Anonymization for Multiple Released Social Network Graphs

Chih-Jui Lin Wang[1], En Tzu Wang[2], and Arbee L.P. Chen[3,*]

[1] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.
s9862588@m98.nthu.edu.tw
[2] Cloud Computing Center for Mobile Applications, Industrial Technology Research Institute, Hsinchu, Taiwan
m9221009@em92.ndhu.edu.tw
[3] Department of Computer Science, National Chengchi University, Taipei, Taiwan
alpchen@cs.nccu.edu.tw

Abstract. Recently, people share their information via social platforms such as Facebook and Twitter in their daily life. Social networks on the Internet can be regarded as a microcosm of the real world and worth being analyzed. Since the data in social networks can be private and sensitive, privacy preservation in social networks has been a focused study. Previous works develop anonymization methods for a single social network represented by a single graph, which are not enough for the analysis on the evolution of the social network. In this paper, we study the privacy preserving problem considering the evolution of a social network. A time-series of social network graphs representing the evolution of the corresponding social network are anonymized to a sequence of sanitized graphs to be released for further analysis. We point out that naively applying the existing approaches to each time-series graph will break the privacy purposes, and propose an effective anonymization method extended from an existing approach, which takes into account the effect of time for releasing multiple anonymized graphs at one time. We use two real datasets to test our method and the experiment results demonstrate that our method is very effective in terms of data utility for query answering.

Keywords: social network, privacy, anonymization, time-serial data.

1 Introduction

Recently, people share their information and participate in various activities on the Internet via social platforms such as Google+, Facebook, and Twitter. The data in social networks are worth being analyzed in social science research since they can reflect the real social activities. Since the data in social networks include personal information and interactions, which can be private and sensitive, there is a need to anonymize the data to protect users' privacy before their release for some analysis.

A social network can be represented by a graph consisting of nodes and edges between the nodes. The nodes are used to represent users while the edges represent

* Corresponding author.

J. Pei et al. (Eds.): PAKDD 2013, Part II, LNAI 7819, pp. 99–110, 2013.

the interactions between the users. A trivial method for protecting users' privacy in the released graph is to replace the identities of users (e.g. ID or user name) by random values. This is not enough since the privacy may still be revealed by *malicious attackers* [2]. Achieving complete privacy protection and at the same time offering high utility of the social network data are challenging [4][8][12]. The existing approaches on anonymizing a social network graph are categorizing into two types including nodes with attributes [1][3] and without attributes [6][10][11].

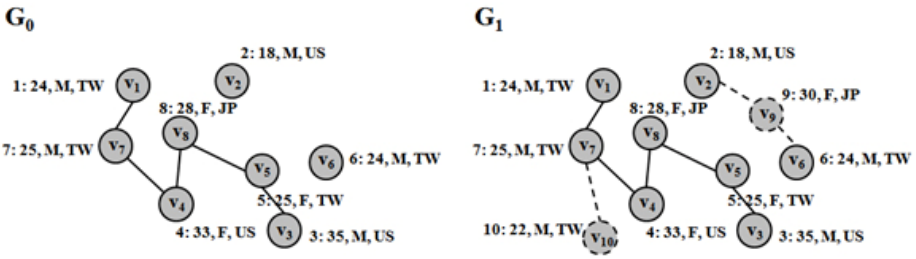
Consider a scenario as follows. A service provider, e.g. Facebook, has social network data since 2004 and wants to anonymize and release the corresponding social network graph from 2004 to 2006 to a data mining company to find interesting patterns from the evolution of the social network. Since the previous studies focus on anonymizing a single social network graph, we may consider applying the existing approaches to anonymize the social network graphs at different timestamps by considering the graph at each timestamp as a specific social network graph. However, to be detailed in Subsection 2.2, this may make the privacy of the social network data leaked.

In this paper, we consider a privacy preserving problem on a time-series of social network graphs representing an evolution of a specific social network. A time-series of social network graphs are anonymized to a sequence of sanitized graphs to be released (called *multiple releases* in this paper). Based on [1], which groups nodes into classes for achieving the privacy purposes, we design a constraint in the grouping procedure, taking into account the effect of time to avoid revealing privacy with multiple releases. Rooted in this constraint, we propose an overall method to anonymize all these time-series social network graphs at distinct timestamps at one time. We use two real datasets to test our method and the experiment results demonstrate that our method is very effective in terms of data utility for query answering.

