Yuan M, Chen L, Yu PS *et al.* Protect you more than blank: Anti-learning sensitive user information in the social networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 29(5): 762–776 Sept. 2014. DOI 10.1007/s11390-014-1466-1

Protect You More Than Blank: Anti-Learning Sensitive User Information in the Social Networks

Mingxuan Yuan[1,2] (袁明轩), Lei Chen[1] (陈　雷), *Member*, *IEEE*, Philip S. Yu[3], *Fellow*, *ACM*, *IEEE* and Hong Mei[4,5,6] (梅　宏), *Fellow*, *CCF*, *Member*, *ACM*, *IEEE*

[1] *Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong China*

[2] *Huawei Noah's Ark Laboratory, Shatin, Hong Kong, China*

[3] *Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60637, U.S.A.*

[4] *Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China*

[5] *Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China*

[6] *Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*

E-mail: yuan.mingxuan@huawei.com; leichen@cse.ust.hk; psyu@cs.uic.edu; meih@pku.edu.cn

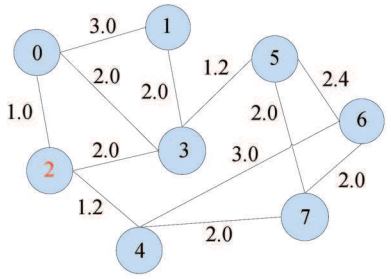
Received April 7, 2014; revised July 4, 2014.

Abstract Social networks are getting more and more attention in recent years. People join social networks to share their information with others. However, due to the different cultures and backgrounds, people have different requirements on what kind of information should be published. Currently, when social network websites publish data, they just leave the information that a user feels sensitive blank. This is not enough due to the existence of the label-structure relationship. A group of analyzing algorithms can be used to learn the blank information with high accuracy. In this paper, we propose a personalized model to protect private information in social networks. Specifically, we break the label-structure association by slightly changing the edges in some users' neighborhoods. More importantly, in order to increase the usability of the published graph, we also preserve the influence value of each user during the privacy protection. We verify the effectiveness of our methods through extensive experiments. The results show that the proposed methods can protect sensitive labels against learning algorithms and at the same time, preserve certain graph utilities.

Keywords social network, privacy, protection

1 Introduction

As a kind of Web 2.0 killer application, social network websites enable users to share their information when joining the network. The social network is often modeled as a weighted graph where nodes represent users and edges represent the relationships among users[1-2]. Fig.1 shows a subgraph we got from Face-



ID	Name	Hobby	Purpose
0	Harry	Cooking	Friendship
1	Michelle	Cooking	Relationship
2	Ben	*	Friendship
3	Bily	Cooking	Networking
4	Aron	Writing	Dating
5	Sam	Writing	Relationship
6	San	Writing	Friendship
7	Tony	Writing	Dating

Fig.1. Social network with blank label.

Regular Paper

This work is supported in part by the Research Grants Council (RGC) of Hong Kong, China, under Grant No. NHKUST612/09, the National Basic Research 973 Program of China under Grant No. 2012CB316200, and the National Natural Science Foundation of China under Grant No. 60931160444.

book. The weight between any two users represents the frequency of posting on each other's wall. The frequency is normalized to a number in [0, 10].

When a user registers an account on a social network, e.g., Facebook, he/she is often requested to fill in some personal information to create an online profile, including hobby, high school name, age, and so on. We call each input item as a *label* or an *attribute* of the user. For example, in Fig.1, each user has two labels, his/her hobby and the purpose of joining the network. Due to the background and culture diversities, users have different concerns to sensitive labels. For a label, some users may not mind others to acquire. Moreover, some of them even want more people to know them by publishing the label. These are *open users* in the network. For example, in Fig.1, Harry, Michelle *et al.*, would like to share their hobby information with friends. However, some users treat this label as sensitive information, and do not want others to know. We call these users as *conservative users*. For these conservative users, the websites normally leave their sensitive information blank to satisfy the privacy requirement. Ben in Fig.1, for instance, is a very sensitive person and wants to hide his own hobby information. As a result, we only obtained a graph as shown in Fig.1, in which Ben's hobby is published as blank.

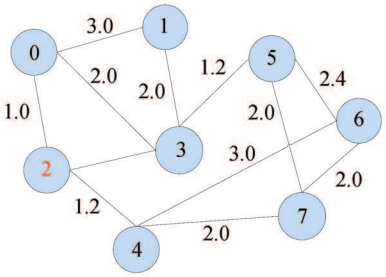
However, simply leaving sensitive labels blank is not enough to protect users' privacy. A group of learning algorithms are introduced in [3-4] to deduce the missing labels in the social networks with high accuracy. The attacks through links, groups, and their combinations could detect the sensitive labels without any notice to users. These algorithms use the fact that users often have common social actions in a network, that is, individuals who have similar labels tend to form a community. For example, although Ben in Fig.1 does not want to publish his hobby, we could guess it as cooking since two friends of Ben (Harry and Bily) like cooking and only one friend (Aron) likes writing①. If Ben indeed likes cooking, then his privacy is violated even when the website protects him by leaving his hobby blank. Therefore, to prevent a user's sensitive labels from being learned, we should break the association between his label and his neighborhood labels. In other words, we want to publish the social graph which does not disclose sensitive labels when learning algorithms are being applied. (It should be noted that the problem cannot be simply solved by replacing the blank attribute with a random label. When the associations between label and neighborhood are not broken, they still can be used to filter out the randomly filled labels.)

We call attacks using learning algorithms[3-7] as "label-structure attack". The current protection models focus on preventing node re-identification[8-13]. The *re-identification* means matching a node in the graph with a user based on the users' information (i.e., certain structure information such as node degree and user's nonsensitive labels). This attack is also called "structure attack". An attacker could know all the information around a node by re-identifying a user. To prevent node re-identification, an anonymized graph must be published according to the assumption about what information an attacker may use to re-identify a user. Therefore, each user is totally hidden by anonymization, which leads the information that users want to share also unusable. However, social network websites publish the network data (through API) to the third parties who can get some knowledge about the graph in order to provide more interesting services to the target users. Without the information from the hidden users, it is difficult to achieve such a goal. Therefore, it is necessary to design a new model to protect sensitive information against being attacked by a learning algorithm and meanwhile to publish the useful information for sharing.

In this paper, we propose a utility-oriented random method to break the label-neighborhood association. For open users, their real labels are published without any changes. For each conservative user, we modify his/her neighborhood by randomly changing $p\%$ edges which connect this user to other users with the same label. $p\%$ is a constant set by each user or the data publisher. For example, Ben in Fig.1 is a conservative user and his $p = 50\%$. Through our method, we can delete the edge between 2 and 3. Then, an attacker has no idea about Ben's label from his neighborhood (see Fig.2). Since the changed edges are randomly selected, the constant $p\%$ of each conservative user can be published to the attacker as well.

In order to increase the utility of the published graph, we also preserve important characteristics of social network, "the influential values" of nodes. As a common graph property, influential value is widely used in different applications, such as expert finding, the most valuable people searching, and so on[14-15]. The value represents how important a user is in the whole network or to his/her neighbors. If each user has the same influential value in the published graph as that in the original graph, he/she has the same "position" in the two graphs. Therefore, the influential value should be preserved in order to keep a graph as useful as before. We achieve the influential value preserving objec-

①We can draw the same conclusion when considering the edge weights.



ID	Name	Hobby	Purpose
0	Harry	Cooking	Friendship
1	Michelle	Cooking	Relationship
2	Ben	*	Friendship
3	Bily	Cooking	Networking
4	Aron	Writing	Dating
5	Sam	Writing	Relationship
6	San	Writing	Friendship
7	Tony	Writing	Dating

Fig.2. Edge editing to change Ben's neighborhood.

tive by random edge editing. We make sure the random edge editing for utility preserving does not change the random effectiveness of the privacy protection. An interesting result we observed is that through preserving the influential values, the maximum eigenvalue of the graph's adjacent matrix is also preserved. Eigenvalues have been shown as intimately connected to many important topological features in paper [16]. Fig.3 is an influential preserved graph of Fig.1 in case that Ben wants to protect his label. If using the propagation-based expert computation model② in [15, 17] to analyze Fig.1 and Fig.3, the corresponding nodes in these two figures have the same influential value.

To summarize, the differences of our work and the work to prevent the node re-identification in social networks are:

• We publish a graph with all the information that a user wants to share. The graph leaves the labels that users feel sensitive blank. We protect node sensitive labels instead of the node itself. We make sure unpublished labels could not be discovered by common label-structure analysis algorithms with high confidence.

• We protect labels according to the user's personal setting. Each user has the privileges to set his/her own preference in terms of hiding the sensitive labels.

• We preserve the influential value in the published social network. When using the algorithms[15,17] to analyze the user influential values, the results obtained from the original graph and the published graph are the same.

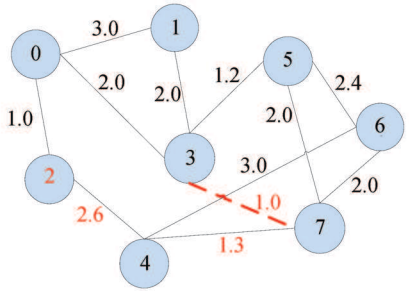
The rest of the paper is arranged as follows. Section 2 gives an overall description of the problem. Section 3 introduces the detailed designs of our algorithms. We report the experimental results on real datasets in Section 4. Finally, we compare our work with related proposals in Section 5 and conclude the paper in Section 6.

