Utility-Oriented K-Anonymization on Social Networks

Yazhe WANG[1], Long XIE[1], Baihua ZHENG[1], and Ken C. K. LEE[2]

[1] Singapore Management University
{yazhe.wang.2008, longxie, bhzheng}@smu.edu.sg
[2] University of Massachusetts Dartmouth
ken.ck.lee@umassd.edu

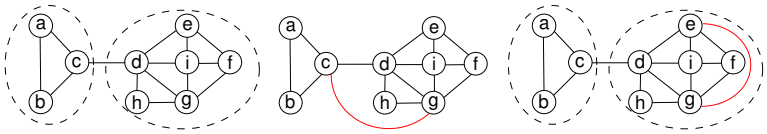
Abstract. "Identity disclosure" problem on publishing social network data has gained intensive focus from academia. Existing k-anonymization algorithms on social network may result in nontrivial utility loss. The reason is that the number of the edges modified when anonymizing the social network is the only metric to evaluate utility loss, not considering the fact that different edge modifications have different impact on the network structure. To tackle this issue, we propose a novel *utility-oriented social network anonymization* scheme to achieve privacy protection with relatively low utility loss. First, a proper utility evaluation model is proposed. It focuses on the changes on social network topological feature, but not purely the number of edge modifications. Second, an efficient algorithm is designed to anonymize a given social network with relatively low utility loss. Experimental evaluation shows that our approach effectively generates anonymized social network with high utility.

Keywords: social networks, privacy,k-anonymity, utility, HRG

1 Introduction

With the rapid growing of social network applications and the proliferation of the social network data in recent years, social network data privacy has attracted more and more attentions from academia [1–4]. Among various privacy problems on social networks, *identity disclosure* [1] in publishing social network data is most concerned. Usually, a social network is modeled as a complex graph. Given a social network G, a published social network G^* has identity disclosure problem if there is a vertex v in G^* that can be mapped to an original entity t in G with a high probability. It has been demonstrated that even after removing all identifiable personal information (e.g., names and identity card numbers), an attacker is still able to identify an original entity in a published social network with high confidence based on the knowledge of the topological structure around the entity, such as degree, neighborhood and subgraph.

To tackle this issue, various anonymization models have been proposed based on the principle of *k-anonymity*. They all have to make changes to the original social networks in order to protect the privacy. Generally, from privacy protection



(a) Social network G (b) Published G_1^* (c) Published G_2^*

Fig. 1. An example of the impact of adding edges to achieve 2-degree anonymity.

point of view, more changes on the original social network are preferred. However, it will greatly affect the *utility* of the social network. Ideally, we prefer that a modified social network does not disclose the true identity of each vertex, and meanwhile it still provides comparable level of accuracy with the original data for the corresponding mining and analysis activities. The trade-off between privacy and utility in publishing tabular data has been well studied [5], however, it is still new in the field of social network publishing.

To the best of our knowledge, most of previous works use the total number of modified edges to measure the social network utility loss. In other words, they try to achieve anonymity with minimum number of edge modifications. However, this measurement is not effective as it assumes each edge modification has an equal impact on the original social network properties. For example, a social network G is given in Fig. 1(a). Its vertices are naturally divided into two communities, as indicated by the dash circles. The vertices within the same community are strongly connected, while connections between the vertices of different communities are weak. Assume there are two corresponding social networks G_1^* and G_2^* published based on G, as illustrated in Fig. 1(b) and Fig. 1(c) respectively. In terms of privacy, both G_1^* and G_2^* satisfy 2-degree anonymity, that is for any given vertex, there is at least one other vertex sharing the same degree. In terms of utility, they are same as they both only add one edge to the original social network. However, the change that G_1^* makes (i.e., adding edge between vertex g and vertex c) is more significant, compared with the change made by G_2^* (i.e., adding edge between vertex g and vertex e), as G_2^* remains the two-community structure of G, while G_1^* blurs the boundary of the communities.

Based on the above observations, we believe that the number of edge modifications alone is not a good measurement of the utility loss and hence the existing k-anonymization algorithms based on this measurement have nature flaws in providing high-utility anonymized social network data. To address this concern, we propose a novel *utility-oriented social network anonymization* approach in this paper to achieve high privacy protection and low utility loss. First, a proper utility model is proposed based on the *hierarchical community structure* of the social network, to measure the utility loss of a published social network. It focuses on social network topological feature changes instead of purely the number of edge modifications. Second, an efficient k-anonymization algorithm is designed to modify a given social network G to G^*, where G^* satisfies the privacy requirement (e.g., k-degree anonymity) with relatively low utility loss.

The rest of the paper is organized as follows. Section 2 presents some background knowledge and reviews related works about social network privacy protection. Section 3 details the new utility model based on the hierarchical community structure of the social network. Section 4 presents the k-anonymization algorithm based on the proposed utility model. Section 5 reports the experiment results. Finally, section 6 concludes the paper.

