Privacy Preserving Social Network Publication on Bipartite Graphs

Jian Zhou, Jiwu Jing, Ji Xiang, Lei Wang

Privacy Preserving Social Network Publication on Bipartite Graphs

Jian Zhou*, Jiwu Jing and Ji Xiang, and Lei Wang

The State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences, China.
{jzhou,jing,jixiang,lwang}@is.ac.cn

Abstract. In social networks, some data may come in the form of bipartite graphs, where properties of nodes are public while the associations between two nodes are private and should be protected. When publishing the above data, in order to protect privacy, we propose to adopt the idea generalizing the graphs to super-nodes and super-edges. We investigate the problem of how to preserve utility as much as possible and propose an approach to partition the nodes in the process of generalization. Our approach can give privacy guarantees against both static attacks and dynamic attacks, and at the same time effectively answer aggregate queries on published data.

Key words: data publishing, privacy preservation, bipartite graph, generalization

1 Introduction

Due to the rapid growth in the number of services and applications that leverage social networks, privacy in social networks becomes a serious concern, particularly when social network data is published. Recently, there have been many works devoting to the privacy preserving publication of social network data.

Among these works, [1] studied a particular type of network data that can be modeled as bipartite graphs: there are two types of entities, and an association only exists between two entities of different types. One typical example is customers and products bought from a pharmacy. The association between two nodes is considered to be private and needs to be protected while properties of entities are public. For example, the set of customers of a pharmacy may not be considered particularly sensitive, and the set of products which it sells may also be public knowledge. However, the set of products bought by a particular customer is considered private, and should not be revealed.

Rather than masking or altering the graph structure, their methodology focuses on masking the mapping from entities to nodes. The approach not only

* This work was supported by National Natural Science Foundation of China (Grant No. 70890084/G021102, and 61003274) and Knowledge Innovation Program of Chinese Academy of Sciences (Grant No. YYYJ-1013).

ensures the privacy is protected, but also allows a wide variety of ad hoc analyses and novel valid uses of the data. More details about the work of [1] are described in Section 2.

However, their approach only performs well against *static attacks*, where the adversary solely uses the published data. In *dynamic attacks* where the adversary has some background knowledge such as existing associations or degrees of nodes, privacy issues may arise.

Following the work of [1], we propose an improved approach to publish an anonymized version of a given bipartite graph via generalization. Our work and contributions include:

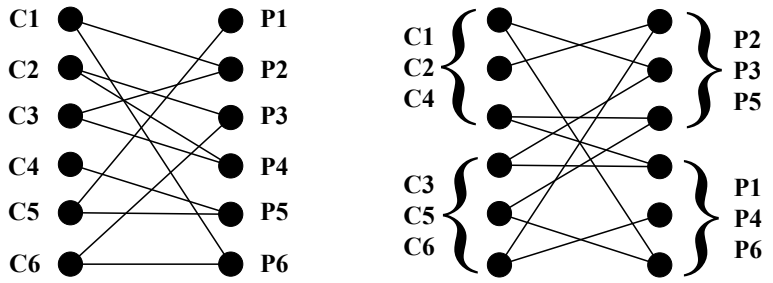
- We investigate how to measure and optimize the utility when generalizing social network data that can be modeled as a bipartite graph.
- We propose an approach to partition the nodes in order to preserve as much utility as possible under a given level of privacy.
- In order to identify our approach, we compare it with the naive one called *Simple Generalization*. Although both maintain privacy against static attacks and dynamic attacks, our approach can answer a broad class of queries more accurately. All experiments are carried out with real social network data.

The remainder of the paper is organized as follows: in Section 2 we introduce some important notions. In Section 3 we investigate how to optimize utility and propose our approach. In Section 4, we demonstrate our approach by experimental study using some real social network data. In Section 5, we introduce some related works. At last, in Section 6 we make a conclusion and discuss some open problems.