2 Problem Description

In this paper, the social network is represented as an un-directed weighted graph.

Definition 1 (Social Network Graph). *A social network graph is a six tuple $G(V, E, W, \sigma, \sigma', \lambda)$, where V is a set of nodes, and each node represents an entity in the social network. $E \subseteq V \times V$ is the set of edges between nodes. $W : E \longrightarrow R^+$ maps each edge to a positive real number. σ maps each node to a group of non-sensitive labels, where S is the sensitive label set. $\sigma' : V \longrightarrow s|s \in S$ maps each node to a sensitive label. $\lambda : V \longrightarrow$ boolean maps each node to a boolean value, indicating whether a node wants to protect its sensitive label.*

If a node wants to protect its sensitive label, λ is set as true and vice versa. We call a node which needs to hide its label as a conservative node and a node which does not care that as an open node. In the remaining



ID	Name	Hobby	Purpose
0	Harry	Cooking	Friendship
1	Michelle	Cooking	Relationship
2	Ben	*	Friendship
3	Bily	Cooking	Networking
4	Aron	Writing	Dating
5	Sam	Writing	Relationship
6	San	Writing	Friendship
7	Tony	Writing	Dating

Fig.3. Edge editing to reserve the influential values.

②Propagation-based expert computation model is a method to compute the importance/influence of nodes in a graph. It is a recursive algorithm, which considers the interactive propagation of the importance/influence between neighbors. For example, PageRank is one of the most famous algorithms.

part of this paper, we also use G or $G(V, E, W)$ to represent a graph $G(V, E, W, \sigma, \sigma', \lambda)$ for simplicity.

For a weighted graph, one important data mining task is to analyze the node influential values (i.e., the expert finding[14-15,17]). We focus on the undirected weighted graph in which weights reflect the cooperation between users. The node influential values in a weighted graph G are defined as:

Definition 2. *The influential value f_i of each node u_i in a weighted graph G is defined as*[15,17]:

$$\forall u_i \in V, ft_i = f_i + \sum_{\forall e(i,j)} w_{(i,j)} \times f_j,$$
$$T = \sum_{i=1}^{|V|} ft_i,$$
$$\forall u_i \in V, f_i = \frac{ft_i}{T}. \quad (1)$$

(1)[15,17] is a typical propagation model to compute node influential values. All f_is can be calculated when the above formula converges at the stable state. We choose this formula, which is similar to pageranking algorithm, since the pageranking algorithm is tightly related with the maximum eigenvalue of a graph. We observed that through preserving the influential values computed by (1), the maximum eigenvalue of the graph's adjacent matrix is also preserved. It has been shown that the eigenvalues have intimately connected to many important topological features in paper [16]. In the rest of this paper, we use $u.label$ and $u.influence$ to represent user u's label and influential value respectively.

The "label-structure attack" is the attack which learns the blank label of a conservative node u by analyzing its neighborhood. Since our method randomly changes the neighborhood of each conservative user, it can provide protection to any learning method using the connection information in neighborhoods. In our experiment, we show the protection effectiveness against six different learning methods.

The problem we solve in this paper is:

Problem 1. *Given a graph $G(V, E, W)$ and a constant p, generate a new graph $G'(V', E', W')$ from G that satisfies*:

- $V \equiv V'$;
- *for each conservative user, at least $p\%$*③ *edges, which connect this user to another user with the same label, are changed*;
- *the influential values of any node u in G and G' are the same.*

3 Algorithms

We generate a privacy preserving graph by edge editing. The operations include two parts:

- break the label-neighborhood association;
- preserve the influential values.

For each conservative user, we delete $p\%$ edges which connect this user to another user with the same label. After that, we continue editing several edges to preserve the influential values of all the nodes. We make sure that the second step still guarantees the $p\%$ change ratio in each conservative user's neighborhood.

3.1 Edge Editing for Association Breaking

This step is simple, for each conservative user, we randomly delete $p\%$ edges which connect this user to another user with the same label.

3.2 Edge Editing for Influential Value Preserving

After breaking the label-neighborhood relationship, some nodes' influential values are changed. We design a novel method to make up the changed influential values. When we reduce the weight on the edge $e(u, v)$ (i.e., delete an edge), if we can maintain both ft_u, ft_v and T in (1) unchanged, the influential value computation formulas still retain at the converging state. The influential values of all nodes are not changed either.

We use an adjusting method to add the changes of influential values back for each modified edge. Suppose an edge $e(u, v)$ with weight a is deleted, we randomly select an edge $e(x, y)$ with $u.label \neq x.label \wedge v.label \neq y.label$. We change the weights on the edges $e(u, x)$, $e(v, y)$ (if they do not exist, we first create these edges with weight 0) and adjust the weight on the edge $e(x, y)$. The operation is shown in Fig.4. ft_u, ft_v, ft_x and ft_y should be preserved in order to retain f_u, f_v, f_x and f_y. Suppose the weight on the edge $e(u, v)$ is reduced by a. Before reducing weight on edge e, f_u, f_v, f_x and f_y satisfy:

$$ft_u = f_u + \sum_{\forall e(u,j) \wedge j \neq v} w_e \times f_j + a \times f_v,$$
$$ft_v = f_v + \sum_{\forall e(v,j) \wedge j \neq u} w_e \times f_j + a \times f_u,$$
$$ft_x = f_x + \sum_{\forall e(x,j) \wedge j \neq y} w_e \times f_j + b \times f_y,$$
$$ft_y = f_y + \sum_{\forall e(y,j) \wedge j \neq x} w_e \times f_j + b \times f_x.$$

③In this paper, we use the same p for all conservative nodes to demonstrate our algorithms. Our algorithms can support different ps as well, which means our solution could be used as a personalized model.

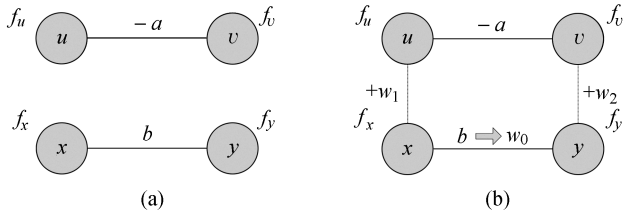


Fig.4. Adjusting after deleting case 1. (a) Original. (b) Adjusting.

After the reducing and adjusting operations, f_u, f_v, f_x and f_y should satisfy:

$$ft_u = f_u + \sum_{\forall e(u,j) \wedge j \neq v} w_e \times f_j + w_1 \times f_x,$$
$$ft_v = f_v + \sum_{\forall e(v,j) \wedge j \neq u} w_e \times f_j + w_2 \times f_y,$$
$$ft_x = f_x + \sum_{\forall e(x,j) \wedge j \neq y} w_e \times f_j + w_1 \times f_u + w_0 \times f_y,$$
$$ft_y = f_y + \sum_{\forall e(y,j) \wedge j \neq x} w_e \times f_j + w_2 \times f_v + w_0 \times f_x.$$

To keep f_u, f_v, f_x, f_y, ft_u, ft_v, ft_x, ft_y and T unchanged,

$$a \times f_v = w_1 \times f_x,$$
$$a \times f_u = w_2 \times f_y,$$
$$b \times f_y = w_1 \times f_u + w_0 \times f_y,$$
$$b \times f_x = w_2 \times f_v + w_0 \times f_x.$$

We can get the new weights:

$$w_0 = \frac{b \times f_x \times f_y - a \times f_u \times f_v}{f_x \times f_y},$$
$$w_1 = \frac{a \times f_v}{f_x},$$
$$w_2 = \frac{a \times f_u}{f_y}.$$

We can see if there is an edge $e(x, y)$ with $b \times f_x \times f_y - a \times f_u \times f_v \geqslant 0$, we can always get a solution. For example, after deleting edge $e(2, 3)$ in Fig.1, we select edge $e(4, 7)$ to retain the influential value unchanged.

In the case all $b \times f_x \times f_y - a \times f_u \times f_v < 0$, we randomly select a group of edges $\{e(x_j, y_j)|j = 0, 1, \ldots\}$ with $u.label \neq x_j.label \wedge v.label \neq y_j.label$ to preserve the influential values. For each edge $e(x_j, y_j)$ in this group, we adjust the weights on edges $e(u, x_j)$, $e(v, y_j)$, and $e(x_j, y_j)$. Fig.5 shows an example of using two edges to do the adjustment.

When using a group of edges $e(x_i, y_i)$ to make up the influential values, the new weights should satisfy:

$$a \times f_v = \sum_i w_{1,i} \times f_{x,i},$$
$$a \times f_u = \sum_i w_{2,i} \times f_{y,i},$$
$$\forall i,\ b_i \times f_{x,i} = w_{1,i} \times f_u + w_{0,i} \times f_{y,i},$$
$$\forall i,\ b_i \times f_{y,i} = w_{2,i} \times f_v + w_{0,i} \times f_{x,i}.$$

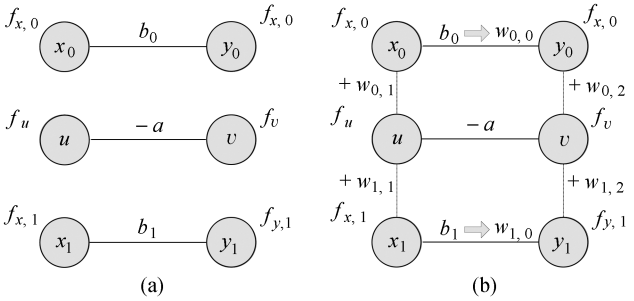


Fig.5. Adjusting after deleting case 2. (a) Original. (b) Adjusting.

