure of users; 2) we connect nodes by constructing a correlation-based utility matrix, which maintains the relationship between the original social network nodes in a large extent.

We evaluate the influence of nodes with EVC which attempts to find the most important nodes in networks. In Fig. 4(a) and 4(b), we observe that the MAE of EVC varies with ε where EVC decreases as ε increases. Note that as privacy budget increases, the effect of privacy protection constantly decreases. In other words, the noise data added to the original social network graph using local differential privacy method is declining. Therefore, the more sensitive data exposed, the error of EVC becomes smaller and smaller. However, our method with the ε changes gradually, which shows that it can accurately identify important nodes and better protect the sensitive information of important nodes than other methods. From Fig. 5, we calculate the EVC of common nodes among the

most influential top-k nodes in social networks. We select top 10, top 20, top 40, 1% and 5% influential nodes to test. As the number of influential nodes increase, we can find a smaller MAE of EVC. Then, Figs. 4(a) and 4(b) illustrate the effect of ℙ𝕊𝔾 is better than other methods. This is because synthetic graph generated by the proposed optimization method is closer to the original graph, and the privacy of the original graph node is better protected based on local differential privacy. That means our methods can better protect privacy while providing accurate node analysis with the deepening of privacy protection.

6 Conclusion

In this paper, we propose ℙ𝕊𝔾, a novel graph generation model that generates a synthetic social network graph with the attributes similar to original graph under local differential privacy. The key idea of this paper is that we turn the problem of graph generation in social networks into a traditional optimal combination problem. ℙ𝕊𝔾 proposes an utility matrix based on correlation between nodes and generated edges, which is not used in probability models as the same way as the previous ones. In order to preserve community structure, we utilize k-means to classify groups and according to different groups, users locally perform privacy with truncated Laplace mechanism to protect sensitive information and send noise data to server, so as to better protect the privacy based on the user own privacy consideration. By theoretical analysis and experiments, we verify that ℙ𝕊𝔾 can not only protect user privacy, but also improve efficiency of data.

Acknowledgements. This work was partially supported by the National Natural Science Foundation of China under Grants 62072061 and U20A20176, and by the Fundamental Research Funds for the Central Universities under Grant 2021CDJQY-026.

References

1. Seshadhri, C., Kolda, T.G., Pinar, A.: Community structure and scale-free collections of erdős-rényi graphs. Phys. Rev. E **85**(5), 056109 (2012)
2. Chen, L., et al.: A privacy settings prediction model for textual posts on social networks. In: Romdhani, I., Shu, L., Takahiro, H., Zhou, Z., Gordon, T., Zeng, D. (eds.) CollaborateCom 2017. LNICST, vol. 252, pp. 578–588. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00916-8_53
3. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(05), 557–570 (2002)
4. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data (TKDD) **1**(1), 3-es (2007)
5. Zhang, X., Chen, R., Xu, J., Meng, X., Xie, Y.: Towards accurate histogram publication under differential privacy. In: Proceedings of the 2014 SIAM International Conference on Data Mining, pp. 587–595. SIAM (2014)
6. Ding, X., Zhang, X., Bao, Z., Jin, H.: Privacy-preserving triangle counting in large graphs. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1283–1292 (2018)

7. Sun, H., et al.: Analyzing subgraph statistics from extended local views with decentralized differential privacy. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 703–717 (2019)
8. Erdös, P., Rényi, A.: On the evolution of random graphs. In: The Structure and Dynamics of Networks, pp. 38–82. Princeton University Press (2011)
9. Aiello, W., Chung, F., Lu, L.: A random graph model for massive graphs. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pp. 171–180 (2000)
10. Qin, Z., Yu, T., Yang, Y., Khalil, I., Xiao, X., Ren, K.: Generating synthetic decentralized social graphs with local differential privacy. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 425–438 (2017)
11. Zhu, H., Zuo, X., Xie, M.: DP-FT: a differential privacy graph generation with field theory for social network data release. IEEE Access **7**, 164304–164319 (2019)
12. Zhu, T., Li, G., Zhou, W., Yu, P.S.: Preliminary of differential privacy. In: Differential Privacy and Applications. Advances in Information Security, vol. 69, pp. 7–16. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62004-6
13. Day, W.Y., Li, N., Lyu, M.: Publishing graph degree distribution with node differential privacy. In: Proceedings of the 2016 International Conference on Management of Data, pp. 123–138 (2016)
14. Karwa, V., Slavković, A.B.: Differentially private graphical degree sequences and synthetic graphs. In: Domingo-Ferrer, J., Tinnirello, I. (eds.) PSD 2012. LNCS, vol. 7556, pp. 273–285. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33627-0_21
15. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. In: 26th USENIX Security Symposium (USENIX Security 2017), pp. 729–745 (2017)
16. Leskovec, J., Krevl, A.: Snap datasets: Stanford large network dataset collection (2014)
17. Luce, R.D., Perry, A.D.: A method of matrix analysis of group structure. Psychometrika **14**(2), 95–116 (1949). https://doi.org/10.1007/BF02289146
18. Rousseau, R., Egghe, L., Guns, R.: Becoming Metric-Wise. A Bibliometric Guide for Researchers. Chandos-Elsevier, Kidlington (2018)
19. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971)
20. Pinar, A., Seshadhri, C., Kolda, T.G.: The similarity between stochastic Kronecker and Chung-Lu graph models. In: Proceedings of the 2012 SIAM International Conference on Data Mining, pp. 1071–1082. SIAM (2012)