2 Preliminaries

The social network data we investigate are those that can be represented in the form of large, sparse bipartite graphs G = (V,W,E). That is, the bipartite graph G consists of $|V|$ nodes of one type, $|W|$ nodes of a second type, and a set of $|E|$ edges $E \subseteq V \times W$. 'Sparse' indicates the edges account for a tiny portion of all possible ones, which holds in many real social networks. We want to protect the associations between nodes rather than the information of nodes themselves. For example, in Figure 1(a), information such as C1 is a 26-year-old male or P2 is an OTC drug is public knowledge. In contrast, information such as C1 bought P2 is considered to be sensitive. Thus, the final published data should guarantee those sensitive information can't be deduced. As in prior work, we are only concerned with attacks making positive inferences rather than those making negative inferences. For example, it is not allowed that an adversary could deduce C1 bought P2, but we don't guarantee the information such as C1 did not buy P1 is protected. Meanwhile, there is another requirement the published data should satisfy: it can be utilized for analyzing or mining. In our scenario, it should be able to answer a broad class of queries based on standard SQL aggregates (sum,

count, avg, min, max) with relatively acceptant bias. In [1], the author listed three typical types of queries with increasing complexity.



(a) Graph of Raw Data (b) Graph of Published Data

Fig. 1. Safe Grouping

- Type 0 - Graph structure only: compute an aggregate over all neighbors of nodes in V that satisfy some P_n (i.e., predicates over solely graph properties of nodes).
  E.g.: Count the average number of products bought by each customer.
- Type 1 - Attribute predicate on one side only: Compute an aggregate for nodes in V satisfying P_a(i.e., predicates over attributes of the entities); Compute an aggregate on edges to nodes in V satisfying P_n from nodes in W satisfying P_a.
  E.g.: Find the average number of male customers buying only a single product.
- Type 2 - Attribute predicate on both sides: Compute an aggregate for nodes in V satisfying P_a to nodes in W satisfying P'_a.
  E.g.: Find total number of OTC products bought by customers aged from 20 to 30.

A novel approach is proposed in [1] to anonymize the data. As illustrated in Figure 1, they partition nodes into groups but preserve the structure of the graph exactly. Thus analysis principally based on the graph structure is correct. On the other hand, privacy is ensured because given a group of nodes, there is a secret mapping from these nodes to the corresponding group of entities. It is impossible for an adversary to learn, within a group, which node corresponds to which entity solely based on the published data. A (k, l)-grouping means that V is split into size k groups, W into size l groups. It is convenient to offer a tradeoff between privacy and utility by setting the values of k and l.

However, if we just simply group the nodes, an adversary may be able to deduce some associations due to the uniformly dense interaction pattern between two groups. E.g., if a published graph happens to contain complete subgraph between group $\{v1, v2, v3\}$ and group $\{w1, w2, w3\}$, an adversary can immediately

infer all the connections between these six nodes. In order to avoid such attacks caused by lack of diversity [2], the notion of *safe grouping* is proposed. Denote by H the function mapping nodes into groups, we say H_V is a safe grouping of V if it satisfies:

$$\forall v_i \neq v_j \in V : H_V(v_i) = H_V(v_j) \Rightarrow \nexists w \in W : (v_i, w) \in E \wedge (v_j, w) \in E$$

A (k, l)-grouping is safe if both H_V and H_W are safe groupings. Intuitively, for either type of nodes, a safe grouping ensures any two nodes in the same group have no common neighbors of the other type. Safe grouping not only ensures a level of sparsity between groups, but also restricts the pattern of allowed links. The author proved that based on the anonymized data via safe groupings, an adversary can make a correct inference with the probability no more than $1/max(k, l)$.

A greedy algorithm which we called *Simple Safe Grouping* is also provided to find safe groupings in practice in [1]. In order to find a safe k-grouping of V, for each node $v \in V$ in turn, the algorithm tries to find the first existing group that satisfies: first, the number of nodes in it is fewer than k; second, the condition of safety won't be breached when v joins. If such a group is successfully found, add v to it. Otherwise, a new group is started, containing v alone. After processing all nodes, there may be a few groups with fewer than k nodes. The algorithm collects those nodes together, increases the allowed group size by one, and begins the next round. The algorithm continues until each node is placed in a group with the size no less than k or some large group size is reached. The process of finding a safe l-grouping of W is the same as V, and they are totally independent. Although this method is possible to fail, it is easy to find safe groupings for those sparse bipartite graphs.

3 Bipartite Graph Generalization

3.1 Motivation

Although the approach described in Section 2 is resilient to static attacks, where the adversary tries to deduce explicit associations solely by analyzing the published information. However, in dynamic attack models where an adversary already knows some background knowledge, he may be able to identify the relevant nodes, and further infer additional associations. Background knowledge may be available through sources external to the released data or obtained by intruding into the network. For example, considering the published data shown in Figure 2, the adversary is able to identify the nodes using various background knowledge. Note that dashed lines represent edges to other groups, which do not affect this example.