One sufficient condition that the above formulas have a non-negative solution is: $\sum_i b_i \times f_{x,i} \times f_{y,i} \geqslant a \times f_u \times f_v$. For a large social network, expectively there are enough edges to hold the above condition. In case $\sum_i b_i \times f_{x,i} \times f_{y,i} < a \times f_u \times f_v$, we will compute $a' = \frac{\sum_i b_i \times f_{x,i} \times f_{y,i}}{f_u \times f_v}$ ($\forall i,\ x_i.label \neq u.label \wedge y_i.label \neq v.label \wedge x_i \neq u \wedge y_i \neq v$) and do the influential value make up by assuming the weight of $e(u, v)$ is reduced by a'. This is the best influential value preserving effect we can achieve. In our experiment, the adjustment algorithm can always find a solution to make up the changed influential values.

Since we only select edge $e(x, y)$ with $u.label \neq x.label \wedge v.label \neq y.label$ to make up the influential values, we never connect two nodes with the same label in the influential value preserving. The second step still guarantees the edge changing ratio $p\%$ in each conservative user's neighborhood.

The detailed adjusting process is shown in Algorithm 1. For each conservative user u, we firstly compute the number of edges that connect u to another user with the same label ($|\{e(u, v)|u.label \equiv v.label\}|$). Then the number of edges to delete is num_{del}. We randomly delete num_{del} edges that connect u to another user with the same label. When deleting an edge $e(u, v)$, we use the method introduced in this subsection to make up the change of influential values. We randomly select an edge with $u.label \neq x.label \wedge v.label \neq y.label \wedge u \neq x \wedge v \neq y$. If $w_{e(u,v)} \times f_u \times f_v > w_{e(x,y)} \times f_x \times f_y$, it means by deleting $e(x, y)$, the weight of $e(u, v)$ can only be decreased by $a' = \frac{w_{e(x,y)} \times f_x \times f_y}{f_u \times f_v}$. In this case, we suppose the weight of $e(u, v)$ only needs to be decreased by a'

and use (1) to adjust the weights of related edges[④]. We repeat the above process to reduce $e(u,v)$'s weight until we find an edge $e(x,y)$ with $w_{e(u,v)} \times f_u \times f_v \leqslant w_{e(x,y)} \times f_x \times f_y$. Then we can directly use (1) to delete $e(u,v)$. The whole process guarantees no user's influential value is changed. If an edge $e(x,y)$ with $u.label \neq x.label \wedge v.label \neq y.label \wedge u \neq x \wedge v \neq y$ cannot be found, it means the extreme case that the influential value changes caused by deleting $e(u,v)$ cannot be completely made up appears. The algorithm has achieved the best preserving effect and no more recovering can be made. Then, we delete $e(u,v)$, recompute the influential values of nodes in G and finish the making up process for deleting $e(u,v)$.

Algorithm 1. Random Adjusting Algorithm
Compute the influential values of each node in G;
Set $S_E = E$;
for each conservative user u in G **do**
  int $num_e = |\{e(u,v)|u.label \equiv v.label\}|$;
  int $num_{\text{del}} = \lceil num_e \times p \rceil$;
  for (int $i = 0; i < num_{\text{del}}; i++$)
    edge $e(u,v)$ = randomly select an edge with $u.label \equiv v.label$;
    $S_E = S_E - \{e(u,v)\}$;
    while $w_{e(u,v)} \times f_u \times f_v > 0$ **do**
      edge $e(x,y)$ = randomly select an edge from S_E with $u.label \neq x.label \wedge v.label \neq y.label \wedge u \neq x \wedge v \neq y$;
      if ($e(x,y) \equiv null$)
        Delete $e(u,v)$;
        Re-compute the influential values of nodes in G;
        Break;
      double $reduced = 0$;
      if ($w_{e(u,v)} \times f_u \times f_v > w_{e(x,y)} \times f_x \times f_y$)
        $reduced = w_{e(x,y)} \times f_x \times f_y$;
      else
        $reduced = w_{e(u,v)} \times f_u \times f_v$;
      $a' = \frac{reduced}{f_u \times f_v}$;
      $w_0 = \frac{w_{e(x,y)} \times f_x \times f_y - reduced}{f_x \times f_y}$;
      $w_1 = \frac{a' \times f_v}{f_x}$, $w_2 = \frac{a' \times f_u}{f_y}$;
      Set weight $w_{e(u,v)} - a'$ to $e(u,v)$, set weight w_0 to $e(x,y)$;
      Add weight w_1 to $e(u,x)$, add weight w_2 to $e(v,y)$;
      Update S_E;

We suppose each user has one sensitive label in this paper. This method can be easily extended to multiple sensitive labels. We use $l_{\text{pub}}(u)$ and $l_{\text{pub}}(v)$ to represent the published label lists of user u and v respectively. Suppose the learning algorithms show that when $sim(l_{\text{pub}}(u), l_{\text{pub}}(v)) \geqslant t$ ($sim(l_{\text{pub}}(u), l_{\text{pub}}(v))$ is the similarity between $l_{\text{pub}}(u)$ and $l_{\text{pub}}(v)$), the missing labels of a conservative user can be predicted from his/her friend. We can extend our algorithm as followings. Our first step needs to delete $p\%$ edges, which connect this user to another user with their published label lists' similarity larger than or equal to t. When we do the influential value make up, we never connect two nodes whose label lists' similarity is larger than or equal to t. The second step still guarantees the edge changing ratio $p\%$ in each conservative user's neighborhood.

4 Experiment

4.1 Datasets

We test our algorithms on three real datasets.

• *Facebook Data*. We extract a subgraph of Facebook, which contains 1 422 nodes. We set each user's label as his/her graduation school. The weights on the edges represent the frequency of posting on each other's wall.

• *ArXiv Data*. ArXiv(arXiv.org) is an e-print service system in physics, mathematics, computer science, etc. We extract a co-author graph in computer science, which contains 6 563 nodes. Each node denotes an author, and each edge means two authors have at least one co-authored paper. Each edge has a weight that represents the number of papers co-authored by the two endpoints. We set each author's research field as his/her node label. There are totally 37 different node label values.

• *Arnet Data*. ArnetMiner (http://www.arnetminer.net/) is an academic researcher social network collected by Tsinghua University. We use the country where each researcher belongs to as the node label. We extract a subset of the whole graph, which contains 11 932 nodes and 20 different country labels.

4.2 Learning Algorithms

We test six learning algorithms which are similar to or the same as the algorithms proposed in [3-4].

• *Most Influential Neighbor Label by Frequency* (*MI Frequency*). Frequency is defined as the occurrence number of l in u's neighborhood. The label with the maximum frequency is set as u's real label.

• *Most Influential Neighbor Label by Weight* (*MI Weight*). Weight is defined as the total weight of nodes whose label is l in u's neighborhood: $\sum_{e(u,y) \wedge y.label=l} w_e$. The label with the maximum weight is set as u's real label.

• *Most Influential Neighbor Label by Influence* (*MI Influence*). Influence is defined as the related influen-

[④]When we set a new weight to an edge, if the new weight is 0, we delete this edge. If $e(u,x)/e(v,y)$ does not exist, we create edge $e(u,x)/e(v,y)$ with weight w_1/w_2. Otherwise, we increase the weight of $e(u,x)/e(v,y)$ by w_1/w_2.

tial values of nodes whose label is l in u's neighborhood $\sum_{e(u,y)\wedge y.label=l} w_e \times f_y$. The label with the maximum influence is set as u's real label.

• *Most Influential Label by Overlapping Node Number* (*MI Number Overlapping*). The studies for community mining often use the overlapping ratio of two nodes' neighborhood graphs to represent the similarity between these two nodes. Suppose u's neighborhood graph is G_u and v's neighborhood graph is G_v, $similarity(u,v) = \frac{|G_u \cap G_v|}{|G_u \cup G_v|}$. We compute the similarity on overlapping node number: $similarity(u,v) = \frac{\sum_{\forall x \in G_u \wedge x \in G_v} 1}{\sum_{\forall x \in G_u \vee x \in G_v} 1}$. Label l's impact on u is computed as: $\sum_{\forall v \wedge v.label=l} similarity(G_u, G_v)$. The label with the maximum impact is set as u's real label.

• *Most Influential Label by Overlapping Influence* (*MI Influence Overlapping*). We compute the similarity of two nodes on overlapping node influence: $similarity(u,v) = \frac{\sum_{\forall x \in G_u \wedge x \in G_v} x.influence}{\sum_{\forall x \in G_u \vee x \in G_v} x.influence}$. Label l's impact on u is computed as: $\sum_{\forall v \wedge v.label=l} similarity(G_u, G_v)$. The label with the maximum impact is set as u's real label.

