TECHNICAL REPORT

TR-CS-NMSU-3-20-2013

Yifan Hao
Huiping Cao
Kabi Bhattarai
Satyajayant Misra

Department of Computer Science
New Mexico State University

March 20, 2013

STK-Anonymity: K-anonymity for Social Networks Containing both Structural and Textual Information

Yifan Hao, Huiping Cao, Kabi Bhattarai, Satyajayant Misra
yifan@nmsu.edu, hcao@cs.nmsu.edu, kabi@nmsu.edu, misra@cs.nmsu.edu
Computer Science, New Mexico State University

Abstract

When social networks are released for analysis, individuals' sensitive information (e.g., node identities) in the network may be exposed. To avoid unwanted information exposure, social networks need to be anonymized before they are published. In the literature, a lot of approaches exist to anonymize social networks to prevent attacks by adversaries that know the network structures (e.g., node degrees, neighbors). However, these techniques cannot prevent the leakage of valuable identification information during social network analysis if the social network graphs contain both structural and textual information.

In this paper, we study the problem of anonymizing social networks to prevent individual identifications which use both structural (node degrees) and textual (edge labels) information in graphs. We introduce the concept of Structural and Textual (ST)-equivalence of individuals at two levels (strict and loose). We formally define the problem as Structure and Text aware K-anonymity of social networks (STK-Anonymity). In an STK-anonymized network, each individual is ST-equivalent to at least K-1 other nodes. The major challenge in achieving STK-Anonymity comes from the correlation of edge labels, which causes the propagation of edge anonymization. It has been shown in the literature that it is intractable to optimally K-anonymizing the label sequences of edge-labeled graphs. To address the challenge, we present a two-phase approach which consists of two heuristics in the first phase to process partial graph structures (node degrees) and a set-enumeration tree based approach in the second phase to anonymize edge labels. Results from extensive experiments on both real and synthetic datasets are presented to show the effectiveness and efficiency of our approaches.

1 Introduction

Online social networks, such as Facebook, MySpace, and LinkedIn for connecting with friends, family, and colleagues have grown explosively in the recent

few years. These social networks, when published, provide a lot of opportunities to study dyadic ties between individuals in these networks. The published social network data have been used to conduct meaningful analysis, such as understanding the community formation and evolution [3, 17], discovering topics, roles, authorities, and influential individuals in social networks [1,25], constructing multidimensional representation for further visualization [31], and so on. Such social network analysis uses rich network information [9] which consists of both structural information (i.e., network topology) and textual information associated to network nodes and edges. From the perspective of analyzing social networks to acquire useful knowledge, the more detailed data a released network contains, the more useful the network is for analysis purposes.

However, a big issue in publishing social network data is that we may expose individuals' private or sensitive information (e.g., salary, disease, connection to a specific group of people). A simple method used in the past, to address this issue, is to replace the true identity of the individuals in a network with random pseudo-identifiers before releasing the data. However, Backstrom et al. [2] and Narayanan et al. [28] have shown that this simple approach cannot prevent the identification of individuals when adversaries have background knowledge about the network. To protect people's privacy from different types of attacks from adversaries, various data anonymization methodologies [19,24,29,33,40,41] have been proposed. Among these, the K-anonymity technique (originally introduced in [33]) has been used to protect individuals by ensuring that each individual is indistinguishable from at least K-1 other individuals.

Several approaches have been proposed to achieve K-anonymity of individuals in social networks for preventing attacks using various topological information, such as node degrees [20], node neighborhood [14, 41], embedding subgraphs [7], or multiple structural information [43]. However, very few techniques (such as [22, 38, 39]) exist to anonymize a network considering *both the topological structure and the textual information of the network*. And, even these works have limitations, as we will illustrate in Section 2.

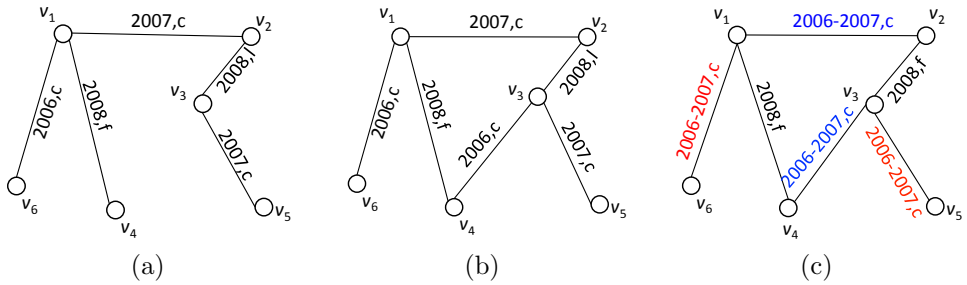


Figure 1: (a) the social network with two edge labels (the relation creation time and relation type where c, l, f represents friends from classes, friends leisure activities, and general friends respectively); (b) the 2-anonymity social network w.r.t. degree; (c) the 2-anonymity network w.r.t. degree and edge labels

The use of both topological (structure of the network) and textual infor-

mation (labels on the edges) is very common to analyze social ties [6, 9, 12]. For instance, to analyze the evolution of relationships in a network over time, we need to count the newly built relationships in different time periods (e.g., months, years, etc.) using a longitudinal dataset. However, such analysis may result in a higher probability of exposure of an individual's identity [9]. For example, given the network shown in Figure 1(a), if an adversary knows that Alice added *only* one friend before 2007, it can be inferred from the network that v_6 is Alice.