The remainder of this paper is organized as follows. The preliminaries of this paper are introduced in Section 2. We describe the details on anonymizing a single social network graph proposed in [1] and point out that if we apply it to deal with the time-series social network graphs, the privacy guarantees may fail. Our approach is detailed in Section 3 and after that, the experiment results are presented in Section 4. Finally, Section 5 concludes this work

2 Preliminaries

We consider a privacy preserving problem on time-series social network graphs in this paper. A time-series of social network graphs denoted $g = <G_0, G_1, \ldots, G_T>$ represent the evolution of a specific social network. The social network graph at time t is denoted $G_t = (V_t, E_t, L_t)$, where V_t is a set of vertices representing users at time t, E_t is a set of edges representing the interaction among users at time t, and L_t is a set of labels, each of which is used to descript a specific user. We assume that g is *incremental*, i.e. the vertices and edges are only added to but not deleted from the graph in the next timestamp. Accordingly, $V_{t-1} \subseteq V_t$, $E_{t-1} \subseteq E_t$, and $L_{t-1} \subseteq L_t$, where t = 1 to T. Obviously, G_T is equal to the union of $G_0, G_1, \ldots, G_T$. Figure 1 shows two snapshots of a time-serial social network graph at $t = 0$ and $t = 1$. The corresponding vertex of a user is associated with a label and each label has a number of attributes, e.g. age, gender, and location, to descript the user.



(*a*) $t = 0$ (*b*) $t = 1$

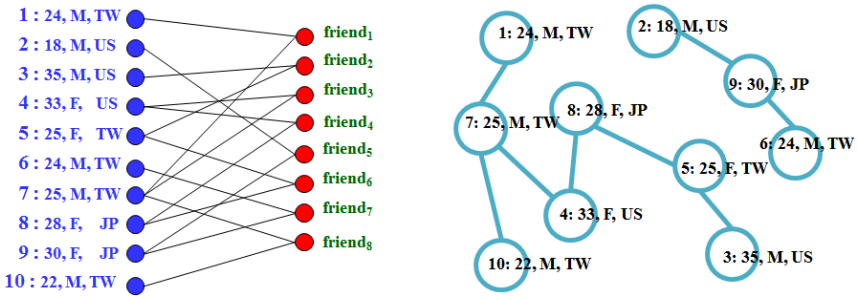
Fig. 1. Snapshots of a time-series of social network graphs at $t = 0$ and $t = 1$

Problem Statement. Given a time-series of social network graphs $g = <G_0, G_1, \ldots, G_T>$ and a constant k, we want to release the anonymized graphs $<G_0', G_1', \ldots, G_T'>$, each of which should follow the guarantees below:

1. For any edge e in an anonymized graph, an attacker who has no background knowledge about the original graph can correctly guess that a specific user u participates in e with a probability at most equal to $1/k$.
2. For any two users u_x and u_y, an attacker who has no background knowledge about the original graph can correctly guess that these users have interaction with a probability at most equal to $1/k$.

2.1 Single Graph Anonymization

Distinct from [1], only one type of interaction is considered in this paper. Then, the bipartite graph to represent the interaction among users in [1] as shown in Figure 2 (a) is easily transformed into a general social network graph as shown in Figure 2 (b).



(*a*) The interaction graph (one type) (*b*) The general graph

Fig. 2. The transformation of an interaction graph

In order to achieve the above privacy objectives, a *label list* denoted $l(v)$ is used in [1] to replace the true label of a node v and moreover, the true label of v must be contained in $l(v)$. Label lists are generated by dividing all nodes into classes with a size equal to k. Nodes in the same class have the same label list. Accordingly, an

attacker has a probability of $1/k$ to guess the correct label of a node. For example, suppose $k = 2$. The nodes shown in Figure 2(b) are divided into classes with a size equal to 2. We then get five classes including $A = \{1, 2\}$, $B = \{3, 4\}$, $C = \{5, 6\}$, $D = \{7, 8\}$, and $E = \{9, 10\}$. Then, each node is assigned a label list according to its corresponding class. Figure 3 shows the anonymized graph.