• *Graph-Based Semi-Supervised Learning Algorithm* (*SSL*). Local and global consistency semi-supervised learning algorithm[4] is a learning method that considers both local consistency and global consistency in a graph G. The algorithm is designed to predict missing labels in social networks.

4.3 Utilities

When publishing a privacy preserving graph, besides privacy protection, the other important issue is to check the utilities of the published graph. People test different graph characteristics through experiments to represent the utilities[9-12,18-20]. The proposed method already preserves one important utility, the influential values. Similar to other studies, we also test several other frequently used utilities.

• *Clustering Coefficient* (*CC*). The *CC* of a node in a graph quantifies how close its neighbors are to be a clique (complete graph). The *CC* of the whole graph is: $CC_G = \frac{1}{N} \sum_{\text{all node } i} CC_i$, where N is the number of nodes in graph G. We test the change ratio of *CC*:

$$\frac{|CC_G - CC_{G'}|}{CC_G} \times 100\%.$$

• *Average Hop Distance* (*AHD*). The AHD is a concept in network topology that is defined as the average number of steps along the shortest paths (the length of the shortest path is represented by the number of hops) for all possible pairs of network nodes. It is defined as: $AHD_G = \frac{2}{N(N-1)} \sum_{\forall n_i, n_j \in G} d(n_i, n_j)$. We test the change ratio of AHD:

$$\frac{|AHD_G - AHD_{G'}|}{AHD_G} \times 100\%.$$

• *Most Influential Neighbor Label* (*MI*). This utility measures the neighborhood changes for open users (since a conservative user's neighborhood by default should be changed). We use the first five different learning methods introduced in Subsection 4.2 (The SSL method is a global method, which is unsuitable to be used to represent the neighborhood of a single user) to compute the most influential label l to a node u respectively. The percentage of the open users whose most influential label changed is used to measure this utility.

• *Queries*. In this experiment, we use one-hop queries to test the graph change. The one-hop query[8] has the following form: "the number of labels l_1 and l_2 that have a direct link". We test all possible queries whose results are not smaller than $1\% \times |E|$. We use the average change ratio of the answers to measure this utility. The change ratio of a query q is calculated as

$$r = \frac{|n - n'|}{n} \times 100\%,$$

where n is the query result from the original graph and n' is the result from the published graph.

• *Average Shortest Length* (*APL*). The APL in a weighted graph is defined as the average shortest path length (the length of the shortest path is represented by the sum of edge weights in the path) for all possible pairs of network nodes. It is defined as: $APL_G = \frac{2}{N(N-1)} \sum_{\forall n_i, n_j \in G} d(n_i, n_j)$. We test the change ratio of APL:

$$\frac{|APL_G - APL_{G'}|}{APL_G} \times 100\%.$$

• *Average Degree* (*AD*). Since the degree of nodes is an important property of a graph, we also test the change of average degree. Suppose the degree of a node u is $D(u)$, the average degree AD of a graph G is $\frac{\sum_{\forall u \text{ in } G} D(u)}{N}$. We test the change ratio of AD:

$$\frac{|AD_G - AD_{G'}|}{AD_G} \times 100\%.$$

4.4 Results

In our experiment, we first test the existence of the label-neighborhood relationship on these three graphs. Then we show our edge editing method can guarantee the privacy. Next, we demonstrate the maximum change ratio of the influential values to confirm this utility has been successfully preserved. Finally, we test a group of other graph utilities to show how much change our method brings to these utilities besides preserving the influential values.

4.4.1 Learning Accuracy on the Original Graph

Firstly, we randomly select the conservative nodes and use the six learning algorithms to predict the blank labels in the social networks. If a label is frequently appearing in the graph (e.g., if 70% nodes in this graph have this label, then the label does not need to be protected), we do not select nodes with such a label as conservative nodes. In our experiment, we set this frequent bound as 50%. Fig.6 shows the learning accuracy of the three graphs respectively. From the results, we can see the label-neighborhood relationship exists in the real datasets. In Facebook, the three methods using one-hop neighborhood information can achieve more than 50% accuracy. In ArXiv, most methods achieve 60%∼80% accuracy. In Arnet, the accuracy is around 25%∼35%. The high accuracy of learning algorithms confirms that it is necessary to preserve privacy against the learning algorithms[⑤].

4.4.2 Learning Accuracy on the Modified Graph

To test the privacy protecting effects of our method, we run the six learning algorithms on the published graphs with different edge change ratios respectively. Figs. 7∼12 show the learning accuracy of MI frequency,

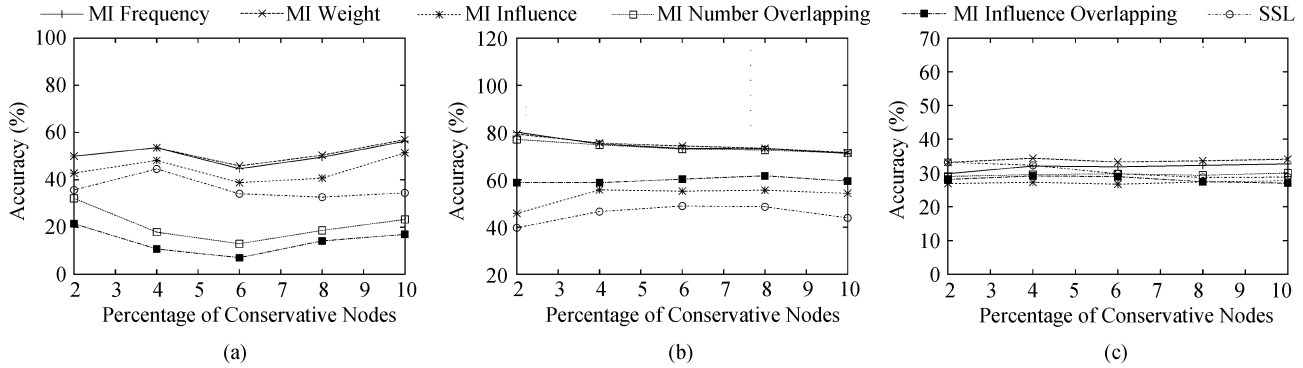


Fig.6. Learning accuracy on the original graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

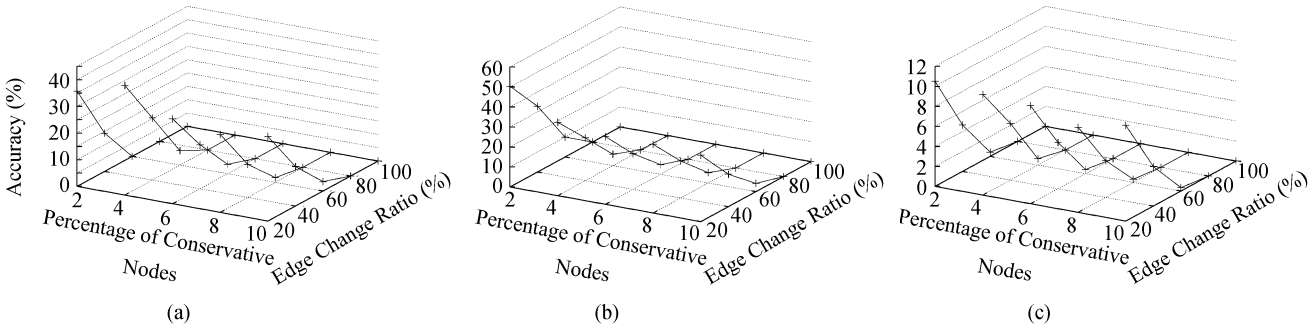


Fig.7. Learning accuracy of MI frequency on the published graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

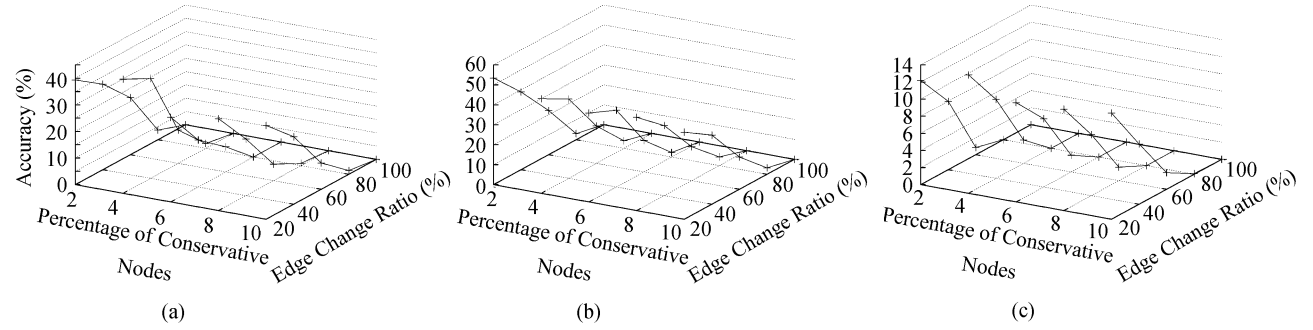


Fig.8. Learning accuracy of MI weight on the published graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

⑤We randomly select 2% to 10% users whose labels are minority (less than 50%). For example, in Facebook dataset, the majority label occupies 80%. Therefore, the relative ratios of conservative users we tested are from 10% to 50%.