In this paper, we study the problem of protecting a node's identity [20, 43] by preventing attacks from an adversary who is armed with both structural (in particular node degrees) and textual (edge labels) information, and propose efficient techniques, which build upon the set-enumeration tree structure, for identity protection.

1.1 Problem and Challenges

We introduce the problem of STK-anonymization, which seeks to anonymize graph nodes and edges in a network graph, such that each node u in the graph is indistinguishable from K-1 other nodes in the graph, i.e., the probability of accurately identifying u is at most $\frac{1}{K}$, even for an adversary armed with the degree of nodes in the graph and labels information. Structural and textual K-anonymity of a node u in the graph implies that it is indistinguishable from K-1 other nodes in the networks not only in the degree, but also in the nature of the edge labels. For instance, in Figure 1(c), v_2 and v_4 are strictly indistinguishable from each other since they are both degree-2 nodes and have edge labels ((2006-2007,c), (2008,f)). We denote such a network anonymization as Structural and Textual K-anonymization and abbreviate it as STK-anonymization.

Our problem does not consider preventing attacks using only textual information associated to graph nodes. It is because traditional techniques on anonymizing tabular data (i.e., microdata) [16, 18, 33] can be directly applied by (1) treating each node's information as a row in a table and (2) applying attribute generalization schemes [16,18,33] which treat all values for a given attribute collectively (i.e., all values are generalized using the same unique domain generalization strategy).

In a social network graph, a node u and the labels on its adjacent edges are analogous to a row and cell values in tabular data. Theoretically, the K-anonymity techniques in tabular data can be applied to social networks. However, there are a couple of challenges.

Challenge 1: STK-anonymization of social networks is different from K-anonymizing microdata in that (1) each entity for anonymization in microdata is one tuple with a fixed number ($|QI|$, quasi-identifier) of values[1] while each entity for anonymization in social networks is a node with different ($d(v)$, degree of v) number of value lists (edge labels), (2) the value of one tuple in microdata is not correlated with other tuples; on the other hand, each node's edge labels

[1] A table with missing values is an exception.

in social network graphs are connected (hence correlated) with other nodes. Because the nodes in a network graph are correlated, directly applying tabular data anonymization techniques to anonymize a node may change the edge labels on its edges. Such changes of one node's edge label(s) may destroy the anonymization property of adjacent nodes which may be already anonymized. Thus, such anonymization can cause a back-propagation effect. For instance, the label changes of one edge $v_i \rightarrow v_j$ to achieve the anonymization of node v_i may automatically cause a change for v_j, which has already been anonymized. These two major differences make STK-anonymization a much more complicated problem than the problem of K-anonymizing microdata.

Challenge 2: The second challenge comes from the exponential number of ways in which edge labels can be anonymized. This search space explosion is analogous to that in the cell-based approach to anonymize microdata, which has been proved to be NP-hard [27]. In our problem setting, the back-propagation during anonymization makes things worse. In fact, Chester et al. [8] have proved that it is intractable to optimally K-anonymizing the label sequences of edge-labeled graphs. Existing techniques based on topological structures [7, 14, 20, 42] do not process the correlated edge labels; approaches on tabular textual information [16, 33] anonymize text, but do not deal with the text correlation and possible back-propagation. So, these methods cannot be directly used to achieve the STK-anonymization of a network in our problem setting.

To achieve the STK-anonymity of a network, the questions to answer are (1) how to choose which K nodes to be anonymized as a group and (2) how to anonymize these nodes when considering both node degrees and edge labels.

1.2 Contributions

The major contributions of this paper are:

- We formally define the problem of STK-anonymization based on different definitions of node ST-equivalence.
- We present a two-phase approach to achieve STK-anonymity of social networks. In the first phase, two heuristic algorithms are proposed to get degree anonymized graph.
- In the second phase, to avoid the problem of back-propagation during anonymization, we present a set-enumeration tree [16] based approach to achieve textual anonymity. In this phase, we further introduce three pruning strategies to improve the baseline approach.
- We conduct extensive experiments on real datasets and synthetic datasets to illustrate the performance of our presented techniques.

The rest of the paper is organized as follows. Section 2 reviews related work in the literature. Section 3 formally defines the STK-anonymity problem and related terminologies. Section 4 presents our two-phase approach to STK-anonymize a social network. Section 5 shows the experimental results of our proposed approaches. Finally, Section 6 concludes this paper.