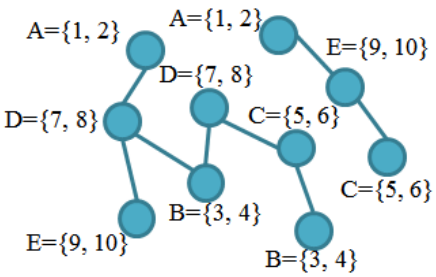


Fig. 3. The full list anonymized graph at $k = 2$

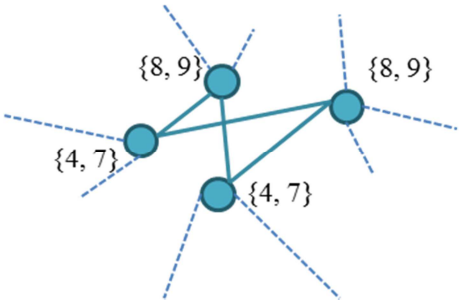


Fig. 4. An example of the dense links between two classes at $k = 2$

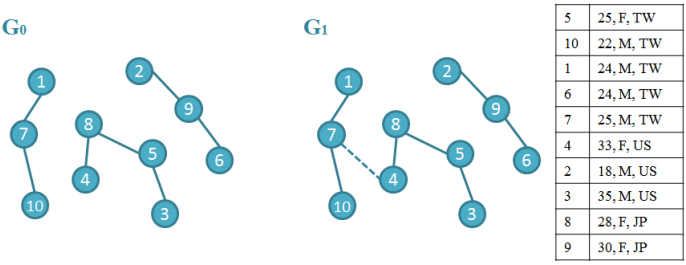
Merely partitioning the nodes into classes cannot achieve the privacy objectives. If the links among nodes in a same class are dense, an attacker can imply with a high probability that a certain link must exist. For example, suppose user 1 and user 7 in Figure 2(b) are grouped in the same class at $k = 2$. Then, an attacker can be sure that user 1 and user 7 have interaction, without recognizing the true labels related to the nodes. Moreover, the links between two classes should not be dense. For example, the interaction between two classes is dense as shown in Figure 4, and an attacker can exactly know user 4 and user 7 have the same friends including user 8 and user 9. Bhagat et al. propose the *class safety condition* in [1], defined as follows, to avoid the above attacks.

Definition 1. *Class Safety Condition [1]: Division of nodes V into classes satisfies the Class Safety Condition if any node* $v \in V$ *and any class* $C \subset V$ *follow 1)* $\forall(v, w)$ *and* $(v, z) \in E$*: if* $w \in C \wedge z \in C \Rightarrow w = z$ *and 2)* $\forall(v, w) \in E$*: if* $v \in C \wedge w \in C \Rightarrow v = w$.

A simple greedy approach for partitioning nodes into classes is proposed in [1]. To improve the utility of the anonymized graph, they consider sorting the attributes of nodes according to their importance to queries and then follow the sorted *attribute priority list* in the division procedure to make sure that nodes with similar attributes are divided into the same or nearby classes under the condition of observing the class safety condition. Following [1], we assume that the attribute priority list is given.

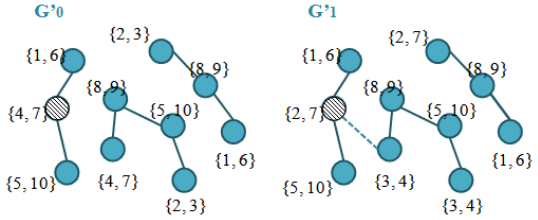
2.2 Privacy Revealed across Multiple Releases

To solve the problem addressed in this paper, a naïve solution based on [1] is descripted as follows.



5	25, F, TW
10	22, M, TW
1	24, M, TW
6	24, M, TW
7	25, M, TW
4	33, F, US
2	18, M, US
3	35, M, US
8	28, F, JP
9	30, F, JP

(*a*) The original graphs G_0, G_1.



5	25, F, TW
10	22, M, TW
1	24, M, TW
6	24, M, TW
7	25, M, TW
4	33, F, US
2	18, M, US
3	35, M, US
8	28, F, JP
9	30, F, JP

(*b*) The released graphs G'_0, G'_1.

Fig. 5. An example of privacy revealed across multiple releases at $k = 2$

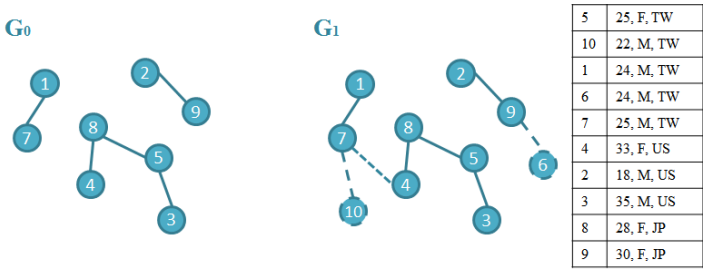
Naïve Solution. G_0, G_1, …, G_T are individually anonymized using the approach proposed in [1]. Since the classes may be different at distinct timestamp, the label list of v may change at distinct timestamp.