2 Preliminaries and Related Work

We first present the terminology that will be used in this paper. Similar as other works, we model the social network as an undirected graph $G(V, E)$, where vertex set V represents the entities (e.g., persons, organizations, et al), and edge set E represents the relationships between two entities (e.g., friendship, collaboration, et al). An edge between vertex v_i and v_j is denoted as $e(v_i, v_j) \in E$. [1]

2.1 Structural Re-identification Attack and K-Anonymity

Social network data publishing faces various privacy challenges, and we only focus on the identity privacy problem in this work. We assume the entities' true identities in the original social network G are sensitive, and hence they are eliminated in the released social network G'. An attacker tries to locate a target entity in G' based on her background knowledge about the target. We use F to denote the type of background knowledge that an attacker uses and $F(t)$ to represent the evaluated value of F for a target t. If F is based on the structure of the graph, such as degree, neighborhood and subgraph, this attack is called *structural re-identification attack (SRA)* [6], as defined in Definition 1.[2]

Definition 1 (Structural Re-identification Attack (SRA)). *Given a social network $G(V, E)$, its published graph $G'(V', E')$, a target entity $t \in V$ and the attacker's background knowledge $F(t)$, the attacker performs the structural re-identification attack by searching for all the vertices in G' that could be mapped to t, i.e., $V_{F(t)} = \{v \in V' | F(v) = F(t)\}$. If $|V_{F(t)}| << |V'|$,[3] then t has a high probability to be re-identified.*

K-anonymity is a widely adopted principle to prevent the SRA on social networks [1–4], formally defined in Definition 2. Please note that we need to specify the type of background knowledge F (e.g., degree, neighborhood) that an attacker has in order to formally define k-anonymity. However, when the context of F is clear, we use k-anonymity in this paper for the brevity of presentation.

Definition 2 (K-Anonymity). *Given a graph $G(V, E)$, and a type of attacker's background knowledge F, G satisfies k-anonymity against F, iff for each $v \in V$, there are at least $(k-1)$ other vertices in V with the same F value of $F(v)$.*

[1] For ease of presentation, we use "graph" and "social network" interchangeably.

[2] The attacker could possess some non-structural information as well (e.g., the vertex(edge) labels), but we only consider the structural information in this paper.

[3] Notation $|V|$ refers to the cardinality of a set V.

Different approaches have been proposed to convert a given graph into a k-anonymized graph. In this work, we only focus on edge modification, that is to modify a graph into a k-anonymized graph only via inserting and/or deleting edges. It is expected that the topological structure of a graph will be changed by modifications and the published graph is expected to lose some utility of the original one. Consequently, social network publishing should take both the privacy and utility into consideration. Ideally a published social network G' should satisfy k-anonymity and meanwhile cause a utility loss as small as possible.

2.2 Related Work

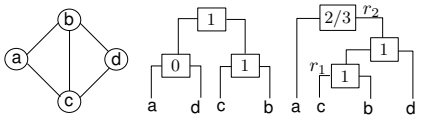
Structural re-identification is one of the major privacy concerns in social network publishing [7]. The initial study demonstrates that simply removing the identification information of the entities is not sufficient to protect privacy as the true identities of the vertices can be inferred due to the structural uniqueness of some embedded small subgraphs (i.e., SRA). Various classes of SRAs are thereafter proposed based on the types of attackers' background knowledge, including vertex refinement queries, subgraph queries and hub-fingerprint queries [6].

To counter the SRAs, various protection schemes were proposed [1–4, 6, 8]. For example, the random permutation approach protects privacy by randomly inserting and deleting edges [8], which is simple but may significantly affect the graph utility. Then graph generalization based approaches abstract an original graph into a super graph by grouping the vertices into small blocks represented by super nodes and linking super nodes via edges if the corresponding blocks are connected [6]. The super graph introduces great uncertainty in the released data, thus increasing the difficulties of using the data.

Recently, researchers start to apply the principle of k-anonymity [9] to protect the social network privacy. Based on the types of attacker's background knowledge, various k-anonymity schemes and algorithms have been proposed. For example, k-degree anonymity scheme is to against the attackers with knowledge of entity degree [1]; scheme proposed in [3] considers the attackers with the information about the vertices' neighborhood; k-automorphism and k-symmetry schemes can resist multiple structural attacks. k-automorphism modifies the graph such that for each vertex, there are at least $(k-1)$ other structurally equivalent vertices [4]; and k-symmetry utilizes the symmetry property of the social network to modify the graph [2]. All these algorithms anonymize graphs based on edge/node modification operations (i.e., addition and/or deletion), and try to preserve the utility of the released graphs. However, most of them employ the number of edge/node changes as the only measurement to quantify the utility loss, which is *not* effective, as demonstrated in Section 1. The new model we propose, as will be detailed in next section, actually gives a better measurement.

3 Graph Utility Measurement

In order to support utility-oriented k-anonymization, the first issue to address is how to measure the utility loss of a published social network, compared with



(a) Graph G (b) H_G^1 (c) H_G^2

Fig. 2. A graph G and its corresponding HRGs.

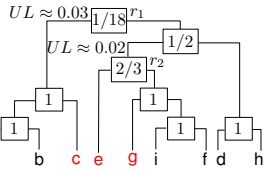


Fig. 3. An example of HRG and utility loss.

the original social network. As pointed out in previous sections, the number of edge modifications, the most common utility loss measurement, is not effective as it treats all the edges equally. In our work, we aim at developing a new measurement that reflects the different impacts of various edge operations on the social network structure. Given the fact that social network is a complex graph, there are many aspects of its topological properties, such as transitivity, eigenvector, and community structure. Among them, the community structure is a central organizing principle of complex graph and it is a core graph topological feature which has a strong correlation with other important features (e.g., transitivity and betweenness). Consequently, we decide to use the community structure to represent the topological features of social networks as it provides a simple representative to reflect the influence of the edge modification on social network structure in a micro perspective. In this section, we first introduce the *Hierarchical Random Graph* (HRG) [10] for modeling the community structure, then present its construction algorithm, and finally introduce a novel hierarchical community entropy to quantify the graph structural features (i.e., utility).