– *learned link attack*: If the adversary learns C1 bought P2, he can use the fact that there is only one edge between the group of C1 and the group of P2 to identify C1 and P2 with nodes in the anonymized graph. This attack is

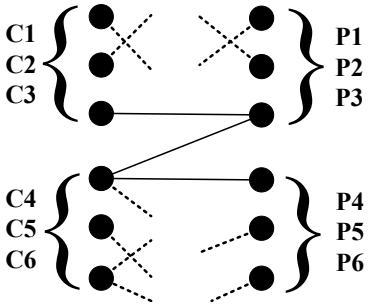


Fig. 2. Attack Examples

also discussed in [1]. Although the author proved that their approach could mitigate the damage, it still suffers from such attack.
– *learned degree attack*: If the adversary learns C4 bought 3 products meaning the degree of C4 is 3, which is unique in the group of C4, he can easily identity C4 with the node representing it.

So, if the adversary learns C1 bought P2 and C4 bought 3 products, he can infer that C4 bought P2, no matter how many other nodes are in the groups. In fact, any structural background information can be possibly leveraged by the adversary to compromise the privacy. The adversary can easily de-anonymize a node with a unique subgraph around it.

In order to preclude such attacks, we consider the generalization technique, which partitions nodes and edges into groups called super-nodes and super-edges. Compared with [1], we only release the number of edges between each subset instead of the full edge information. Using the generalization technique, it is impossible for the adversary to distinguish between individuals in the same group even he has some structural background knowledge. For a super-node containing k nodes, the probability of any one being the target is at most $1/k$. As described in Section 2, in order to defend against the attacks caused by lack of diversity, we should still follow the condition of safety such that for a (k, l)-grouping there are at most $min(k, l)$ edges between two groups. For a super-node containing at least k nodes, as the probability of any one node being the target is at most $1/k$, which has been proved in [3], the probability of correctly guessing the existence of an edge in the original data is bounded by $min(k, l)/kl = 1/max(k, l)$. Therefore, the privacy is guaranteed against powerful dynamic attacks, let alone static attacks.

In detail, generalization approaches can be divided into two steps: first, partition the nodes into groups; second, merge grouped nodes into super-nodes and incident edges into super-edges, then give each an aggregate description. In our approach, each super-node is labeled with all information of nodes in it and each super-edge is labeled with the number of edges between two super-nodes. Step two is quite simple and direct, so the core of generalization is how to partition the nodes in order to retain utility. A natural method is to use Simple

Safe Grouping. However, this greedy algorithm is aim to find a safe grouping as quickly as possible. In fact, for a sparse bipartite graph, there are many possible safe groupings and our target is to find such one that the loss of utility is as little as possible after generalization.

3.2 Algorithm Description

First of all, we should provide an approach to measure utility. We adopt the same method as [3] which evaluates utility by possible worlds. Possible worlds are those consistent with the data published. A possible world could be interpreted as an input that will result in the anonymization being produced. Assuming that all possible worlds are treated equally, the smaller the size of possible worlds introduced via generalization is, the better the utility is.

From the perspective of a single safe group G_i with k_i nodes in it, if G_i connects to n groups of the other type and the number of edges between them is $c_{i1}, c_{i2}, \ldots, c_{in}$ respectively, because each node in G_i has at most one edge to another group of the other type, the size of possible worlds induced by G_i is:

$$|W(G_i)| = \prod_{j=1}^{n} \binom{k_i}{c_{ij}}$$

Note that in our work we are only concerned with structural configurations and consider two isomorphic possible worlds as different ones, which is also implied by [3]. Therefore, in order to reduce the size of possible worlds, for any safe group g, we propose two rules:

- **R1**: The number of groups that g connects to should be minimized.
- **R2**: Under the precondition of R1, the number of edges between two groups should be either maximized or minimized.