2 Related Work

The K-anonymity technique [16, 18, 27, 33] is one of the pioneering and widely accepted scheme in the community of privacy protection of microdata (i.e., tabular data) where each row represents one entity's information. However, as discussed in the previous section, the approaches to achieve K-anonymity of microdata do not directly deal with correlation of tuple values.

In recent years, many techniques have been proposed to protect different types of sensitive information in social networks. According to the information that the privacy-protection techniques target to protect, these works fall into different categories: protecting nodes' identities [4, 7, 15, 20, 23, 26, 34–36, 36, 38, 39, 41, 42], nodes' sensitive labels [32], sensitive links [11, 21, 37, 40], and more complicated information [4, 9, 10, 13].

Our work falls into the category of node identity protection. So, we discuss the research in this area in more details. Several related work deal with attacks utilizing structural knowledge of vertices for identity disclosure, a concept, which was first introduced by Hay et al. in [15]. Several works [15, 20, 23, 36] have studied the K-degree generalization scheme, in which a node in the anonymized graph has the same degree has at least K-1 nodes in the graph. In these works, the adversaries are assumed to have knowledge of the node degrees. Zhou et al. in [41] and [42] considered the scenario where an adversary knows the 1-neighborhood structure of nodes (i.e., the structure formed by a node and all the nodes directly connected to this node). Their heuristic approach sorts graph nodes by applying an encoding schema on all the nodes based on their neighborhood information. Then, the nodes are anonymized in the sorted order. Cheng et al. in [7] dealt with a more complex scenario where the neighborhood structure over several hops is known to an adversary. Wu et al. [35] proposed a K-symmetry model to achieve privacy protection of vertices against attacks using any possible structural knowledge. Tai et al. in [34] introduced the problem of friendship attack, which may also lead to identity disclosure. In a friendship attack, adversaries utilize the knowledge about the degrees of two vertices connected by an edge. Bonchi et al. in [4] presented a novel information-theoretic analysis on utilizing randomization techniques for identity obfuscation and show that randomization techniques could meaningfully protect privacy while still preserving characteristics of the original graph. Medforth and Wang [26] proposed approaches to protect node privacy against attacks which use node degrees in a sequence of published graphs. All the above works assume that adversaries have knowledge about the topological structure. However, they do not consider the textual information related to the graph edges that adversaries may use to get private information.

Most recent works that define privacy based on textual information are [9, 38], and [39]. In [9], Cormode et al. introduced a new family of anonymization, called (k, l)-groupings for bipartite graphs. In [39], Yuan et al. defined three levels of user privacy protection by considering the node degree information and the textual information associated with both nodes and edges. However, in their problem definition, a small portion of the nodes fall into the category of

anonymization with respect to node degrees and edge labels. Thus, the process of this category is not critical. Yuan and Chen in [38] studied the problem of preventing node re-identification in social networks with weighted edges – the weights can be treated as a special type of textual (edge) information. In their work, a node was modeled to be a sequence of edge weights and two nodes are considered equivalent if the distance between their edge weight sequences is within a threshold. Liu and Yang [22] also studied the problem of graph anonymization over edge-weighted social networks. In comparison, our problem setting is more general in that we allow different types of text, instead of mere numerical weights, to label edges and nodes.

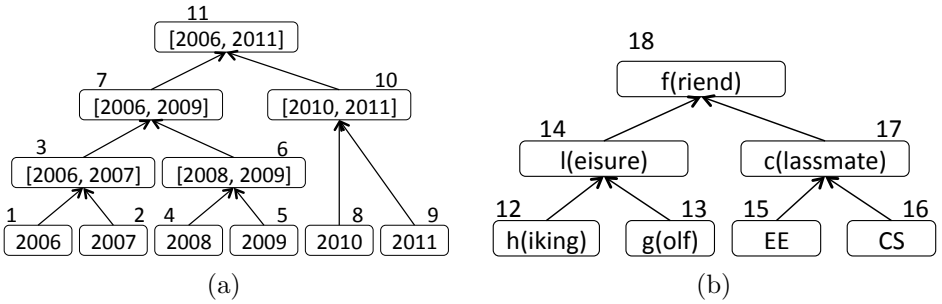
3 Problem Definition

In this section, we introduce and formalize the problem of Structure and Text aware K-anonymity (STK-anonymity). We also introduce the metrics that are used to measure the anonymization quality.

3.1 Structure and Text Aware K-Anonymity

Definition 3.1 (Annotated graph) *A graph $G(V, E, A)$, where $E \subseteq V \times V$, and V and E are alternatively represented as $G.V$ and $G.E$ respectively, is an annotated graph, if $A(.)$ applies textual information to every edge $e \in E$.*

The labels of each edge are from different domains D_1, $\cdots$, $D_{|D|}$, and the values in each domain can be numerical or categorical. These values may be partitioned according to specific criteria (such as equal-width), or be organized by experts as a hierarchy. In a hierarchy, we use $l_i \prec l_j$ to denote that l_i is a descendent of l_j. Figure 2 shows two hierarchies for temporal data and for people's relationships. In the temporal relationship, $2006 \prec [2006, 2007]$.