3.1 Hierarchical Random Graph

The community structure of a graph is a nature grouping of its vertices. The vertices have dense connections within the groups, but sparse connections between groups. Recent studies suggest that the communities of social networks often exhibit hierarchical organization (i.e. the large communities further contain small communities). Consequently, we adopt a *Hierarchical Random Graph* (HRG) model to capture this hierarchical organization of communities [10].

Given a graph $G(V, E)$, the HRG is a binary tree, denoted as H_G. The leaf nodes of H_G correspond to vertices of G, and each internal node r is associated with a *connection probability* p_r. Given a sub-tree T_r that is rooted at node r, p_r is the probability that a vertex in the left subtree T_r^L has an edge with a vertex in the right subtree T_r^R. It reflects the connection strength between the vertices in the left and right subtrees. The larger the p_r is, the stronger the connection is. Mathematically, connection probability p_r is defined in Equation (1).

$$p_r = |E_r|/(|T_r^L| \cdot |T_r^R|) \,, \tag{1}$$

where $|E_r|$ is the number of edges $e(v_i, v_j) \in E$ with $v_i \in T_r^L$ and $v_j \in T_r^R$, and $|T_r^L|$ ($|T_r^R|$) is the number of vertices in r's left (right) subtree. A graph G and

its two possible HRGs are depicted in Fig. 2. Naturally, the vertices in sub-tree T_r rooted at node r are regarded as a community C_r.

3.2 Constructing HRG

As mentioned above, the tree-structure of HRG organizes the underlying social network hierarchically. However, for a given social network G, there are multiple possible HRGs. How to construct an HRG that captures the topological structure of a given social network G best is a key issue we have to address in order to use HRG to model social network. In the literature, a likelihood function $\mathcal{L}$ has been developed to evaluate the fitness of a given HRG H_G to G, with $\mathcal{L}(H_G) = \prod_{r \in H_G} \left[p_r^{p_r}(1-p_r)^{1-p_r}\right]^{|T_r^L| \cdot |T_r^R|}$ [10]. Accordingly, a representative HRG construction algorithm uses Markov chain Monte Carlo method to sample the space of all possible HRGs with probability proportional to $\mathcal{L}$ and returns the sampled HRG having the maximum $\mathcal{L}$ value.

Essentially, $\mathcal{L}(H_G)$ is the posterior probability that the model H_G generates G. However, in fact, a model H_G generates G with high probability does not necessarily mean that H_G is a good model of the hierarchical community structure of G. We use example depicted in Fig. 2 to support our statement. We observe that the partition of H_G^2 is more meaningful than H_G^1 in terms of the community structure. This is because H_G^1 groups a and d into the same community which is improper as there is even no edge between them. However, $\mathcal{L}(H_G^1) = 1 >> \mathcal{L}(H_G^2) = 0.148$. Consequently, maximize the likelihood value $\mathcal{L}(H_G)$ does not necessarily reflect a good community organization of a social network G. Thus, the Monte Carlo sample algorithm which is developed based on $\mathcal{L}(H_G)$ cannot return the best HRG that preserves most, if not all, the topological structure properties of a given social network G as we expect, not to mention its extremely high construction cost.

To overcome the shortcomings of the existing HRG construction approach, we propose a simple greedy bottom-up construction algorithm. Initially, the algorithm forms each vertex of G as one community (i.e. the leaf nodes of H_G). Thereafter, communities (i.e. subtrees) with strong connections are merged from bottom to up until one unified community is achieved. The connection strength of two community C_i and C_j is again evaluated by the connection probability $p_{C_i,C_j} = |E_{ij}|/(|C_i||C_j|)$, with $|E_{ij}|$ the number of edges connecting vertices of C_i with vertices of C_j, and $|C_i|$ ($|C_j|$) the number of vertices of C_i (C_j). Due to the limitation of the space, we omit the detailed HRG construction algorithm.

3.3 Hierarchical Community Entropy

As mentioned above, we use an HRG H_G to represent the topological features of a given social network G. In this subsection, we introduce an information entropy based utility function to quantify the information (i.e. utility) of G reflected by H_G. In the literature, there are various graph entropy definitions available, based on different focuses. For example, entropy of the degree distribution, target

entropy and road entropy [11]. However, none of the above entropy definitions considers the graph's hierarchical community information. Consequently, we propose a new *Hierarchical Community Entropy* (HCE) to represent the information embedded in the graph community structure.