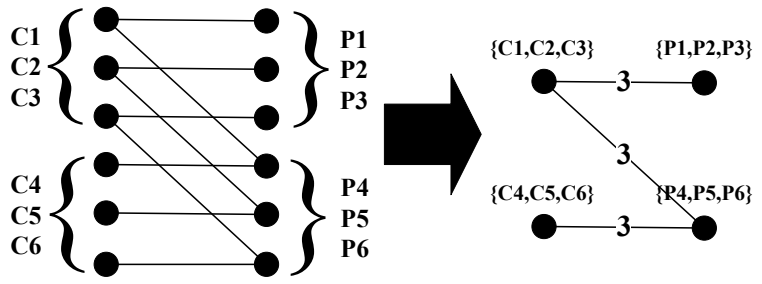


Fig. 3. Ideal Safe Grouping

Due to the symmetric property, in order to better find safe groupings consistent with $R1$ and $R2$, we only consider (k, k)-groupings in our work. Figure

3 illustrates the ideal safe grouping, where for any two safe groups, the number of edges between them is either zero or maximum. In this ideal situation, the process of merging edges doesn't introduce any additional possible worlds. However, in most cases, this ideal safe grouping doesn't exist. Instead, based on $R1$ and $R2$, we propose an improved safe grouping algorithm to partition nodes.

Algorithm 1: ImprovedSafeGrouping

Input: $G = (V, W, E), k$
begin
  Sort V by degree in descending order
  $VG \leftarrow$ Find a Safe Grouping of V
  foreach $w \in W$ **do**
    $VG_W(w) \leftarrow \{vg \in VG | \exists v \in vg : (v, w) \in E\}$
  end
  $WG \leftarrow \varnothing; RC \leftarrow \varnothing; C \leftarrow W$
  while $|C| > 0$ **do**
    $c \leftarrow$ select the one with the maximum degree from C
    $wg \leftarrow \varnothing \cup c$ /*Create a new group*/
    $C \leftarrow C - c$
    for $i \leftarrow 0$ *to* $k - 1$ **do**
      $c \leftarrow SelectGroupMember(G, VG, wg, C)$
      if $c = \varnothing$ **then** break;
      $wg \leftarrow wg \cup c$
      $C \leftarrow C - c$
    end
    if $|wg| < k$ **then** $RC \leftarrow RC \cup \{w \in wg\}$
    else $WG \leftarrow WG \cup wg$
  end
  while $|RC| > 0$ **do**
    foreach $wg \in WG$ **do**
      $c \leftarrow SelectGroupMember(G, VG, wg, RC)$
      $wg \leftarrow wg \cup c$
      $RC \leftarrow RC - c$
    end
  end
end

For a bipartite graph $G = (V, W, E)$, we first sort V by degree in descending order before finding a safe grouping because of two reasons: first, putting nodes with similar degree in one group are more likely to lead to less super-edges in generalized graph, which is helpful to reduce the number of groups involved to form possible worlds; second, sorting by node degree will greatly improve accuracy on queries involving node degree as illustrated in [1]. Then we partition V into groups VG using Simple Safe Grouping. Because of the symmetry, it is also available to deal with W first. In our work, we first partition the type of

Algorithm 2: SelectGroupMember

Input: $G = (V, W, E), VG$(/*groups of V*/) , wg (/*a temporary group of W*/) , C (/*nodes not processed*/)
Output: c (/*the node selected to insert into wg*)

```
1  begin
2      SC ← {c ∈ C|∄v ∈ V, w ∈ wg : (v, w) ∈ E ∧ (v, c) ∈ E} /*Obtain
       Candidates*/
3      if |SC| = 0 then return ∅
4      foreach c ∈ SC do
5          weight(c) ← 0
6          foreach vg ∈ {g ∈ VG|∃v ∈ g : (v, c) ∈ E} do
7              x ← |{a ∈ wg|vg ∈ VG_W(a)}|
8              if x > 0 then
9                  weight(c) ← weight(c) + 1 + δ(x)
10             else
11                 weight(c) ← weight(c) − 1
12         end
13     end
14     c ← select the one with the maximum weight from SC
15     return c
16 end
```

nodes with less number as this provides more opportunity to construct better groups of the other type. When the number is approximate, we can randomly choose one type. Next, we partition W according to VG. This is the crucial difference between our approach and the original one. The original algorithm deals with V and W in the same way and they are totally independent from each other, while in our algorithm, the result of partitioning W is influenced by VG. The process of partition W is described as follows:

1. Create a new group.
2. Pick nodes satisfying the condition of safety to fill the group until its size reaches k or there is no candidate. On each turn, the node is picked according to VG and those already picked.
3. Repeat step 1 and step 2 until all nodes in W have been processed.