(a) (b)

Figure 2: Generalization hierarchies for (a) temporal information and (b) people's relationships. The number associated with each node is generated by the postorder traversal of the two trees.

Given a node v, its degree is denoted as $d(v)$, which is the number of edges connecting to v. Among the structural information (e.g., node degree, node

neighborhood) of social networks, the node degrees are the most general background information that an adversary can acquire.

The textual information of a d-degree node v, denoted as $A(v)$, is represented as a list of edge labels

$$A(v_i) = (A(e_{i1}), \cdots, A(e_{id}))$$

where $A(e_{ij}) = (l_{ij1}, \cdots, l_{ijx})$ (x is the number of edge labels for edge e_{ij}) represents the textual information attached to the j-th edge of v_i. For example, Figure 1(a) shows an annotated graph. The structure information of node v_2 is $d(v_2) = 2$. The textual annotation of node v_2 is a list of its edge labels $A(v_2)$=$((2007, c), (2008, l))$.

Definition 3.2 (Labels Compatibility) *$A(v) = (A(e_{i1}), \cdots, A(e_{id}))$ and $A(v') = (A(e'_{i1}), \cdots, A(e'_{id}))$ are compatible iff there is a one-to-one mapping $\mu : e_{ix} \leftrightarrow e'_{iy}$ from edges of v to edges of v' such that $\forall (l_i, l'_i)$ s.t. $l_i \in A(e_{ix}), l'_i \in A(e'_{iy})$, either $l_i \preceq l'_i$ or $l'_i \preceq l_i$.*

Given the domain hierarchies in Figure 2, $(2006, c)$ is compatible with $([2006, 2007], c)$ since $2006 \prec [2006, 2007]$ and $c \preceq c$. For the graph in Figure 1(b), $A(v_1)$ is compatible (although not the same) with $A(v_3)$ since $A(v_1) = ((2006, c), (2007, c), (2008, f))$ and $A(v_3) = ((2006, c), (2007, c), (2008, l))$, where the labels for the first two edges are the same, and the labels for the last edge are compatible (c stands for classmates and l stands for leisure).

Definition 3.3 (ST-equivalent) *Given two vertices v and v' in an annotated graph G, v and v' are Structural and Textual (ST)-equivalent, denoted as $v \sim v'$, iff (1) $d(v) = d(v')$ and (2) $A(v)$ and $A(v')$ are compatible. When $A(v) = A(v')$, v and v' are* strictly *ST-equivalent to each other. Otherwise, they are* loosely *ST-equivalent.*

Given the annotated graph in Figure 1(b), $v_1 \sim v_3$ (loosely) because their degrees are the same and annotations are compatible. However, $v_5 \nsim v_6$ although their degrees are the same because their annotations are incompatible.

Definition 3.4 (STK-anonymity problem) *Given a positive integer K and an annotated graph $G(V, E, A)$, the problem of Structure and Text (ST) aware K-anonymity is to construct another annotated graph $G'(V', E', A')$ with a map function μ such that*

- *$\forall v \in V$, $\exists v' \in V'$ s.t. $\mu(v) = v'$, where v' can be null, referring to a suppressed node;*
- *$\forall e \in E$, $\exists e' \in E'$ s.t. $\mu(e) = e'$, where e' can be null, referring to a suppressed edge;*
- *$\forall v' \in V'$, there exists at least K-1 other nodes $v'' \in V'$ s.t. $v'' \sim v'$. I.e., $\forall v' \in V'$, $|\{v'' | v'' \sim v', v'' \neq v'\}| \geq K-1$.*

3.2 Information Loss

We use two operations widely to anonymize a graph G to another graph G', thus achieving STK-anonymity. The first operation is *edge addition*, which not only changes the node degree of graph G, but also adds new edge labels (in particular, from *null* to a particular label). The second operation is the modification of labels for existing edges. A widely accepted option to modify labels is *generalization* [33], which changes a specific value to a more general value by applying hierarchical domain knowledge.

During graph anonymization, the exact information in the original graph G is changed (or more accurately, lost). So, a desired property is to lose as little information as possible, to preserve the usability of the anonymized graph. To measure the information loss in anonymizing a graph, intuitively, the edge change for generalizing existing edge labels should incur less information loss compared with adding a new edge to the new graph since the latter operation also adds related edge labels. To achieve this intuitive intention, we define information loss at different levels (from finer granularity to coarser granularity) for (1) individual edge label l_{ijk}, (2) edge annotation $A(e_{ij})$, (3) node annotation $A(v_i)$, and (4) the whole graph G. In our description, the generalization hierarchies in Figure 2 are used for illustration purpose.

Given a domain D_x, two elements l_i and l_i' in domain D_x, the information loss in moving from l_i to l_i' is

$$\phi(l_i, l_i') = \frac{mhop(l_i, l_i')}{|D_x|}$$

where $mhop(l_i, l_i')$ is the number of edges in the shortest path from the tree node with label l_i to the tree node with label l_i' obtained by treating the generalization hierarchy tree as an undirected graph and $|\cdot|$ represents the cardinality.