For example, given two snapshots of a time-serial social network graph, G_0 and G_1, at $t = 0$ and 1 respectively, a constant $k = 2$, and the sorted nodes according to the attribute priority list <location, gender, age> as input of the approach proposed in [1], G_0' and G_1' are generated as shown in Figure 5(b). Let us focus on the grey node v in G_0' and G_1', the label lists of v at two timestamps are $l_0(v) = \{4, 7\}$ and $l_1(v) = \{2, 7\}$, respectively. Since we assume the evolution of a specific social network graph is incremental and only the grey node links both to {1, 6} and {5, 10} in G_0' and G_1', the true identity of v must be 7. Obviously, it violates the privacy purposes.

Observation 1. Given a time-series of social network graphs $g = <G_0, G_1, \ldots, G_T>$, a constant k, and an attribute priority list denoted $list_{ap}$, we first anonymize G_T to generate G_T' and in the next step, we generate G_{t-1}' by removing the edges and nodes arriving at t from G_t' for $t = 1$ to T at each iteration. In this case, the privacy may be revealed when the nodes are removed since the size of the label list containing the removed entities may be less than k.

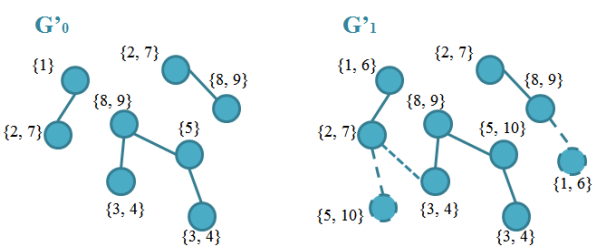
For example, given two snapshots of a time-serial social network graph, G_0 and G_1, at $t = 0$ and 1 respectively, a constant $k = 2$, and the sorted nodes according to the attribute priority list <location, gender, age> as input shown in Figure 6(a), we first anonymize G_1 to generate G_1' using the approach in [1] as shown in Figure 6(*b*). Next,

G_0' is generated by deleting the edges and nodes arriving at t = 1 from G_1'; furthermore, the corresponding labels of the removed nodes should be deleted from the label lists as shown in Figure 6(*b*). Obviously, the identity of node e.g. user 1, is revealed. To solve the problem of revealing the identities of nodes, we may put the nodes arriving at the same timestamp into the same classes. Therefore, the nodes arriving at the same time and their corresponding classes will be deleted together. The main idea of our solution extended from Observation 1 is detailed in the following section.



5	25, F, TW
10	22, M, TW
1	24, M, TW
6	24, M, TW
7	25, M, TW
4	33, F, US
2	18, M, US
3	35, M, US
8	28, F, JP
9	30, F, JP

(*a*) The original graphs G_0 and G_1.



5	25, F, TW
10	22, M, TW
1	24, M, TW
6	24, M, TW
7	25, M, TW
4	33, F, US
2	18, M, US
3	35, M, US
8	28, F, JP
9	30, F, JP

(*b*) The anonymized graphs G'_0 and G'_1.

Fig. 6. An example of privacy revealed in Observation 1 at $k = 2$

3 The Anonymizing Method

Our solution to anonymizing a time-series of social network graphs is detailed in this section. The definition of the *Time-Series Class Safety Condition* (TSCSC) used in our solution is described in Subsection 3.1 and then, we present the main algorithm in Subsection 3.2 and discuss the security of our solution in Subsection 3.3.

3.1 Time-Series Class Safety Condition

We modify the class safety condition in Definition 1 for the time-series social network graphs to obtain the *Time-Series Class Safety Condition* (TSCSC). Dividing nodes into classes that satisfy TSCSC can achieve the privacy objectives mentioned. How TSCSC to guarantee the privacy objectives will be discussed later.

Definition 2. *Time-Series Class Safety Condition: Division of nodes V_t into classes satisfies the Time-Series Class Safety Condition if any node $v \in V_t$ and any class $C \subset V_t$ follow 1) $\forall v \in V_t$ and $w \in V_{t'}$: if $v \in C \wedge w \in C \Rightarrow t = t'$, 2) $\forall (v, w) \in E_t$: if $v \in C \wedge w \in C \Rightarrow v = w$, and 3) $\forall C_a$ and $C_b \subset V_t$, n_e is the number of edges between C_a and $C_b \Rightarrow n_e \leq k$.*

The first condition indicates that the nodes in a same class arrive at the same time. The second condition similar to Definition1 constraints that at each timestamp, no edges exist in a class. The third condition constraints the number of interaction between any pairs of classes at each timestamp.