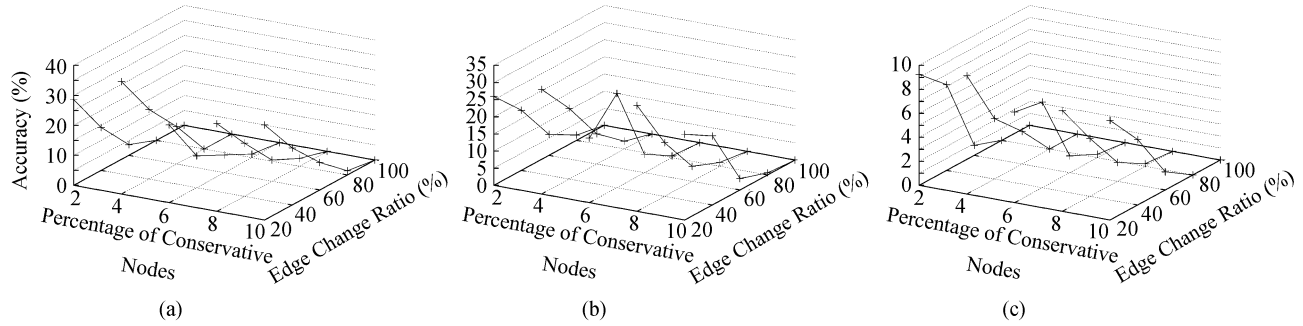


Fig.9. Learning accuracy of MI influence on the published graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

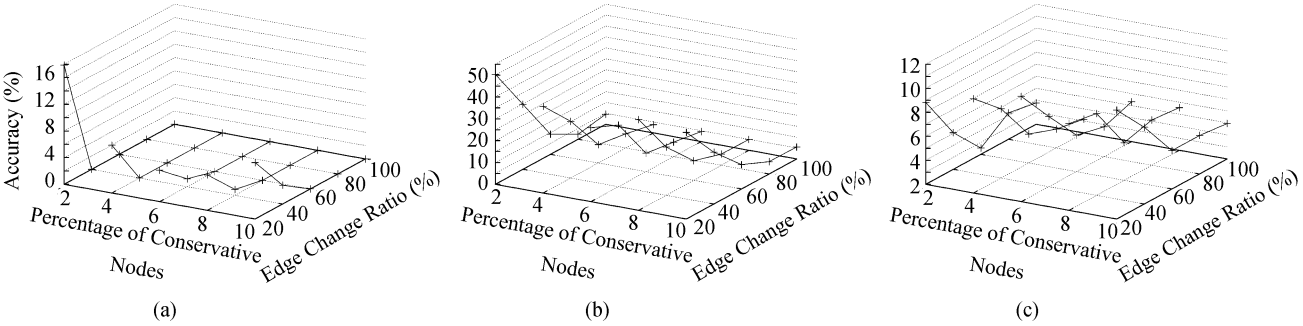


Fig.10. Learning accuracy of MI number overlapping on the published graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

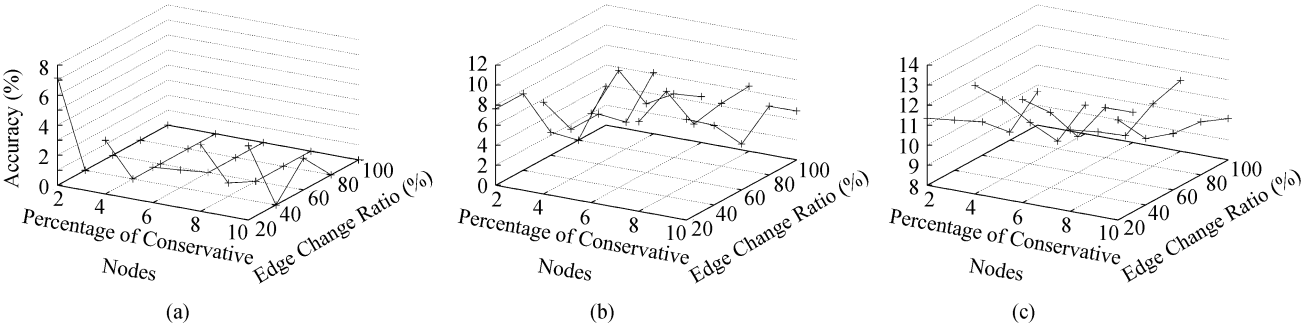


Fig.11. Learning accuracy of MI influence overlapping on the published graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

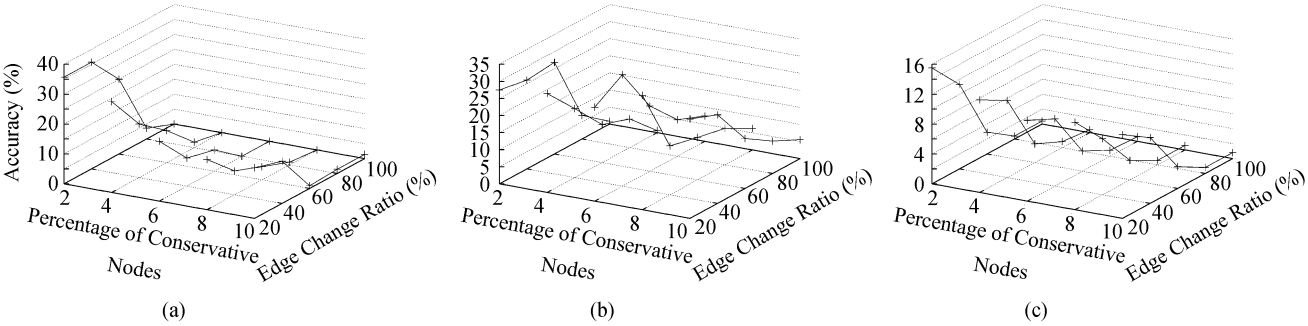


Fig.12. Learning accuracy of SSL on the published graphs. (a) Facebook. (b) ArXiv. (c) Arnet.

MI weight, MI influence, MI number overlapping, MI influence overlapping and SSL on the published graphs respectively. From the result, we can see that the learning accuracy of the learning algorithms decreases with the increase of edge change ratios. When the edge change ratio $p\%$ reaches 60%, the accuracy of algo-

rithms becomes quite low in almost all cases. When 100% edges are changed, most algorithms can only get 0% accuracy. This result confirms the protecting effectiveness of our method. In practice, the data publisher can set different $p\%s$ for different users to further increase the published graph's randomness.

We generate 5-degree-anonymous graphs[11] for the three datasets and run the learning algorithm MI frequency on them. The results are shown in Table 1, where we can find the learning algorithm can still learn the blank labels correctly with high accuracy. This shows that the previous models for preventing "structure attack" cannot prevent "label-structure attack".

Table 1. Learning Accuracy of MI Frequency on the 5-Degree Anonymous Graphs

Percentage of Conservative Nodes	Facebook (%)	ArXiv (%)	Arnet (%)
2	53.6	47.3	39.9
4	53.6	54.6	39.4
6	41.2	53.9	40.4
8	46.0	54.9	41.5
10	46.3	52.4	41.8

4.4.3 Preservation of Influential Values

In order to show the influential values are indeed preserved, we test the maximum change ratio of each node's influential value after the edge editing. The results are shown in Fig.13. The maximum change ratios of Facebook, ArXiv and Arnet are 0.0065%, 0.2% and 0.26% respectively. From the result, we can see the change ratios are very small and can be neglected. The maximum change ratio is not zero due to the error of real number computation in computer. We also observe that the maximum eigenvalue of a graph's adjacent matrix is retained using our algorithm in the experiment. The maximum eigenvalues of the three graphs' adjacent matrices are 53.183 9, 84.659 4 and 27.234 respectively. We find the graphs generated by using our algorithms retain these three values unchanged under all cases.

4.4.4 Other Utilities of the Modified Graph

The above results show that our method preserves an important metric of the weighted social network. Next, we demonstrate our testing results on other utilities. Figs. 14∼23 give the results of different utilities described in Subsection 4.3 respectively. All the ten utilities become worse with the increasing of conservative users and the ratio of changed edges. This is obvious since the graph is changed more when the number of conservative users or the ratio of changed edges increases. For the path lengths, the maximum change ratios of AHD and APL are 40% and 80% respectively. This is an acceptable result since we have already preserved the influential values (the maximum eigenvalue as well). The change of AHD is smaller than the change of APL. This is because when we compute AHD, only

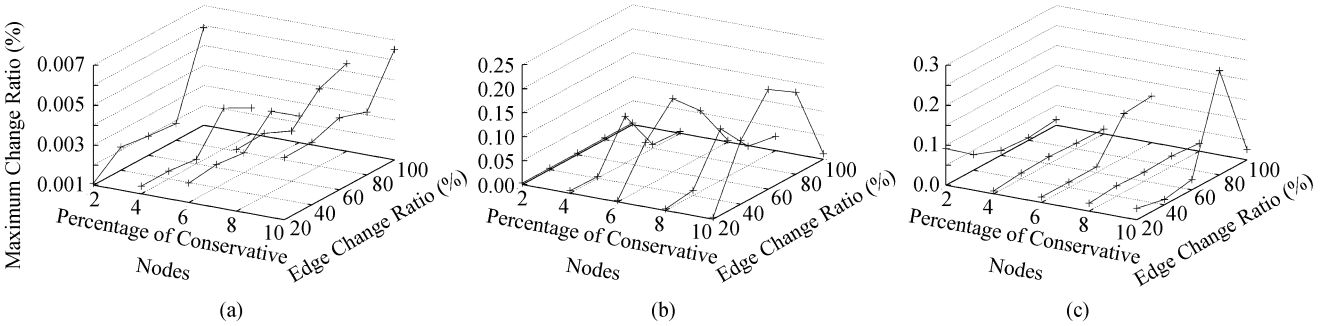


Fig.13. Maximum change ratio of influential values. (a) Facebook. (b) ArXiv. (c) Arnet.