Information loss for generalizing a special label *null* to a specific label l in the tree is

$$\phi(null, l) = \phi(l, null) = \frac{mhop(null, l)}{|D_x|}$$

The generalization of the special label *null* to a node label in the generalization graph can capture the *structural information loss* incurred in adding a new edge or suppressing an existing edge. Intuitively, this cost should be more than generalizing any existing edge label, which is bounded by $\frac{h_x}{|D_x|}$, where h_x is the height of the hierarchy tree. To characterize this information loss, we define this to be a factor of the worst case cost $\frac{\theta \cdot h_x}{|D_x|}$ $(\theta > 1)$. This definition guarantees the desired property that the information loss of anonymizing existing edges should be less than that of adding new edges.

With the information loss of generalizing each individual label, we define the information loss of anonymizing edges and nodes by applying the following rules.

The information loss of anonymizing one edge annotation $A(e)$ to another

edge annotation $A'(e)$ is defined as

$$\phi(A(e), A'(e)) = \frac{\sum_{i=1}^{|A(e)|} \phi(l_i, l'_i)}{|A(e)|}$$

where $A(e) = \{l_1, \cdots, l_{|A(e)|}\}$ and $|\cdot|$ is the cardinality. For example, the information loss of anonymizing an edge label $(2006, CS)$ to $([2006, 2007], c)$ is the averaged cost from 2006 to $[2006, 2007]$ $(\frac{1}{11})$ and from CS to c $(\frac{1}{7})$. I.e., it is $\frac{1/11+1/7}{2} = \frac{9}{77}$.

The information loss of anonymizing one annotation of a vertex $A(v)$ to another annotation $A'(v)$ is the averaged cost of changing each label in $A(v)$ to the corresponding edge label in $A'(v)$. I.e.,

$$\phi(A(v), A'(v)) = \frac{\sum_{i=1}^{d(v)} \phi(A(e_i), A'(e_i))}{|d(v)|}$$

Since $A(v)$ encodes the node degree information, $\phi(A(v), A'(v))$ can capture the information loss incurred due to changes in both node degrees and edge labels.

Definition 3.5 (Info. loss of graph anonymization) *The information loss of STK-anonymizing G to G' is*

$$\phi(G, G') = \sum_{v_i \in V, v'_i = \mu(v_i)} \phi(A(v_i), A'(v'_i)) \quad (1)$$

Definition 3.6 (Optimal solution) *The optimal anonymization solution sol_{opt} for G is defined as*

$$sol_{opt} = \{sol_x | \phi(G, sol_x) = min_i(\phi(G, sol_i))\}$$

4 Solution framework

As we discuss in Section 1, the major challenges in STK-anonymizing social networks come from the correlation between edge labels and the large number of possible candidates. To address these challenges, we propose a two-phase solution framework to achieve the STK-anonymity of a social network.

Before introducing our solution framework, we explain in more detail the *anonymization back-propagation* issue caused by edge correlations if a naive approach is applied to anonymize graphs. In particular, this naive approach first partitions graph nodes into groups so that the nodes in one group have similar $A(\cdot)$ values. Then, it anonymizes edge labels for the nodes in each group. To anonymize the nodes in one group, changes are needed in the edge labels (through generalization) to make them loosely or strictly equivalent. The change to one node's edge label can propagate to its neighbor nodes, which propagate this change further. Inevitably, such changes can propagate back (back-propagation) to the already anonymized nodes and trigger repeated anonymization.

Example 4.1 *To achieve strict ST2-anonymity of the network in Figure 1(b), the nodes are partitioned into three groups:* $\{v_1, v_3\}$ *(degree 3),* $\{v_2, v_4\}$ *(degree 2), and* $\{v_5, v_6\}$ *(degree 1). Let us first anonymize* v_1 *and* v_3 *by changing the edge labels between* v_2 *and* v_3 *to* $(2008, f)$*. After this change, the edge labels for this group of nodes are the same* $A(v_1) = A(v_3) = ((2006, c), (2007, c), (2008, f))$*. Next, we move to the group* $\{v_2, v_4\}$ *to anonymize the edge labels* $A(v_2) = ((2007, c), (2008, f))$ *and* $A(v_4) = ((2006, c), (2008, f))$*. Since 2006 and 2007 can be generalized to the range* $[2006, 2007]$*, we get* $A(v_2) = A(v_4) = (([2006, 2007], c), (2008, f))$*. This change is reflected in labels of edges between* v_1 *and* v_2 *and between* v_3 *and* v_4*. As a result, this edge label change propagates back to* v_1 *and* v_3*. This propagation, however, results in the already anonymized* v_1 *and* v_3 *not satisfying ST2-anonymization anymore (with* $A(v_1) = ((2006, c), ([2006, 2007], c), (2008, f))$ $A(v_3) = ((2007, c), ([2006, 2007], c), (2008, f))$*. Thus, they need to be re-anonymized to get the anonymized network shown in Figure 1(c). In this example, the anonymization of* $\{v_2, v_4\}$ *is propagated back to the already anonymized node group* $\{v_1, v_3\}$*. When the network is larger, the effect of such back-propagation may be worse.*