HCE is defined based on the edge grouping. Given a graph $G(V, E)$ and its community structure represented by H_G, there are $|V| - 1$ internal nodes in H_G as H_G is a complete binary tree with $|V|$ leaf nodes. Each internal node r in H_G roots a subtree corresponding to a group of crossing edges E_r of G. Given the numbers of vertices in left subtree and right subtree represented by $|T_r^L|$ and $|T_r^R|$ respectively, $|E_r| = |T_r^L| \cdot |T_r^R| \cdot p_r$. The HCE of a given H_G of a social network G, denoted as $HCE(G, H_G)$ is defined in Equation (2), with p_t represents the connection probability. For example, the graph depicted in Fig. 1(a), the HCE of its H_G shown in Fig. 3 is 2.807.

$$HCE(G, H_G) = -\sum_{t=1}^{|V|-1} \frac{|T_t^L| \cdot |T_t^R| \cdot p_t}{|E|} \log(\frac{|T_t^L| \cdot |T_t^R| \cdot p_t}{|E|}) \ . \tag{2}$$

When we insert/delete an edge on a graph G, the modification will be reflected by the connection probability change of an internal node on H_G, thus changing the HCE value. Continue our example. When we add a new edge $e(v_g, v_c)$ to G in Fig. 1(a), the connection probability of the lowest common ancestor of v_g and v_c (i.e. the root) is changed from $\frac{1}{18}$ to $\frac{1}{9}$, with the new HCE value of the modified graph being 2.840. Similarly, if we add a new edge $e(v_g, v_e)$ to G, its HCE value will be 2.790. The utility loss caused by the edge operation is evaluated by the change of the HCE value, as defined in Equation 3.

$$UL(G, G') = |HCE(G, H_G) - HCE(G', H'_G)| \ , \tag{3}$$

where G' is the modified graph, and H'_G is the corresponding HRG derived from H_G with updated connection probabilities. The main goal of this work is to anonymize the social network while making the utility loss as small as possible. Continue above example. As adding edge $e(v_g, v_c)$ causes the utility loss of $|2.840 - 2.807| = 0.033$, and adding edge $e(v_g, v_e)$ causes utility loss of 0.017, the second modification has a less significant impact on the graph structure and hence is preferred. It also confirms our observation in Fig. 1.

4 HRG based *K*-Anonymization

After introducing the HRG model and the information entropy based utility measurement, we are ready to present HRG-based k-anonymization algorithm that tries to anonymize a given social network via edge operations with the utility loss as small as possible. In the following, we first present the basic idea of HRG-based k-anonymization and then detail its main components individually. Notice that although we only focus on k-degree anonymity in this section, our approach is general and it is applicable to other k-anonymity based privacy protection schemes on social networks (e.g. k-neighborhood anonymity).

Algorithm 1: HRG based k-anonymization algorithm

Input: Graph $G(V, E)$, H_G, F, and k
Output: K-anonymized graph G'

```
1  G'(V', E') = G(V, E);
2  D* = estimate(G, F, k);
3  while G' is not k-anonymized do
4      Set_op = findcandidateOp(G', D*, H_G);
5      while Set_op ≠ ∅ do
6          operation p = Set_op.min_op();
7          execute(p, G', H_G);
8          Set_op = findcandidateOp(G', D*, H_G);
9      if G' is not k-anonymized then D*=refine(D*,G');
10 return G';
```

4.1 Basic Idea and Algorithm Framework

The optimal k-anonymization problem (i.e. k-anonymization with minimum utility loss) on social networks is NP-hard. [1] To simplify the problem, we assume the utility loss is affected by the number of edge operations performed and the utility loss caused by each edge operation. In other words, we try to solve the problem by reducing the number of edge operations and meanwhile always performing the edge operations that cause smaller utility loss first. A greedy algorithm is designed accordingly.

The basic idea of our algorithm is as follows. Given a graph G, the attack model F and the privacy requirement k, we perform edge operations one at a time on G to achieve k-anonymity. To restrain the utility loss, we perform the edge operation that directs the current G towards its "nearest" k-anonymized graph and meanwhile causes the smallest utility loss. Here, "nearest" k-anonymized graph refers to the graph that satisfies k-anonymity with the smallest number of edge operations, which is denoted as G^* to facilitate our explanation. The knowledge of G^* is essential for our algorithm. However, G^* is unknown in advance and it is hard to locate. Given that forming G^* directly is not always possible, we try to estimate the local structure information of the vertices of G^* (e.g., the degrees and/or the degrees of the neighbors of each vertex) which, based on the given G, F and k, is possible. Then, according to the local structure information, a set of candidate edge operations are generated to lead G towards G^*.

Algorithm 1 sketches a high-level outline of our HRG based k-anonymization algorithm. It takes a graph G, its HRG H_G, attacker's background knowledge F and privacy parameter k as inputs, and outputs a modified graph G' that is k-anonymized and meanwhile has small utility loss. Initially, the algorithm sets G' to G, and sets D^* as an estimation of G^* based on G, F and k (lines 1-2). Thereafter, it generates a set of candidate edge operations, maintained

[1] The NP-hardness is proved by reducing the traditional set-packing problem [12] to the optimal k-anonymization problem.

by a set Set_{op} with the utility loss caused by each edge operation (line 4). At each step, it gets the edge operation which causes the smallest utility loss, performs that edge operation on G', at the same time, updates the corresponding connection probability on H_G, and then re-generates the candidate set based on the updated G' (lines 6-8). This process continues until Set_{op} becomes empty (lines 5-8). After performing all the identified candidate edge operations, there are two possible outcomes, i.e., the current G' is k-anonymized or not. In case G' still does not satisfy the privacy requirement, it means the k-anonymized graph which has the local structure information D^* is not achievable by the current executed operation sequence and we need to refine D^* via small adjustments and continue previous process (line 9). We would like to point out that when refining D^*, we only consider additive adjustment, i.e. adjust the graph via adding edges. Thus, in the worst case, G' will be modified towards a complete graph, which always satisfies the privacy requirement. Therefore, our algorithm is convergent.