3.2 The Anonymizing Method for Time-Series Social Network Graphs

Our greedy algorithm name *DMRA* (***D**ecreasing **M**ultiple **R**eleases **A**nonymization*) for anonymizing the time-series social network graphs is descripted in this subsection. Given a sequence of time-series social network graphs $g = <G_0, G_1, \ldots, G_T>$, a constant k, and an attribute priority list denoted $list_{ap}$:

Step 1. We sort all vertices in G_T according to $list_{ap}$ to generate an order list of vertices, V_{list}, in which the vertices with similar attributes are nearby. Then, we start to anonymize G_T.

Step 2. Initially, no class exists; we thus create a new class only containing the first vertex in V_{list}. Then, for each $v \in V_{list}$, we sequentially insert the node v into the first fit class which contains the nodes with a number smaller than k and moreover, the insertion must satisfy the Time-Series Class Safety Condition. If we cannot find a class with a size smaller than k or observing TSCSC after considering v, a new class is created.

Step 3. After Step 2, some classes may not have k nodes. To reduce the number of classes with a size smaller than k, the classes need refinement. Since this is a heuristic method, it is possible that partitioning nodes into classes fails while interaction is dense. However, the social network usually follows *Power Law Distribution*, making many nodes with low degrees. This can be effectively used to reduce the number of classes with a size smaller than k, also mentioned [1]. For each class with a size smaller than k, we check whether it can be merged with another class with a size smaller than k (observing TSCSC), if yes, we merge the two classes to reduce the number of classes with a size smaller than k.

Step 4. Now, we add *dummy nodes* to those classes still with a size smaller than k. The attributes of a dummy node are decided by randomly picking from the attributes of the nodes of the class which it belongs to. For example, suppose Class A = {2, 7} and k = 3, user 2 is with attributes = {18, M, US} and user 7 is with attributes = {25, M, TW}. We add a dummy node corresponding to user 11 to Class A and its attributes will be either {18, M, US} or {25, M, TW}. Finally, each class has k nodes and we assign the corresponding label list to each node according to its class to get G_T'.

Step 5. After Step 4, G_T' is generated. Then, we can generate G_{t-1}' by removing all vertices $\in V_t \setminus V_{t-1}$ and all edges $\in E_t \setminus E_{t-1}$ from G_t' for t = 1 to T.

As shown in Figure 7, we sort all vertices in G_1 according to $list_{ap}$ = <location, gender, age> to generate V_{list}. Following Steps 2 and 3, the nodes are divided into A = {5, 8}, B = {1, 10}, C = {6, 7}, D = {3, 4}, E = {2}, and F = {9}. Since Classes E and F are both with a size smaller than k, dummy nodes are added to them. E becomes {2, 11} and F becomes {9, 12}. Next, we assign the corresponding label list to each node according to its class to get G_1'. Finally, we generate G_0' by removing the vertices and edges belong to $V_1 \setminus V_0$ and $E_1 \setminus E_0$ from G_1', respectively.

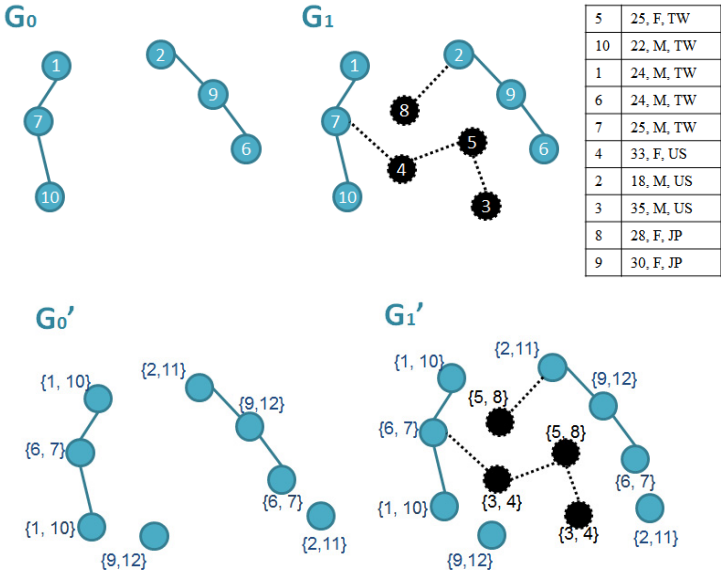


Fig. 7. An illustration of the running DMRA at k = 2

3.3 The Security of Time-Series Class Safety Condition

Three conditions for ensuring the privacy objectives of our method on multiple releases for time-series social network graphs are descripted. The first condition ($\forall\ v \in V_t$ *and* $w \in V_{t'}$: *if* $v \in C \wedge w \in C \Rightarrow t = t'$) indicates that nodes in the same class arrive at the same time in each anonymized graph. As a result, DMRA ensures that the deleted nodes are in the same classes, thus making each class to have k members.