We design a two-phase approach to alleviate the issue of anonymization back-propagation of edge labels. In our approach, the first phase aims to achieve edge degree anonymization of the given graph. This phase addresses the edge correlation issue at a much simpler level (on node degrees). The second phase anonymizes edge labels of graphs by utilizing a *global paradigm* to avoid the back-propagation of edge label changes. This framework works for both strict and loose ST-equivalence definition.

4.1 Degree anonymization

The first phase to achieve STK-anonymity of a graph is to perform edge degree anonymization. The purpose of this step is to add as few edges as possible to the graph G to create graph G' such that each node in G' has the same degree as at least K-1 other nodes. Intuitively, to add few edges to G, the nodes with similar original node degrees should be anonymized to have the same degree (i.e., belong to an anonymized group). Based on this intuition, the degree anonymization approach first sorts nodes in the descending order of their original node degrees such that nodes with similar degrees are closer to each other. We use S_d (listed in Table 1) to denote this sorted node degree sequence. From S_d, we introduce two heuristics to achieve the degree anonymization.

4.1.1 Heuristic based on dynamic programming

Liu et al. in [20] presented a dynamic programming based approach to achieve degree anonymization. This approach first applies a dynamic programming process on the sorted node degree sequence S_d to generate another degree sequence $\hat{S}_d$. This new degree sequence has two properties. First, each node is guaranteed to be K-*anonymous* in degree (i.e., each distinct node degree occurs at least K

S_d	The sequence with sorted node degrees for the original graph G. $S_d = (S_d(v_1), \cdots, S_d(v_n))$
$\hat{S_d}$	The sequence with node degrees for an anonymized graph G' where the order of nodes are the same to that for S_d. $\hat{S}_d = (\hat{S_d}(v_1), \cdots, \hat{S_d}(v_n))$
ΔS_d	The sequence of newly added node degrees when anonymizing G to G'. $\Delta S_d = S_d - \hat{S_d} = (S_d(v_1) - \hat{S_d}(v_1), \cdots, S_d(v_n) - \hat{S_d}(v_n))$
ΔV	The set of nodes with positive $\Delta S_d(v)$.

Table 1: Notations to describe node-degree sequences

times in $\hat{S_d}$). Second, it has *minimal* number of added degrees among all possible K-anonymous degree sequences. Thus, it incurs the least information loss from G. From $\hat{S_d}$, a new graph G' can be constructed such that $G(E) \subseteq G'(E)$ (i.e., G' covers the original graph G) and the nodes of G' have the new degree sequence $\hat{S_d}$.

However, there may not exist a proper valid graph for degree sequence $\hat{S_d}$. When this happens, $\hat{S_d}$ is said to be unrealizable.

We identified that $\hat{S_d}$ is unrealizable in two situations. In the first situation, the sum of the newly added degrees ($\sum_{i=1}^{n} \Delta S_d(v_i)$) is an odd number. This is possible since the basic dynamic programming routine uses node degree, instead of an edge which contributes two to node degrees, as a basic unit in its operation. However, the odd total node degree cannot come from a valid graph which always has even-number of node degrees ($2 \times |E|$). Hence, when $\sum_{i=1}^{n} \Delta S_d(v_i)$ is odd, $\hat{S_d}$ is unrealizable. In the second situation, there exists some node v_i ($\in \Delta V$) for which we cannot create $\Delta S_d(v_i)$ new edges. It is because v_i is already connected to some nodes in ΔV, and the number of nodes in ΔV which do not have an edge to v_i is smaller than $\Delta S_d(v_i)$. To create $\Delta S_d(v_i)$ edges, the constructed graph G' needs to have multiple edges between some nodes, which is improper[2]. To construct G' from unrealizable $\hat{S_d}$, [20] introduced a probing scheme to add random degree noise to S_d and repeatedly run the dynamic programming routine until it can find a solution (a realizable $\hat{S_d}$). However, such probing can be very expensive (which is verified by our experiments).

We present a more efficient heuristic method *DP-heuristic* (Figure 3) to achieve degree anonymization. This heuristic utilizes the dynamic programming routine as the first step to get the initial node groups for degree anonymization. After the dynamic programming, DP-heuristic creates as many new edges as possible for nodes with positive ΔS_d (Step 4a). After Step 4a, if a node v still has positive $\Delta S_d(v)$, it means we cannot add any more edges between v and any other nodes in ΔV. We denote such nodes as *unrealizable nodes*. When there are unrealizable nodes, DP-heuristic finds other nodes that are not in ΔV to create edges for these unrealizable nodes (Step 4b).