As highlighted in Algorithm 1, there are three key components, i.e., estimation of local structure information, generation of candidate edge operations, and refinement of D^*. Each of them will be detailed in following subsections.

4.2 Estimating Local Structure Information

As pointed out earlier, we only focus on k-degree anonymization for presentation simplicity. In the following, we explain how to find a good estimation of the k-anonymized graph with smallest number of edge operations, i.e., G^*. Our approach is to perform the estimation on the local structure information based on *degree sequence*. Degree sequence D of a graph $G(V, E)$ is a vector of size $|V|$ with each element $D[i] \in D$ representing the degree of vertex v_i in G. We further assume the degree sequence is sorted by the decreasing order of its elements.

Given a graph G, its degree sequence D and k, we want to estimate the degree sequence D^* of its "nearest" k-degree anonymized graph G^*. We list some pre-knowledge that can guide the estimation. First, D^* shares equal size with D, because we only consider graph modification via edge insertion/deletion but not vertex insertion/deletion. Second, D^* must be k-anonymized since D^* is the degree sequence of a k-degree anonymized graph of G. In other word, for each element $D^*[i] \in D^*$, there are at least $(k-1)$ other elements sharing the same value as $D^*[i]$. Third, because that D^* is the degree sequence of the "nearest" k-anonymized graph of G, the L_1 distance between D^* and D should be minimized. Based on the above knowledge, we can employ the dynamic programming method proposed in [1] to find D^*. We ignore the detail due to space limitation.

4.3 Generating Candidate Edge Operation Set

Once D^* that represents the target local structure information is ready, we need to find candidate edge operations that convert G' to a k-anonymized graph with its degree sequence represented by D^*. Before we introduce the detailed algorithm, we first define three basic edge operations, i.e., *edge insertion*, *edge deletion*, and *edge shift*, denoted as $ins(v_i, v_j)$, $del(v_i, v_j)$, and $shift((v_i, v_j), v_k)$.

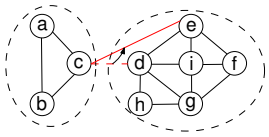


Fig. 4. $shift((v_c, v_d), v_e)$.

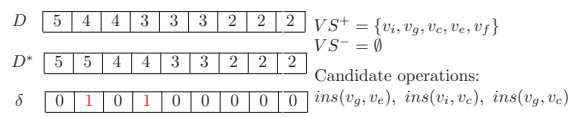


Fig. 5. HRG based 2-degree anonymization.

As suggested by their names, $ins(v_i, v_j)$ is to insert a new edge that links vertex v_i to vertex v_j and $del(v_i, v_j)$ is to remove the edge between v_i and v_j. Operation $shift((v_i, v_j), v_k)$ is to replace the edge $e(v_i, v_j)$ with edge $e(v_i, v_k)$. It is motivated by the observation that the HCE value is only sensitive to the number of the crossing edges between two communities. For example, as shown in Fig. 1(a), G is partitioned into two main communities as demonstrated by the dash circles. Edge $e(v_c, v_d)$ is the crossing edge connecting those two communities, and their lowest common ancestor is the root (based on H_G shown in Fig. 3). If we shift the end point v_d of the edge $e(v_c, v_d)$ to v_e (i.e., $shift((v_c, v_d), v_e)$) as illustrated in Fig. 4, it will not affect the connection probability of the root in H_G and hence HCE value, as the number of the crossing edge is not changed. Therefore, edge shift operation should receive a higher priority when modifying the graph to achieve k-anonymity. Definition 3 gives formal definition of this operation.

Definition 3 (Edge Shift). *Given a graph $G(V, E)$, the corresponding HRG H_G, an edge $e(v_i, v_j) \in E$, and a vertex $v_k \in V$ such that $e(v_i, v_k) \notin E$, let r be the lowest common ancestor of v_j and v_k on H_G, and assume v_i is not in the subtree of r. Edge shift $shift((v_i, v_j), v_k)$ is to replace $e(v_i, v_j)$ with $e(v_i, v_k)$.*

The goal of the edge operations is to modify the graph such that its degree sequence D' matches the target degree sequence D^*. Consequently, the difference sequence $\delta = (D^* - D')$ can give some guidance. For each element $\delta[i] \in \delta$ with $\delta[i] > 0$ (i.e. $D'[i] < D^*[i]$), it means a vertex in G' with degree $D'[i]$ needs to increase its degree, i.e., it should have more edges connected to. We maintain $D'[i]$ with $\delta[i] > 0$ via set DS^+ and maintain all vertices $v \in G'$ that have degree of $D'[i]$ via set VS^+ which includes all the vertices that may require edge insertion operation. Similarly, for each element $\delta[j] \in \delta$ with $\delta[j] < 0$ (i.e. $D'[j] < D^*[j]$), it means a vertex in G' with degree $D'[i]$ needs to decrease its degree, i.e., it should have less edges connected to. We maintain $D'[i]$ with $\delta[j] < 0$ via set DS^- and maintain all the vertices $v \in G'$ that have degree of $D'[j]$ via set VS^- which includes all vertices that may require edge deletion operation. Notice that the degree value of $D'[i]$ or $D'[j]$ may correspond to multiple vertices in G' and we treat them equally in our work. In addition, if the degree $D'[i]$ ($D'[j]$) only appears once in DS^+ (DS^-), we cannot perform edge insertion (deletion) to connect (disconnect) two vertices v_l, v_m both with original degree of $D'[i]$ ($D'[j]$) and hence we mark these vertices *mutual exclusive*, denoted as $EX(v_l, v_m) = True$.