The second condition ($\forall(v, w) \in E_t$: *if* $v \in C \wedge w \in C \Rightarrow v = w$) constraints no edges exist in a class at a timestamp. Accordingly, if there is an edge between two nodes in the anonymized graph at timestamp t, the true labels of the two nodes must belong to different classes. Then, to an edge, there will be k candidate labels for both endpoints.

The third condition ($\forall\ C_a$ and $C_b \subset V_t$, n_e is the number of edges between C_a and $C_b \Rightarrow n_e \leq k$) ensures the number of interaction between any pairs of classes at each timestamp to less than or equal to k. Given two classes C_a and C_b at the same timestamp, suppose that an entity u_x is in C_a and an entity u_y is in C_b, the probability

of guessing these two entities u_x and u_y having interaction can be formulated as following:

$$p\left(e\left(u_x, u_y\right)\right) = \frac{(|C_a|-1)! \times (|C_b|-1)! \times n_e}{|C_a|! \times |C_b|!} \quad (1)$$

Since $|C_a|$ and $|C_b|$ are both equal to k and $p(e(u_x, u_y))$ should be less than or equal to $1/k$, we can imply $n_e \leq k$.

4 Experiments

How to evaluate the utility of the anonymized social network graphs is an important issue. Some researchers [1] [9] [11] conduct aggregation queries on the anonymized social network graphs. In this paper, we use two kinds of queries including the *single hop queries* and *two hops queries* (also used in [1] [9] [11]) to evaluate the utility of the anonymized time-series graphs The formal description of the single hop queries is as follows: How much interaction between the user with one specific attribute and another user with another specific attribute at a time period. For example, how much friendship between the users located in United States and the users located in Japan at the measurement period? The two hops queries involve three user attributes. For example, how much friendship satisfying that Americans have friendship with Japanese, who also have friendship with Chinese at the measurement period? We measure the utility using *average relative error* [1] [9] [11]). The relative error is equal to $|d - d'| / |d|$, where d and d' are the results of querying on the original graphs and on the anonymized graphs, respectively. Since we do not know the true label of each node in the anonymized graphs, how to perform the queries on the anonymized graphs is an issue. We use the Sampling Consistent Graphs method [1] to randomly sample a graph that is consistent with the anonymized graph. The query is performed on the sampled graph.

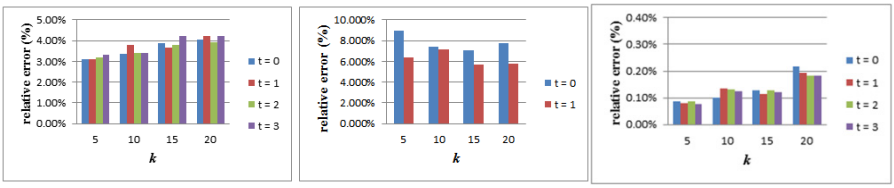
Table 1. The Flickr dataset

t	Timestamp	Nodes	Edges	Nodes added	Edges added
0	Dec 3 '06	1,277,145	6,042,807	1,277,145	6,042,807
1	Mar 3 '07	1,572,674,	8,374,733	295,529	2,331,926
2	Apr 3 '07	1,712,227	9,166,282	139,553	791,549
3	May 18 '07	1,856,342	10,301,741	144,115	1,135,459
Total		1,856,342	10,301,741	1,856,342	10,301,741

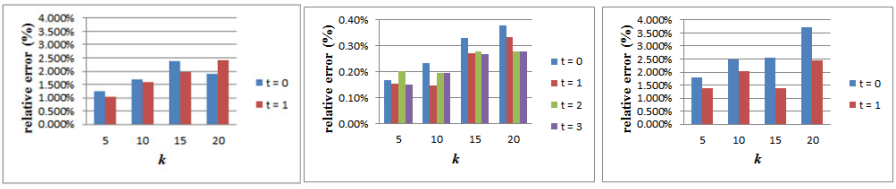
Table 2. The Slashdot dataset

t	Timestamp	Nodes	Edges	Nodes added	Edges added
0	Nov 6 '08	70,668	358,981	70,668	358,981
1	Feb 1 '09	79,940	723,428	9,272	364,447
Total		79,940	723,428	79,940	723,428