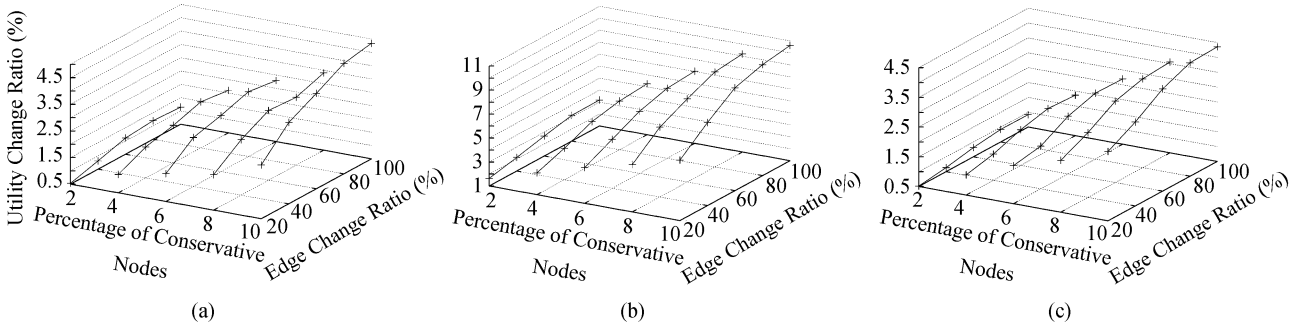


Fig.14. Change ratios of CC. (a) Facebook. (b) ArXiv. (c) Arnet.

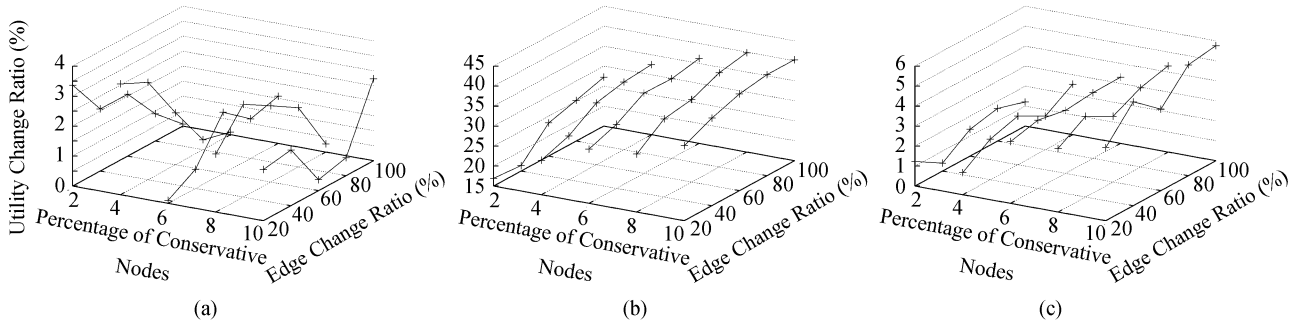


Fig.15. Change ratios of AHD. (a) Facebook. (b) ArXiv. (c) Arnet.

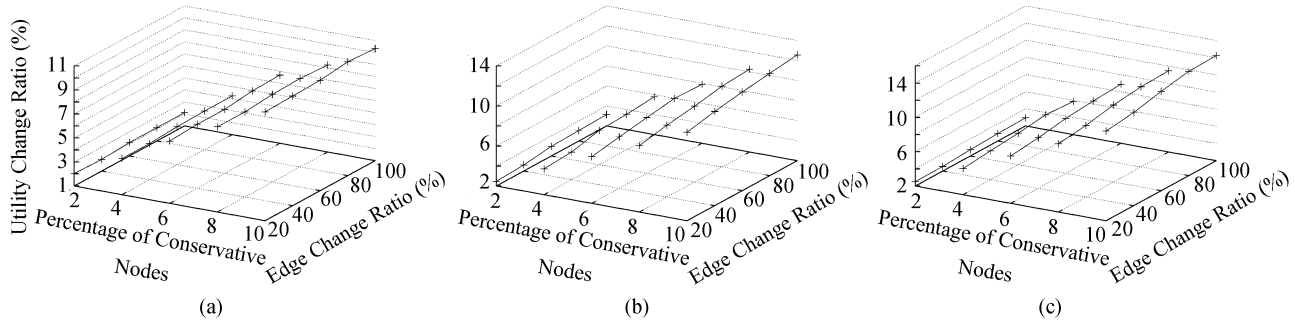


Fig.16. Change ratios of neighborhoods estimated by MI frequency. (a) Facebook. (b) ArXiv. (c) Arnet.

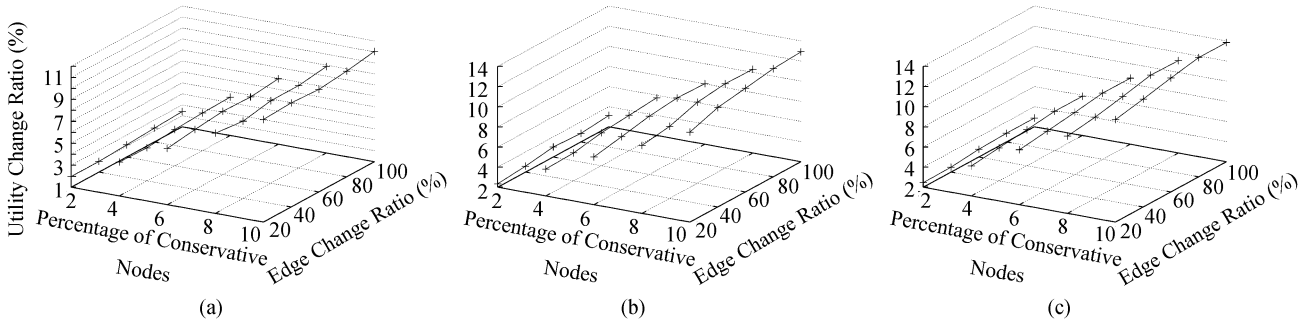


Fig.17. Change ratios of neighborhoods estimated by MI weight. (a) Facebook. (b) ArXiv. (c) Arnet.

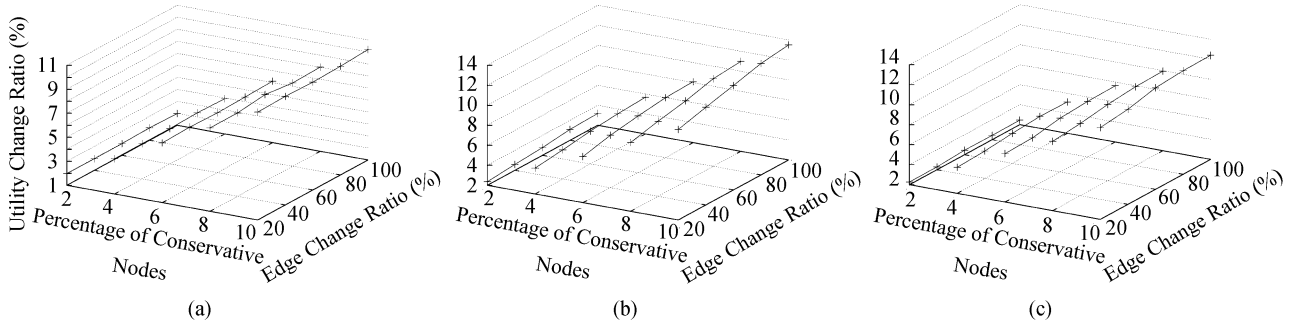


Fig.18. Change ratios of neighborhoods estimated by MI influence. (a) Facebook. (b) ArXiv. (c) Arnet.

the number of changed edges influences it. While both the number of changed edges and the weights on the changed edges influence the APL. Thus the change of APL is larger than that of AHD. The change ratios of all the other utilities are very small. Nearly all of them are less than 10% under all cases. Overall, considering the utilities we tested, our method performs well. The experiments on the utilities show our algorithm preserves other utilities of the graph well besides keeping the influential values unchanged.