The key point when partitioning W is how to pick nodes to fill a new group. In most cases, there are quite a lot of nodes satisfying the condition of safety and we should try to pick those who connect to the same set of VG as much as possible. We first pick the node with the maximum degree from the candidate list to add to a new group g. We deal with the nodes with higher degree first because they have more important affects on utility. We then record the groups that g has already connected to as $VG_{WG}(g)$ and depend on it to assign each candidate a weight indicating the priority it should be picked. Denote by $VG_W(w)$ the groups the node w connects to, the weight of a candidate c can be expressed by $weight(c) = \sum_{vg \in VG_W(c)} (\sigma(c, vg) + \delta(c, vg))$, where $\sigma(c, vg)$ is consistent

with $R1$ and $\delta(c, vg)$ is consistent with $R2$. For each group vg in $VG_W(c)$, we in turn test whether it occurs in $VG_{WG}(g)$. If the answer is yes, the value of $\sigma(c, vg)$ is set to 1, implying that we don't need to introduce an additional group. Otherwise it is set to -1. In fact, $\sum \sigma(c, vg)$ can be seen as the overlap degree between $VG_W(c)$ and $VG_{WG}(g)$. On the other hand, $\delta(c, vg)$ could be any function that increases with the increase of the number of edges between vg and g, and satisfies $|\sum \delta(c, vg)| < 1$, which ensures $R1$ precedes $R2$. The introduction of $\delta(c, vg)$ makes the scheme prefer to form a dense connection and a sparse connection rather than two average connections. On each turn, we pick the one with the maximum weight to add to the group. Then, we update weights of all candidates and continue.

After processing all nodes in W, there may also be some groups with fewer than k nodes in it. We collect them together and deal with them by expanding the allowed group size, which is similar to the original algorithm. However, those nodes are quite few when bipartite graphs are sparse enough and we can try to place them in the groups with as few edges as possible. Therefore, although there may be some groups whose size is $k + 1$ or even $k + 2$, the effect on utility is negligible.

From the perspective of object processed, the original algorithm deals with nodes in turn while we deal with groups in turn. In the original algorithm, the author tries to place a given node in some group. Different from it, in our algorithm, we are given a group, and try to fill it with some nodes. So our algorithm provides an opportunity to form a desired group not only satisfying the condition of safety but also retaining the data utility as much as possible.

4 Experimental Study

4.1 Experimental Framework

In order to demonstrate the efficacy of our approach, we perform our experiments on two real social network data sets that can be represented by bipartite graphs. The first data set (HEP-Th) presents information on papers in theoretical high-energy physics, which is derived from the abstract and citation files provided for the 2003 KDD Cup competition and could be retrieved from `http://kdl.cs.umass.edu/databases/hepth-data.xml.gz`. The data set contains $|V| = 9,200$ distinct authors, $|W| = 29,555$ distinct papers and $|E| = 58,515$ edges. The second data set (Epinions) is collected by Paolo Massa in a 5-week crawl from the shopping web site Epinions.com [4]. Because the original data set is rather huge, we extracted part of it, which contains $|V| = 1,000$ distinct customers, $|W| = 8,477$ distinct products and $|E| = 12,522$ edges.

As in prior work, there are too many possible queries on bipartite graphs. In order to efficiently evaluate the utility of the anonymized data, we specifically study the accuracy of three sample queries with different properties, which is the same as [1]. The three queries are:

- **Query A**: Find the average number of papers of any author satisfying predicate P_a. This is a type-1 query with an attribute predicate only. We vary

the selectivity of P_a from 0.1 to 0.9. Higher selectivity means fewer eligible nodes.

- **Query B**: Find the total number of single author papers satisfying P_a. This is also a type-1 query with both attribute predicates and structural predicates. The selectivity of P_a is varied as above.
- **Query C**: Find the total number of papers satisfying P_a having authors who satisfy P'_a. This is a type-2 query. We vary the selectivity of both P_a and P'_a.