Two real datasets Flickr [7] and Slashdot [5] with synthetic labels are used to test our solution. Flickr was daily crawled from the Flickr network between November 2nd, 2006 and December 3rd, 2006, and again between February 3rd, 2007 and May 18th, 2007. This dataset has a total number of nodes and edges about 1.8M and 10M, respectively. We separate the dataset into four partitions to simulate different timestamps. Slashdot is a technology-related news website. The dataset was collected in November 6th, 2008 and February 1st, 2009, which consists of 79K nodes and 723K edges. We separate this dataset into two partitions to simulate two timestamps. Because the original datasets have no labels, we generate labels containing three attributes, age (10~60), gender (male/female) and location (50 countries) for each entity and all values are within the uniform distribution. Our algorithm is implemented in C++ and performed on a PC with the Intel Core 2 Quad 2.66GHz CPU, 8GB memory, and under the Ubuntu v11.04 64bits operating system.

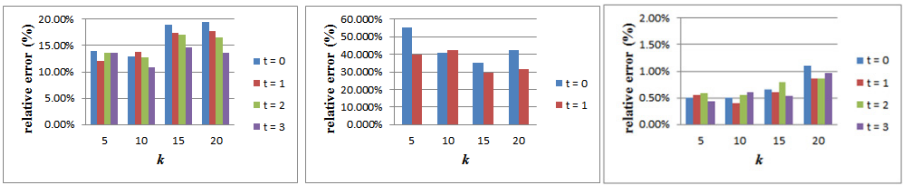


(*a*) Flickr dataset (unsorted) (*b*) Slashdot dataset (unsorted) (*c*) Flickr dataset (ALG)

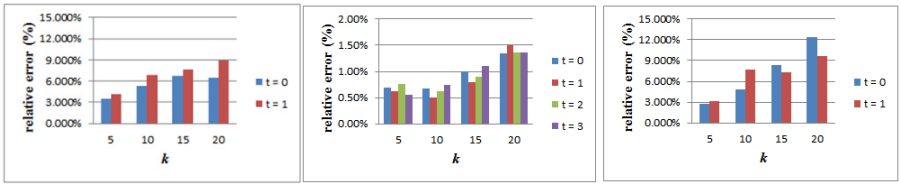


(*d*) Slashdot dataset (ALG) (*e*) Flickr dataset (GLA) (*f*) Slashdot dataset (GLA)

Fig. 8. The average relative errors on single hop queries



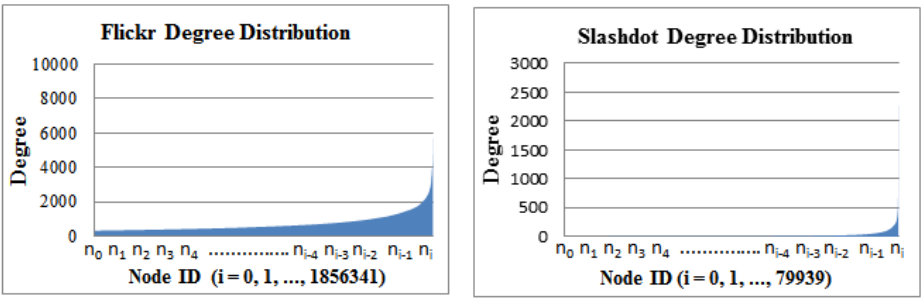
(*a*) Flickr dataset (unsorted) (*b*) Slashdot dataset (unsorted) (*c*) Flickr dataset (ALG)



(*d*) Slashdot dataset (ALG) (*e*) Flickr dataset (GLA) (*f*) Slashdot dataset (GLA)

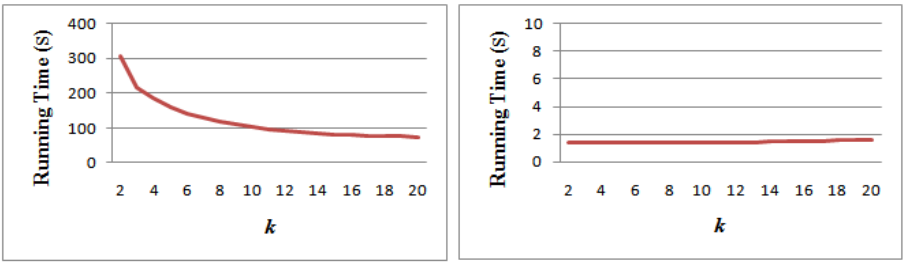
Fig. 9. The average relative errors on two hops queries