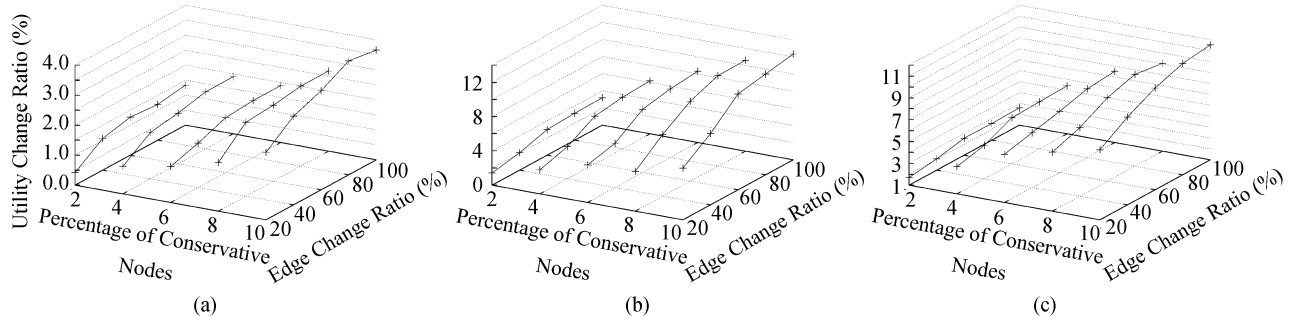


Fig.19. Change ratios of neighborhoods estimated by MI number overlapping. (a) Facebook. (b) ArXiv. (c) Arnet.

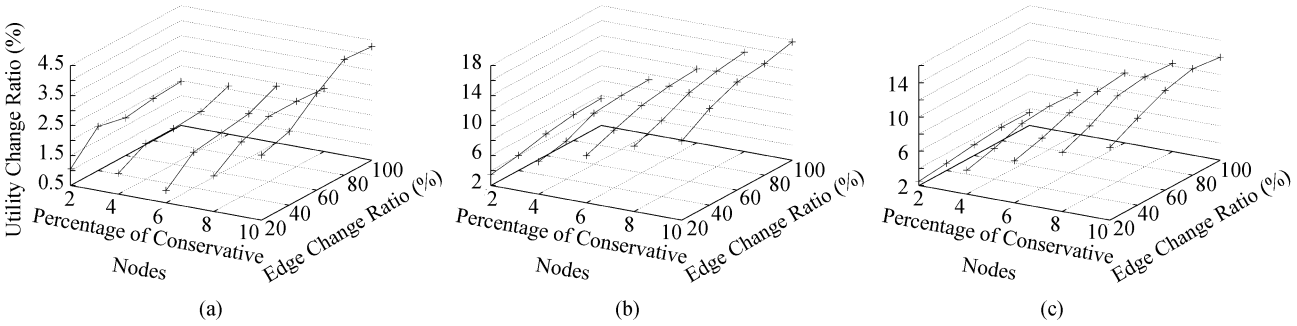


Fig.20. Change ratios of neighborhoods estimated by MI influence overlapping. (a) Facebook. (b) ArXiv. (c) Arnet.

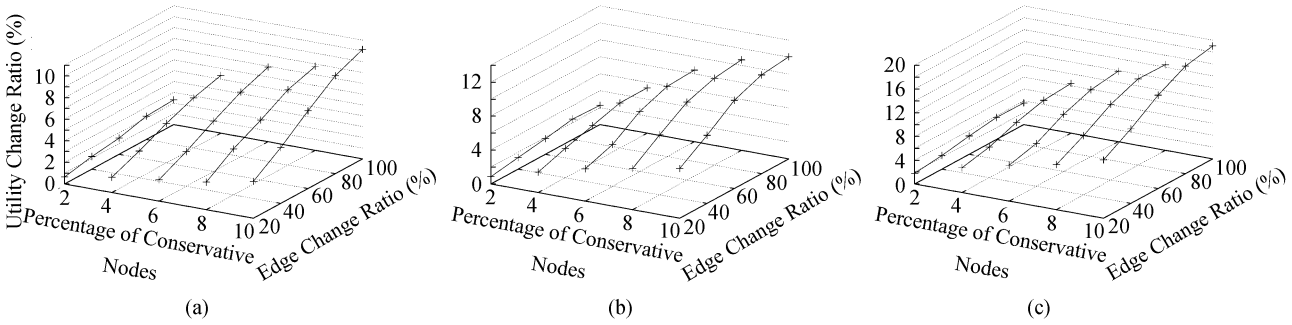


Fig.21. Change ratios of queries. (a) Facebook. (b) ArXiv. (c) Arnet.

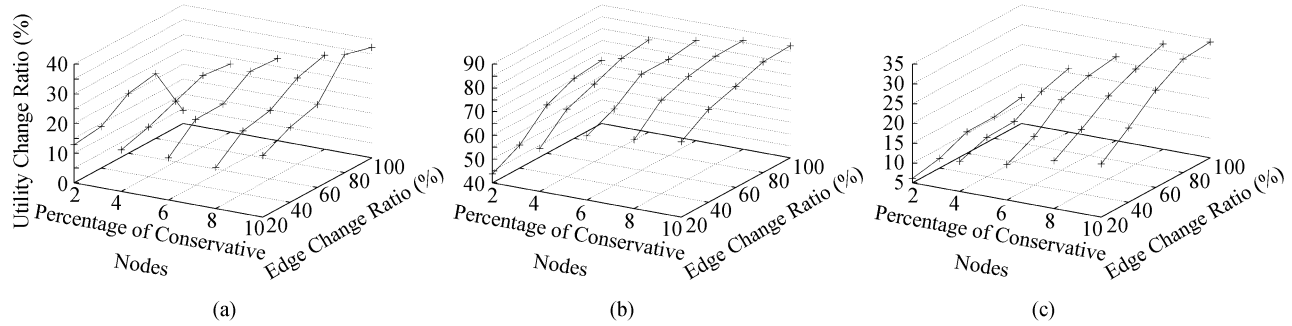


Fig.22. Change ratios of APL. (a) Facebook. (b) ArXiv. (c) Arnet.

5 Related Work

A lot of studies have been proposed to prevent "structure attack" on the published graph. The basic protecting method is to publish an anonymized graph by clustering or edge editing. Clustering is to cluster "similar" nodes together and publish a super node instead of the original nodes. By making each cluster's

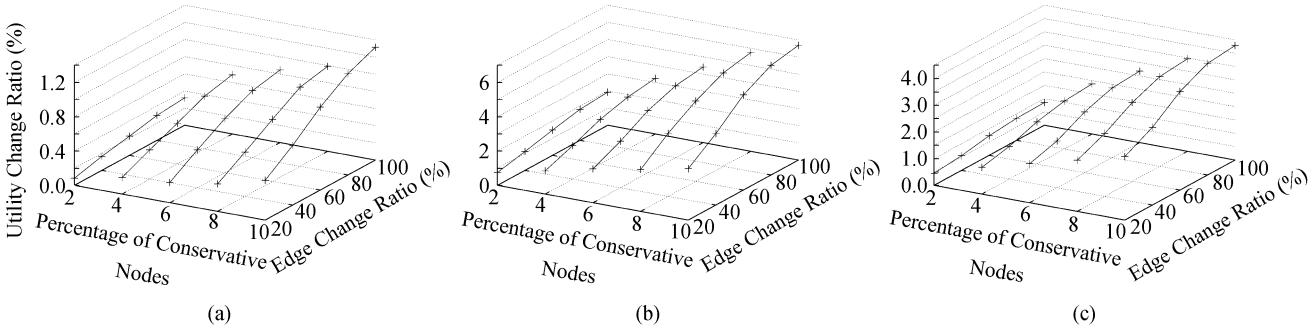


Fig.23. Change ratios of AD. (a) Facebook. (b) ArXiv. (c) Arnet.

size be at least k, k-anonymity is achieved. Since only clusters and the connection information between them are published, the individual's information is protected. A series of clustering models[8,10,18-19,21] were proposed by considering different utility and privacy requirements. The edge editing based approach tries to add or delete edges in order to make the graph satisfy certain structure constraints. Liu and Terzi[11] defined k-degree anonymous model, which requires any degree value appears at least k times in the published graph. Then if an attacker uses degree information to attack a user, he/she will always get at least k candidates. Zhou and Pei[12] proposed a stricter model: each node's neighborhood graph appears at least k times. This model avoids the attack using one-hop subgraph knowledge. Zou *et al.*[20], Wu *et al.*[22] and Cheng *et al.*[9] studied how to generate symmetric graphs to avoid the attack using any structure information. He *et al.*[23] proposed a protection model where each node's local graph with size d ($d \leqslant 1$) must appear at least k times in different communities. Yuan *et al.*[24] implemented an anonymous model which considers the personalized privacy requirements of users. Liu *et al.*[13] designed a graph anonymizing algorithm which preserves the reachability. These studies try to prevent an attacker from matching an anonymous user to an individual. They totally hide the individual users, which leads the information those users want to share unusable. Furthermore, social network websites publish the network data to the third parties who can derive some knowledge about the graph in order to provide more interesting services to the target users. However, without the information from the hidden users, it is difficult to achieve such a goal. The model proposed in this paper protects sensitive information from being attacked by a learning algorithm and meanwhile publishes the useful information for sharing.