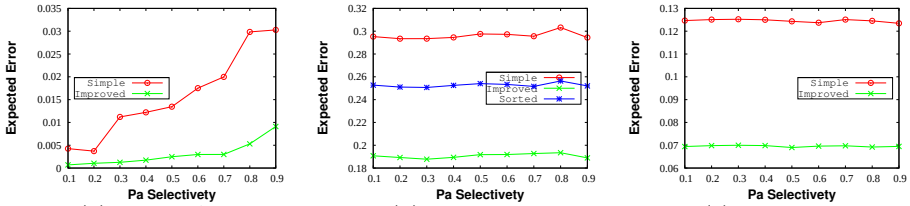
Given a predicate, whether a node is qualified is synthetic in our work. Meanwhile, we assume those eligible ones are uniform distributed. We do not introduce any type-0 queries because the results of type-0 queries are usually single values. In fact, a typical type-0 query can be regarded as a special case of Query A when P_a equals 1.0. So the results of Query A are enough to evaluate the accuracy of type-0 queries by the trends.

We use the approach called *Sampling Consistent Graphs* [5] to answer these queries on anonymized data. The approach randomly samples a graph that is consistent with the anonymized data, and uses this sample to perform the analysis. One could repeat this several times to get an approximate expected answer. To clearly show the trends, we repeat each experiment over ten random choices of predicates and extract ten samples from each anonymized data. For each query, if the correct answer on original data is Q and the expected answer on anonymized data is u, we are able to calculate the expected error $|u - Q|/Q$. We use this expected error to precisely evaluate utility. Smaller values of the expected error indicate better utility.

4.2 Experimental Results

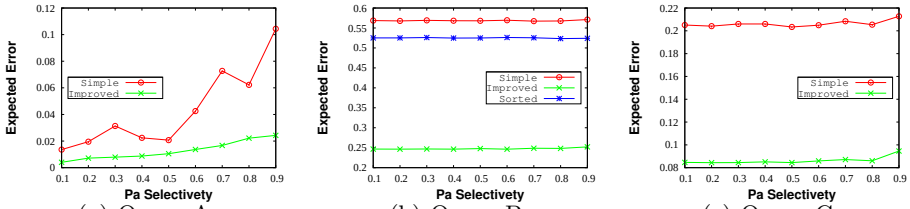
We compare our approach called *Improved Generalization* with *Simple Generalization* that first partitions the nodes using Simple Safe Grouping and then aggregates nodes and edges. Figure 4 and Figure 5 plot the expected error vs. selectivity of P_a under the two approaches, respectively on the HEP-Th data set and the Epinions data set. In these figures, we perform the approaches using a $(10, 10)$ safe grouping. When using other appropriate values of group size, there are a few differences, however, the trends are the same.

Both Improved Generalization and Simple Generalization are robust against static attacks and dynamic attacks as described in 3.1. In other words, they guarantee the same level of privacy. However, the utility of our approach is much better than Simple Generalization, as shown in Figure 4 and Figure 5, and the accuracy of answering queries is almost doubled. Because query B involves a structural predicate, sorting the data by degree before finding a safe grouping is helpful to answer such queries. So we also compare our approach with the sorted one that first sorts nodes by degree in descending order, then performs Simple Generalization. In fact, in Algorithm 1, we also sort V in the beginning. It can be seen from Figure 4(b) and Figure 5(b) that our approach still performs better. For Query C, the selectivity of P'_a is set to 0.5. Experimental results for



(a) Query A (b) Query B (c) Query C

Fig. 4. Expected Error on HEP-Th



(a) Query A (b) Query B (c) Query C

Fig. 5. Expected Error on Epinions

other values of P'_a are similar. Particularly, the expected error using Improved Generalization is bounded by 0.25 in our experiments, which is accurate enough for most analyses.

5 Related Work

In recent years, there have been many works trying to anonymize and publish social network data under various scenarios, which should guarantee both privacy and utility. Generally speaking, the approaches of preserving privacy in social networks can be divided into three categories. One is K-anonymity [6] privacy preservation [3, 7–9], which ensures in the anonymized graph there are at least k nodes indistinguishable with each other in terms of some types of structural patterns. Another approach is edge randomization [10–12] where the privacy is protected in a probabilistic manner. This approach modifies graph structure by randomly adding/deleting edges or switching edges on the premise of preserving certain aggregate characteristics. These works all focus on simple graphs and need to modify the graph structure via a sequence of edge deletions and additions.