Step 4b chooses a group which is already anonymized and increments the degree of every node in this group by one. In particular, when $\sum_{i=1}^{n} \Delta S_d(v_i)$ is odd, a group with odd number of nodes is chosen. Theorem 4.1, which will be explained shortly, guarantees that we can find such a group. On the other hand, when $\sum_{i=1}^{n} \Delta S_d(v_i)$ is even, we either choose a group with even number

[2]Multiple edges are combined to one topological edge with multiple edge labels.

DP-heuristic (G, S_d, K)

1. $\hat{S}_d$ = DP(S_d, K)
2. $\Delta S_d = S_d$ - $\hat{S}_d$
3. ΔV = nodes with $\Delta S_d(v) > 0$
4. While (ΔV is not empty)
   (a) For (each node v in ΔV)
      i. Randomly pick a node v_j from ΔV such that (v, v_j) is not an existing edge
      ii. Add an edge between v and v_j to G
      iii. Decrease one from both $\Delta S_d(v)$ and $\Delta S_d(v_j)$
      iv. If $\Delta S_d(v) = 0$, remove v from ΔV
      v. If $\Delta S_d(v_j) = 0$, remove v_j from ΔV
   (b) If ΔV is not empty
      i. Choose a proper group g with anonymized nodes
      ii. For every $u \in g$, increment $\Delta S_d(u)$ by one and put these nodes into ΔV
5. Return G;

Figure 3: DP-heuristic to anonymize edge degrees based on dynamic programming

of nodes, or we choose two groups with odd number of nodes. With either of the above processes, $\sum_{i=1}^{n} \Delta S_d(v_i)$ after Step 4b will be even. This is to avoid the first unrealizable situation for $\hat{S}_d$.

Theorem 4.1 *In anonymizing S_d to sequence $\hat{S}_d$, if $\sum_{i=1}^{n} \Delta S_d(v_i)$ is odd, there exists a node group in $\hat{S}_d$ with odd number of nodes.*

Proof Sketch: Let there be gn anonymized groups in G', and the sum of node degrees in each group g_i be $d(g_i)$. Then, we have $\sum_{i=1}^{n} \hat{S}_d(v_i) = \sum_{i=1}^{gn} d(g_i)$. We want to show that when $\sum_{i=1}^{n} \hat{S}_d(v_i)$ is odd, there exists an anonymized node group g_i such that the number of nodes in it ($|g_i.V|$) is odd.

First, since $\sum_{i=1}^{n} \hat{S}_d(v_i) = \sum_{i=1}^{gn} d(g_i)$, there must exist at least one g_i such that $d(g_i)$ is odd when $\sum_{i=1}^{n} \hat{S}_d(v_i)$ is odd. Otherwise (i.e., every $d(g_i)$ is even), $\sum_{i=1}^{gn} d(g_i)$ must be even.

Second, if $d(g_i)$ is odd for an anonymized node group g_i, the number of nodes $|g_i.V|$ in this group is odd. This comes from the property of an anonymized node group g_i, for which all the nodes share the same degree $d(v_{g_i})$. I.e., $d(g_i) = d(v_{g_i}) * |g_i.V|$. When $d(v_{g_i})$ is odd, then both the node degree $d(v_{g_i})$ and $|g_i.V|$ are odd. ■

Let us use the graph in Figure 1(a) to illustrate this phase. The edge degrees are: 3 2 2 1 1 1 for vertexes v_1 v_3 v_2 v_4 v_5 v_6 in sequence. To achieve 2-anonymity of node degrees, we add one edge between v_3 and v_4. The new node degree sequence becomes 3 3 2 2 1 1 and it corresponds to the new graph G' in Figure 1(b).

Time complexity: The time complexity of the DP-heuristic is determined by the dynamic programming routine (Step 1) and the construction of the graph G' (Step 4). The dynamic programming routine's complexity is $O(nK)$ as shown in [20]. The construction of the graph G' has worst case complexity $O(n^2K^2)$, which is elaborated in detail here.

The graph reconstruction (Step 4a) from the *initial* $\hat{S}_d$ needs time in $O(|\Delta V|^2)$ because it examines every node in ΔV, and for each node in ΔV, it adds at most $|\Delta V|-1$ edges connecting to other nodes in ΔV. The factor $|\Delta V|$ in the worst case is $n-\lfloor \frac{n}{K} \rfloor$. In particular, the algorithm can generate at most $\lfloor \frac{n}{K} \rfloor$ anonymization groups when each group has the minimum number of nodes of K. For each group, all the node degrees need to be changed to be the same as the first node's degree. Thus, in the worst case, the degrees of all nodes except the first one in each group need to be updated. In this worse case, $|\Delta V| = n-\lfloor \frac{n}{K} \rfloor$. So, the construction of the initial $\hat{S}_d$ has the worst-case complexity $O(n^2)$. However, in real applications, the number of edges that need to be added (i.e., $\Delta S_d(v)$) for each node in ΔV is much smaller than n, and the algorithm shows linear (to n) performance.