Back to the graph G depicted in Fig. 1(a). Its degree sequence D and the target 2-degree anonymized degree sequence D^* are shown in Fig. 5. Based on

$\delta = (D^* - D)$, we find $\delta[2] = \delta[4] = 1 > 0$ and hence $D[2]$ $(= 4)$ and $D[4]$ $(= 3)$ are inserted into DS^+. Consequently, all the vertices in G with degree being 4 or 3 are inserted into $VS^+ = \{v_i, v_g, v_c, v_e, v_f\}$. Notice that all pair of vertices among of $\{v_i, v_g\}$ and among of $\{v_c, v_e, v_f\}$ are marked mutual exclusive. As there is no element of δ with its value smaller than 0, $DS^- = VS^- = \emptyset$.

The reason that we form VS^+ set and VS^- set is to facilitate the generation of candidate edge operations. As VS^+ set contains those vertices that need larger degree, a new edge connecting v_i to v_j (i.e., $ins(v_i, v_j)$) is a candidate, if $v_i, v_j (i \neq j) \in VS^+ \wedge e(v_i, v_j) \notin E' \wedge EX(v_i, v_j) \neq True$. We can enumerate all the candidate edge insertion operations based on VS^+ and preserve them in set Op^{ins}. Similarly, removing edge $e(v_i, v_j)$ (i.e., $del(v_i, v_j)$) forms an edge deletion operation, if $e(v_i, v_j) \in E' \wedge v_i, v_j (i \neq j) \in VS^- \wedge EX(v_i, v_j) \neq True$. Again, we explore all the candidate edge deletion operations and preserve them in set Op^{del}. We also consider the candidate edge shift operation. For a pair of vertices (v_j, v_k) with $v_j \in VS^- \wedge v_k \in VS^+ \wedge (j \neq k)$, if there is a vertex $v_i, (i \neq j, k)$ such that $e(v_i, v_j) \in E' \wedge e(v_i, v_k) \notin E' \wedge v_i$ is not in the subtree of v_j and v_k's lowest common ancestor on the HRG, $shift((v_i, v_j), v_k)$ is a candidate. All possible edge shift operations form another set Op^{shift}. We continue the above example shown in Fig. 5. As $VS^- = \emptyset$, we only need to consider possible edge insertion operations, i.e., $Op^{del} = Op^{shift} = \emptyset$. Based on $VS^+ = \{v_i, v_g, v_c, v_e, v_f\}$, we have $Op^{ins} = \{ins(v_g, v_e), ins(v_i, v_c), ins(v_g, v_c)\}$.

Given all the candidate edge operations maintained in the operation sets Op^{ins}, Op^{del}, and Op^{shift} respectively, we can insert them into the candidate operation set Set_{op} that is used by HRG-based k-anonymization algorithm (i.e., Algorithm 1). We sort Set_{op} by the increasing order of the HCE value changes caused by each operations, so that the edge operation that causes smaller utility loss will be performed earlier. Based on the HRG in Fig. 3, the corresponding Set_{op} is set to $\{\langle ins(v_g, v_e), 0.017\rangle, \langle ins(v_i, v_c), 0.033\rangle, \langle ins(v_g, v_c), 0.033\rangle\}$. The whole process of finding candidate operations is summarized in Algorithm 2.

4.4 Refining Target Local Structure Information

As mentioned above, our HRG-based k-anonymization algorithm generates D^* that estimates the local structure information of the "nearest" k-anonymized graph as the target, and performs edge operation to change graph towards D^*. However, it is possible that k-anonymized graph with degree sequence D^* is not achievable by the current executed operation sequence. If this happens, we need to refine D^* and start another round of attempt. To ensure the convergence of our algorithm, we only consider additive adjustment and we prefer that the new target degree sequence is close to that of the original D^*. The basic idea is to find a point on D^* to make adjustment and hopefully, after the adjustment, we can find executable candidate operations on G'.

In our work, we take VS^+ as a candidate set for the adjustable points. It contains the vertices that have not been k-anonymized and need to increase their degrees. For each $v_i \in VS^+$, we find $v_j \in V, (i \neq j)$, such that $e(v_i, v_j) \notin E'$, and $EX(v_i, v_j) \neq True$, and preserve $ins(v_i, v_j)$ via an operation set Op.