Different from the above two approaches, the generalization approach we adopt in our work simply groups nodes and edges. The idea of generalization was previously proposed in anonymizing tabular data and was adopted by [5] and [3] respectively to anonymize rich interaction graphs and simple graphs, which is different from our work. Moreover, [5] just simply partitioned the nodes into groups without considering the problem of how to improve utility. [3] searched

the approximate optimal partitioning using simulated annealing [13]. However, we require additional structure of the partitions to ensure safe groupings.

Besides, there are some works investigating how to publish social network data that can be represented by some interesting graph models, which usually contain much richer information in addition to the simple graph structure. [14] considered the graphs, in which there are multiple types of edges but only one type of nodes and edges are classified as either sensitive or non-sensitive. The author proposed a greedy algorithm to construct anonymous graphs, which can protect privacy of the sensitive relationships while are still useful. Different from most ongoing works focusing on un-weighted social networks, [15] and [16] studied the situations where the network edges as well as the corresponding weights are considered to be private.

6 Conclusion and Discussion

When publishing social network data on bipartite graphs, in order to make it resilient against both static attacks and dynamic attacks, we propose to anonymize the data based on the idea of generalization. We investigate how to measure and optimize the utility and then propose an approach to produce the published data. Compared with the Simple Generalization approach, our approach improves utility greatly.

In our algorithm, we first partition nodes of one type using Simple Safe Grouping, and then partition nodes of the other type according to the result. We also consider the method that constructs a group of V and a group of W in turn. However, the weight of a candidate indicating how much utility it preserves is hard to evaluate. From the perspective of efficiency, our approach needs to calculate the weight of each candidate every time a new group member is to be picked, so our approach is less efficient than Simple Generalization and is time-consuming when the social network contains a large amount of nodes. However, it can be seen as a tradeoff between algorithm efficiency and utility. Moreover, utility is paid much more attention in most cases when publishing social network data.

References

1. Cormode, G., Srivastava, D., Yu, T., Zhang, Q.: Anonymizing bipartite graph data using safe groupings. The VLDB Journal **19** (2010) 115–139
2. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) **1** (2007) 3
3. Hay, M., Miklau, G., Jensen, D., Towsley, D., Li, C.: Resisting structural re-identification in anonymized social networks. The VLDB Journal **19** (2010) 797–823
4. Massa, P., Avesani, P.: Trust-aware bootstrapping of recommender systems. In: ECAI. Volume 6., Citeseer (2006) 29–33

5. Bhagat, S., Cormode, G., Krishnamurthy, B., Srivastava, D.: Class-based graph anonymization for social network data. Proceedings of the VLDB Endowment **2** (2009) 766–777
6. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty Fuzziness and Knowledgebased Systems **10** (2002) 557–570
7. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society (2008) 506–515
8. Zou, L., Chen, L., Özsu, M.T.: k-automorphism: a general framework for privacy preserving network publication. Proc. VLDB Endow. **2** (2009) 946–957
9. Cheng, J., Fu, A., Liu, J.: K-isomorphism: privacy preserving network publication against structural attacks. In: Proceedings of the 2010 international conference on Management of data, ACM (2010) 459–470
10. Hanhijärvi, S., Garriga, G.C., Puolamäki, K.: Randomization techniques for graphs. In: Proceedings of the 9th SIAM International Conference on Data Mining (SDM '09). (2009) 780–791
11. Ying, X., Wu, X.: Randomizing social networks: a spectrum preserving approach. In: SDM. (2008) 739–750
12. Ying, X., Wu, X.: Graph generation with prescribed feature constraints. In: SDM. (2009) 966–977
13. Russell, S., Norvig, P.: Artificial intelligence: A modern approach. 2003. Section15 (2003)
14. Zheleva, E., Getoor, L.: Preserving the privacy of sensitive relationships in graph data. In: Proceedings of the 1st ACM SIGKDD international conference on Privacy, security, and trust in KDD, Springer-Verlag (2007) 153–171
15. Liu, L., Wang, J., Liu, J., Zhang, J.: Privacy preservation in social networks with sensitive edge weights. In: 2009 SIAM International Conference on Data Mining (SDM 2009), Sparks, Nevada. (2009) 954–965
16. Das, S., Egecioglu, Ö., El Abbadi, A.: Anonymizing edge-weighted social network graphs. Computer Science, UC Santa Barbara, Tech. Rep. CS-2009-03 (2009)