Our work studies how to protect the blank labels in the graph crawled from APIs provided by the social network websites. Lindamood *et al.*[25] designed a random edge changing and label replacing method to prevent the learning of blank labels using Bayes classifier. The differences between our work and this work are: 1) We consider personalized user privacy preference setting (publish the labels that a user wants to publish and make sure the labels that a user does not want to publish are not released); while [25] does not consider the information a user wants to publish, which makes some users' published information incorrect after adjustment. 2) [25] does not have a quantifiable guarantee on the protection for each user; while our work guarantees the protection for each user on the privacy parameter p. 3) We make sure the preserving of an important utility in the published graph. Our work tries to publish a graph which protects the sensitive labels against "label-structure attacks" without changing user influential values.

There are some other related studies. Machanavajjhala *et al.*[26] investigated the relationship between the privacy and the accuracy of personalized social recommendations. The differential privacy is considered in [26]. Differential privacy[27] is a privacy protection method, which returns statistical query results to users through an interface instead of publishing sanitized data. A random noise from the Laplace distribution is added to make sure that the output of a statistical query is approximately the same when any single tuple in the database is added or removed. Machanavajjhala *et al.*[26] found that for the majority of nodes in the network, recommendations must either be inaccurate or violate differential privacy. Recently, a series of papers are proposed to use differential privacy on different scenarios, such as histogram query[28], statistical geospatial data query[29-30], frequent item set mining[31-32] and crowdsourcing[30]. This paper considers the privacy protection data publishing case, which is different with the randomized statistical query result returning scenario. Liu *et al.*[1] treated edge weights as sensitive labels and proposed a shortest path length preserving random weight assignment method. Das *et al.*[2] proposed a linear programming based method to protect

edge weights while preserving certain linear properties. These studies do not consider the protection of node labels. The "label-structure attack" is not considered by them either. Akcora *et al.*[33] designed a mechanism to help users estimate their privacy risk in social networks.

6 Conclusions

In this paper, motivated by users' different privacy preferences in the social network, we proposed a new method to publish a graph for customized privacy protection. We found out that it is not secure after leaving users' sensitive information blank. Through some simple label-structure analysis, it is possible for attackers to know users' sensitive labels due to the label-structure association. Our method could promise that those common learning algorithms could not find out the sensitive information. More importantly, we preserved one important metric of the graph during the privacy protection. With the good utility demonstrated by the proposed approach, the social network websites could guarantee privacy when providing data through API to third party applications by adjusting the communication history and providing a partial friend list.

References

[1] Liu L, Wang J, Liu J, Zhang J. Privacy preserving in social networks against sensitive edge disclosure. Technical Report, CMIDA-HiPSCCS 006-08, Department of Computer Science, University of Kentucky, Lexington, USA, 2008.

[2] Das S, Egecioglu Ö, El Abbadi A. Anonymizing weighted social network graphs. In *Proc. the 26th ICDE*, March 2010, pp.904-907.

[3] Zheleva E, Getoor L. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *Proc. the 18th International Conference on World Wide Web*, April 2009, pp.531-540.

[4] Mo M, King I. Exploit of online social networks with community-based graph semi-supervised learning. In *Proc. the 17th International Conference on Neural Information Processing: Theory and Algorithms - Volume Part I*, November 2010, pp.669-678.

[5] Lu Q, Getoor L. Link-based classification. In *Proc. the 20th International Conference on Machine Learning* (*ICML*), August 2003, pp.496-503.

[6] Neville J, Jensen D. Relational dependency networks. *Journal of Machine Learning Research*, 2007, 8(Mar): 653-692.

[7] Tang L, Liu H. Relational learning via latent social dimensions. In *Proc. the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, June 28-July 1, 2009, pp.817-826.

[8] Bhagat S, Cormode G, Krishnamurthy B, Srivastava D. Class-based graph anonymization for social network data. *Proc. VLDB Endow.*, 2009, 2(1): 766-777.

[9] Cheng J, Fu A W, Liu J. K-isomorphism: Privacy preserving network publication against structural attacks. In *Proc. the 2010 International Conference on Management of Data*, June 2010, pp.459-470.

[10] Hay M, Miklau G, Jensen D, Towsley D, Weis P. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 2008, 1(1): 102-114.

[11] Liu K, Terzi E. Towards identity anonymization on graphs. In *Proc. the 2008 International Conference on Management of Data*, June 2008, pp.93-106.

[12] Zhou B, Pei J. Preserving privacy in social networks against neighborhood attacks. In *Proc. the 24th ICDE*, April 2008, pp.506-515.

[13] Liu X, Wang B, Yang X. Efficiently anonymizing social networks with reachability preservation. In *Proc. the 22nd ACM International Conference on Information & Knowledge Management*, October 27-November 1, 2013, pp.1613-1618.

[14] Campbell C S, Maglio P P, Cozzi A, Dom B. Expertise identification using email communications. In *Proc. the 12th International Conference on Information and Knowledge Management*, November 2003, pp.528-531.

[15] Zhang J, Tang J, Li J. Expert finding in a social network. In *Proc. the 12th Int. Conf. Database Systems for Advanced Applications*, April 2007, pp.1066-1069.

[16] Ying X, Wu X. Randomizing social networks: A spectrum preserving approach. In *Proc. SDM*, April 2008, pp.739-750.

[17] Zhang J, Tang J, Liu L, Li J. A mixture model for expert finding. In *Proc. the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, May 2008, pp.466-478.

[18] Zheleva E, Getoor L. Preserving the privacy of sensitive relationships in graph data. In *Proc. the 1st PinKDD*, August 2007, pp.153-171.

[19] Campan A, Truta T M. Data and structural k-anonymity in social networks. In *Proc. the 2nd PinKDD*, August 2008, pp.33-54.

[20] Zou L, Chen L, Özsu M T. k-automorphism: A general framework for privacy preserving network publication. *Proc. VLDB Endow.*, 2009, 2(1): 946-957.

[21] Cormode G, Srivastava D, Yu T, Zhang Q. Anonymizing bipartite graph data using safe groupings. *Proc. VLDB Endow.*, 2008, 1(1): 833-844.

[22] Wu W, Xiao Y, Wang W, He Z, Wang Z. k-symmetry model for identity anonymization in social networks. In *Proc. the 13th International Conference on Extending Database Technology*, March 2010, pp.111-122.

[23] He X, Vaidya J, Shafiq B, Adam N, Atluri V. Preserving privacy in social networks: A structure-aware approach. In *Proc. the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, Sept. 2009, pp.647-654.

[24] Yuan M, Chen L, Yu P S. Personalized privacy protection in social networks. *Proc. VLDB Endow.*, 2010, 4(2): 141-150.

[25] Lindamood J, Heatherly R, Kantarcioglu M, Thuraisingham B. Inferring private information using social network data. In *Proc. the 18th International Conference on World Wide Web*, April 2009, pp.1145-1146.

[26] Machanavajjhala A, Korolova A, Sarma A D. Personalized social recommendations: Accurate or private? *Proc. VLDB Endow.*, 2011, 4(7): 440-450.

[27] Dwork C. Differential privacy. In *Proc. the 33rd ICALP Part II*, July 2006, pp.1-12.

[28] Xu J, Zhang Z, Xiao X, Yang Y, Yu G. Differentially private histogram publication. In *Proc. the 28th IEEE International Conference on Data Engineering*, April 2012, pp.32-43.

[29] Qardaji W, Yang W, Li N. Differentially private grids for geospatial data. In *Proc. the 29th IEEE International Conference on Data Engineering* (*ICDE*), April 2013, pp.757-768.

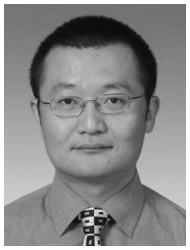
[30] To H, Ghinita G, Shahabi C. A framework for protecting worker location privacy in spatial crowdsourcing. *Proc. VLDB Endow.*, 2014, 7(10): 919-930.

[31] Li N, Qardaji W, Su D, Cao J. PrivBasis: Frequent itemset mining with differential privacy. *Proc. VLDB Endow.*, 2012, 5(11): 1340-1351.

[32] Zeng C, Naughton J F, Cai J Y. On differentially private frequent itemset mining. *Proc. VLDB Endow.*, 2012, 6(1): 25-36.

[33] Akcora C, Carminati B, Ferrari E. Privacy in social networks: How risky is your social graph? In *Proc. the 28th IEEE International Conference on Data Engineering*, April 2012, pp.9-19.

Mingxuan Yuan is currently a researcher of Huawei Noah's Ark Lab, Hong Kong. Before that, he served as a postdoctoral fellow in the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. His research interests include spatiotemporal data storage/mining, telecom data mining, and data privacy.



Lei Chen is currently an associate professor in the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. Prof. Chen got his Ph.D. degree in computer science from University of Waterloo. His research interests include multimedia and time series databases, crowdsourcing, sensor and peer-to-peer databases, and stream and probabilistic databases. He is a member of the IEEE and the IEEE Computer Society.

Philip S. Yu is a professor in the Department of Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in information technology. He spent most of his career in the IBM Thomas J. Watson Research Center and was the manager of the Software Tools and Techniques Group. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. He is a fellow of ACM and IEEE. He is an associate editor of the ACM Transactions on Internet Technology and the ACM Transactions on Knowledge Discovery from Data.

Hong Mei is a professor in the School of Electronics Engineering and Computer Science, Peking University, and the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include software engineering, software reuse, distributed object technology and middleware, and programming languages. Professor Mei received a Ph.D. degree in computer science from Shanghai Jiao Tong University in 1992.