Algorithm 2: findcandidateOp algorithm

Input: $G'(V, E')$, D^*, H_G
Output: Candidate operation set Set_{op}

D' = degree sequence of G';
$\delta = (D^* - D')$;
$DS^+ = \{D'[i] \mid \delta[i] > 0, 1 \leq i \leq |D'|\}$;
$DS^- = \{D'[i] \mid \delta[i] < 0, 1 \leq i \leq |D'|\}$;
$VS^+ = VS^- = \emptyset$;
foreach $d \in DS^+$ **do**
  $VS^+ = VS^+ \cup \{v_i | v_i \in G', v_i.degree = d\}$;
foreach $d \in DS^-$ **do**
  $VS^- = VS^- \cup \{v_i | v_i \in G', v_i.degree = d\}$;
$Op^{ins} = getOp(VS^+, VS^+)$, $Op^{del} = getOp(VS^-, VS^-)$, $Op^{shift} = getOp(VS^+, VS^-)$;
calculate the cost of each operation in Op^{ins}, Op^{del}, and Op^{shift};
$Set_{op}.insert(Op^{ins}, Op^{del}, Op^{shift})$;
return Set_{op};

Within Op, we choose the $ins(v_i, v_j)$ that causes smallest utility loss. Notice that this operation changes degree of v_j even if the degree of v_j does not request adjustment. One simple method to address this issue is to change v_j's degree in D^* but it breaks the k-anonymity of D^*. Therefore, we increase the degree of v_j in the original G (i.e., the corresponding element in the degree sequence D is changed), and re-generate D^* based on the updated D. As the changes made to D are very small, the new D^* should be very similar as the old one.

We consider VS^+ first because we want to make additive change on D^*. However, if VS^+ is empty, we have to use VS^- that contains the vertices having not been k-anonymized and need to decrease their degrees. We can decrease the degrees of those vertices of VS^-, but it is against our goal of only making additive change. Alternatively, for a vertex $v_i \in VS^-$, we increase the degree of another vertex v_j, whose degree value is close to v_i according to G'. The rationale is that because v_i and v_j have similar degrees, they are very likely to have the same degree in the anonymized graph. Increasing the degree of v_j will cause the degree of v_i and v_j in the anonymized graph to be increased. In this case, v_i will not need to decrease its degree anymore. Still, we increase the degree of v_j on the seed degree sequence D instead of D^* by the same reason mentioned above.

5 Experimental Evaluation

In this section, we compare the utility loss of our HRG-based k-anonymization algorithm, referred as HRG, with the existing k-anonymization approaches that only consider minimizing the number of edge modifications. We choose k-degree anonymity as the privacy requirement, and use two existing k-degree anonymization algorithms proposed in [1] as competitors, namely *probing method* that

only considers edge addition operations, and *greedy swap method* that considers both edge addition and deletion operations. We refer them as Prob. and Swap respectively. We implemented all the evaluated algorithms in C++, running on a PC having an Intel Duo 2.13GHz processor and 2GB RAM.

We first examine the utility loss by measuring the HCE value change. In addition, we use some common graph structural properties to further evaluate the utility loss of different algorithms, such as clustering coefficient (**CC**), average path length (**APL**), and average betweenness (**BTN**) (see [11] for more information). We use function $PCR = |P - P'|/|P|$ to measure the property change ratio, where P and P' are the property value (i.e., HCE, CC, APL, or BTN) of the original graph G and the corresponding k-anonymized G' respectively.

Two real datasets are used in our tests, namely **dblp** and **dogster**. The former is extracted from dblp (`dblp.uni-trier.de/xml`), and the latter is crawled from a dog-theme online social network (`www.dogster.com`). We sampled subgraphs from these datasets with the size changing from 500 to 3000 respectively.

5.1 Utility Loss v.s. Graph Size

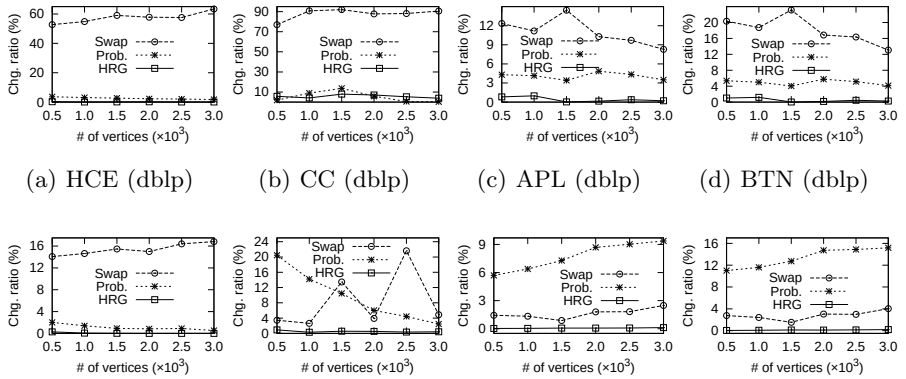
In our first set of experiments, we evaluate the impact of graph size in terms of number of vertices on the graph utility loss (i.e., HCE and other graph properties changes) under different k-anonymization methods. We set $k = 25$.

Fig. 6 shows the change ratio of different graph properties with different graph size of two datasets. Generally, our HRG method is most effective in terms of preserving graph properties. Take HCE value as an example. As depicted in Fig. 6(a) and Fig. 6(e), the change ratio of our method (i.e., HRG) is around 0.1% for both **dblp** and **dogster**, while that under Prob. is 0.5% for **dblp** and 3% for **dogster**, and that under Swap is 60% for **dblp** and 14% for **dogster**. The other example is APL value. As depicted in Fig. 6(c) and Fig. 6(g), as number of vertices increases from 500 to 3000, our HRG method causes around 0.4% and 0.07% utility loss on average for **dblp** and **dogster**, respectively. On contrary, Prob. causes 4% and 7% utility loss on average, and Swap causes around 11% and 1.6% utility loss for **dblp** and **dogster** datasets. All these observations verify that HRG model does successfully capture most, if not all, core features of the social network, as our HRG method which employs HRG model to represent graph feature is most effective.