We consider 50 queries and three sorting methods including unsorted, $list_{ap}$ = <**A**ge, **L**ocation, **G**ender>, and $list_{ap}$ = <**G**ender, **L**ocation, **A**ge> in the experiments. The experiment results regarding single hop queries and two hops queries are shown in Figures 8 and 9, respectively. Obviously, since the larger k leads to the more possible labels for each entity, the average relative error rate increases as the increasing of k. The benefits of sorting vertices before partitioning them into classes are shown in Figures 8 and 9. As can be seen, the unsorted cases have the higher average relative error rates in either Figures 8 or 9. Usually, the results on single hop queries are better than those on two hops queries since the two hops queries involve much interaction. The degree distributions of the two datasets are shown in Figure 10, which indicate that the degree distributions of the two datasets follow the power law distribution. Accordingly, most of the nodes are quickly grouped together since there are many nodes with low degrees in the datasets and therefore, we only need to add few dummy nodes in most of the cases.



(*a*) The distribution of Flickr (*b*) The distribution of Slashdot

Fig. 10. The degree distributions of the two datasets



(*a*) Flickr dataset (AGL) (*b*) Slashdot dataset (AGL)

Fig. 11. The running time of DMRA

The running time of DMRA is shown in Figure 11, which decreases with the increasing of k, since the number of classes will decrease with the increasing of k, making the comparisons on checking *Time-Series Class Safety Condition* reduced. Either single hop queries or two hops queries only consider the interaction among entities with different user attributes. Since dummy nodes do not have any edges in the released graphs, adding dummy nodes does not change the total number of interactions and does not seriously affect the results of queries.

5 Conclusions

In this paper, we address a new problem on privacy preserving for releasing multiple time-series social network graphs. Naively applying the existing approach to each time-series graph will break the privacy purposes. For achieving the privacy purposes, we propose Time-Series Class Safety Condition and DMRA for releasing multiple anonymized graphs at one time. The experiments demonstrate that DMRA is very effective in terms of data utility. Moreover, if we know which attributes are more important and often used in the queries in advance and follow the sorted vertex ordering in our anonymizing algorithm, the relative error rate will be reduced.

References

1. Bhagat, S., Cormode, G., Krishnamurthy, B., Srivastava, D.: Class-Based Graph Anonymization for Social Network Data. In: Proceedings of the 35th International Conference on Very Large Data Base, pp. 766–777 (2009)
2. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore Art Thou R3579X*?* Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In: Proceedings of the 16th International Conference on World Wide Web, pp. 181–190 (2007)
3. Cormode, G., Srivastava, D., Yu, T., Zhang, Q.: Anonymizing Bipartite Graph Data Using Safe Groupings. In: Proceedings of the 34th International Conference on Very Large Data Base, pp. 833–844 (2008)
4. Liu, K., Das, K., Grandison, T., Kargupta, H.: Privacy-Preserving Data Analysis on Graphs and Social Networks. In: Kargupta, H., Han, J., Yu, P., Motwani, R., Kumar, V. (eds.) Next Generation Data Mining, ch. 21, pp. 419–437. CRC Press (December 2008)
5. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.: Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. University of Massachusetts Technical Report, Internet Mathematics 6(1), 29–123 (2009)
6. Liu, K., Terzi, E.: Towards Identity Anonymization on Graphs. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 93–106 (2008)
7. Mislove, A., Koppula, H.S., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Growth of the Flickr Social Network. In: Proceedings of the 1th ACM SIGCOMM Workshop on Online Social Networks, pp. 25–30 (2008)
8. Wu, X., Ying, X., Liu, K., Chen, L.: A Survey of Privacy-Preservation of Graphs and Social Networks. In: Aggarwal, C.C., Wang, H. (eds.) Managing and Mining Graph Data, vol. 40, pp. 421–453. Springer US (2010)
9. Yuan, M., Chen, L., Yu, P.: Personalized Privacy Protection in Social Networks. In: Proceedings of the 37th International Conference on Very Large Data Base, pp. 141–150 (2010)
10. Zou, L., Chen, L., Ozsu, M.T.: K-automorphism: A General Framework for Privacy Preserving Network Publication. In: Proceedings of the 35th International Conference on Very Large Data Base, pp. 946–957 (2009)
11. Zhou, B., Pei, J.: Preserving Privacy in Social Networks against Neighborhood Attacks. In: Proceedings of the 24th IEEE International Conference on Data Engineering, pp. 506–515 (2008)
12. Zhou, B., Pei, J., Luk, W.-S.: A Brief Survey on Anonymization Techniques for Privacy Preserving Publishing of Social Network Data. In: Proceedings of the SIGKDD Explorations, pp. 12–22 (2008)