5.2 Utility Loss v.s. k

In the second set of experiments, we evaluate the impact of k on the graph property change ratio of different k-anonymity methods. Here, the size of the graph is fixed to 2000 vertices.

Fig. 7 presents the results. We can observe that, in most cases, our HRG approach outperforms the others. As privacy requirement increases (i.e., k value increases), the utility loss under HRG and Prob. becomes more significant. This is because more edge operations are needed to achieve k-anonymity with large k under both methods. On the other hand, the utility loss caused by Swap algorithm



(a) HCE (dblp) (b) CC (dblp) (c) APL (dblp) (d) BTN (dblp)

(e) HCE (dogster) (f) CC (dogster) (g) APL (dogster) (h) BTN (dogster)

Fig. 6. Graph property change ratio v.s. the graph size.

is not affected by the change of k value that much. This is because, Swap method has to perform a large number of edge operations even for a small k. When k increases, the number of edge operations does not change much. [1]

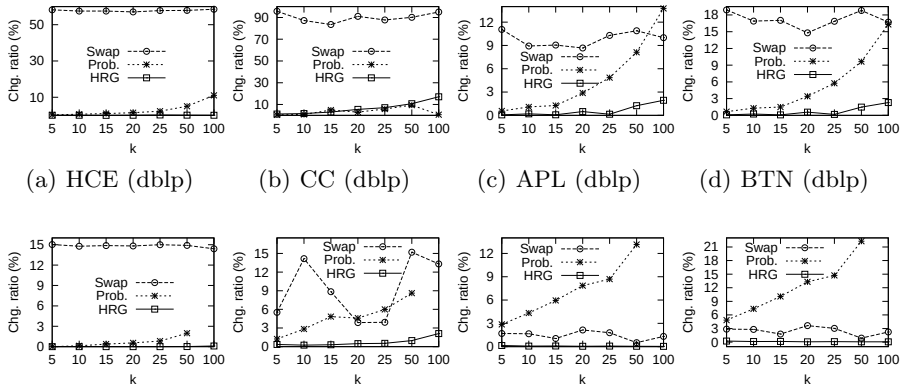
To sum up, our experiments use different graph properties to evaluate the utility loss, although our HRG method is developed based on HCE values. The experimental results clearly verify that our approach can generate anonymized social networks with much lower utility loss.

6 Conclusion

Privacy and utility are two main components of a good privacy protection scheme. Existing k-anonymization approaches on social networks provide good protection for entities' identity privacy, but fail to give an effective utility measurement, thus are unable to generate anonymized data with high utility. Motivated by this issue, in this paper, we propose a novel utility-oriented social network anonymization approach to achieve high privacy protection with low utility loss. We define a new utility measurement HCE based on the HRG model, then design an efficient k-anonymization algorithm to generate anonymized social network with low utility loss. Experimental evaluation on real datasets shows our approach outperforms the existing approaches in terms of the utility with the same privacy requirment.

Acknowledgment. This study was funded through a research grant from the Office of Research, Singapore Management University.

[1] Due to the extremely long converge time of the Prob. method, its results on the **dogster** graph with $k = 100$ were missing. However, it should not affect our observations of the experimental trend.



(a) HCE (dblp) (b) CC (dblp) (c) APL (dblp) (d) BTN (dblp)

(e) HCE (dogster) (f) CC (dogster) (g) APL (dogster) (h) BTN (dogster)

Fig. 7. Graph property change ratio v.s. k.

References

1. Liu, K., Terzi, E.: Towards Identity Anonymization on Graphs. In: SIGMOD'08. (2008) 93–106
2. Wu, W., Xiao, Y., Wang, W., He, Z., Wang, Z.: K-Symmetry Model for Identity Anonymization in Social Networks. In: EDBT'10. (2010) 111–122
3. Zhou, B., Pei, J.: Preserving Privacy in Social Networks Against Neighborhood Attacks. In: ICDE'08. (2008) 506–515
4. Zou, L., Chen, L., Özsu, M.: K-Automorphism: A General Framework for Privacy Preserving Network Publication. VLDB Endowment **2**(1) (2009) 946–957
5. Li, T., Li, N.: On the Tradeoff Between Privacy and Utility in Data Publishing. In: SIGKDD'09. (2009) 517–525
6. Hay, M., Miklau, G., Jensen, D., Towsley, D., Weis, P.: Resisting Structural Re-identification in Anonymized Social Networks. VLDB Endowment **1**(1) (2008) 102–114
7. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In: WWW'07. (2007) 181–190
8. Hay, M., Miklau, G., Jensen, D.: Anonymizing Social Networks. Technical report, UMass Amberst (2007)
9. Sweeney, L.: K-anonymity: A Model for Protecting Privacy. IJUFKS **10**(5) (2002) 557–570
10. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical Structure and The Prediction of Missing Links in Networks. Nature **453**(7191) (2008) 98–101
11. Costa, L.D.F., Rodrigues, F.A., Travieso, G., Boas, P.R.V.: Characterization of Complex Networks: A Survey of Measurements. Advances in Physics **56**(1) (2007) 167–242
12. Karp, R.M.: Reducibility among combinatorial problems. In Miller, R.E., Thatcher, J.W., eds.: Complexity of Computer Computations. Plenum Press (1972) 85–103