When the initial $\hat{S}_d$ is unrealizable, the construction of G' needs more iterations. This leads to time complexity $O(I|\Delta V|^2)$. Different from the first iteration, in these later iterations ΔV is restricted to contain nodes in one or two anonymization groups, so $|\Delta V|$ is bounded by $2K$. In addition, the number of iterations is bounded by n^2, which is the worst case that the graph is anonymized to a complete one. So, the worst-case complexity is $O(n^2K^2)$. Nevertheless, in real applications, we just need to add one edge (not $|\Delta V|$ edges) for every node newly added to ΔV. So, the algorithm runs linear to $|\Delta V|$. In addition, the algorithm can stop before reaching a complete graph, thus it needs much less iterations. In our experiments, the algorithm shows linear performance in both n and K, which is better than the worst case.

4.1.2 Heuristic based on greedy node grouping

DP-heuristic, like *DP*, may suffer from the repeated iterations to construct G if the initial $\hat{S}_d$ is unrealizable. We propose another node grouping based approach, *Group-heuristic*. This algorithm is shown in Figure 4. In this algorithm, the sorted nodes are first partitioned into different groups such that each group (except the last one) consists of K nodes. Then, all the groups are traversed and anonymized (Step 2). In this step, when anonymizing the nodes in one group, the degrees of each node should be increased to be the same as the first node's degree d_0. Once the degrees of a group's nodes are the same, this group is denoted as anonymized, and all its nodes are marked as "anonymized" (Step 2c).

The major operation in anonymizing a group's node is to add edges for every node v to make its degree the same as d_0 (Step 2b). To add an edge for v, the algorithm chooses another unanonymized node (Step 2(b)i). In case there are insufficient number of unanonymized nodes to create edges for v, anonymized nodes are randomly selected to create edges for v. Accordingly,

these anonymized nodes' groups are marked as unanonymized (for further re-anonymization).

Group-heuristic ($ONodes$, K)
/*$ONodes$: nodes sorted in descending order of their degrees*/

1. Partition all nodes in $ONodes$ into groups and mark each group as unanonymized
2. **For each** unanonymized group g
   (a) d_0 = degree of the first node in this group
   (b) For each node v in g such that $S_d(v) < d_0$
      i. If there are $d_0 - S_d(v)$ unanonymized nodes in V, randomly choose $d_0 - S_d(v)$ nodes and create edges between v and these nodes
      ii. Otherwise (i.e., there are insufficient number of unanonymized nodes to create edges for v)
      – Randomly select anonymized nodes v' to create edges between v and these nodes
      – Mark the group that each v' belongs to as unanonymized
   (c) Mark every node in g as anonymized

Figure 4: Group-heuristic to anonymize edge degrees based on node grouping

This approach is simple and easy to deploy. However, it may not generate the best anonymization solution. For instance, given the sequence of ordered node degrees 5 5 5 2 2 2 1 1 1 (for nodes v_1 to v_9 respectively) and we want to get a 2-anonymized graph. The best solution that can be returned from *Group-heuristic* (by adding 3 edges: between v_4 and v_7, v_4 and v_8, v_4 and v_9) is: 5 5 5 5 2 2 2 2 2, where the nodes in the same anonymized group is underlined together. However, there exists a better 2-anonymity degree sequence, which does not need to add any edge: 5 5 5 2 2 2 1 1 1.

Time complexity: This algorithm has the worst case complexity $O(n^2)$. In particular, for each node v we need to add $d_0 - S_d(v)$ edges where d_0 is the degree of the first node in v's group. In the worst case, the anonymized nodes need to be re-anonymized multiple times until its edge number reaches $n-1$, when the anonymized graph becomes a complete graph. Thus, the worst case complexity is $O(n^2)$.

4.2 Edge label anonymization

The second phase is edge label anonymization. In this phase, for each group of nodes with the same degrees, the nodes' edge labels are generalized such that they are compatible and the information loss caused by the generalization is minimal. Formally, given a group g of nodes $\{v_{i1}, \cdots, v_{ix}\}$ ($K \leq x < 2K-1$) with degree d, we need to calculate a new annotation $A'(\cdot)$ such that $\sum_{j=1}^{x} \phi(A(v_{ij}), A'(v_{ij}))$ is minimal when this new annotation is applied to all the edges of nodes in g. I.e., there does not exist another annotation A'' with $\sum_{j=1}^{x} \phi(A(v_{ij}), A''(v_{ij})) < \sum_{j=1}^{x} \phi(A(v_{ij}), A'(v_{ij}))$.

The problem of getting the optimal anonymization of edge labels for all the nodes in each group is extremely expensive. In particular, for every edge label of one node v_i, we need to find its mapping edge labels from other nodes in the same group. Theorem 4.2 shows that the exhaustive examination of edge label mappings is of complexity $O((d!)^K)$, which is prohibitively expensive even for small d and K.

Theorem 4.2 *Given a group with x nodes, each of which has node degree d, the total number of combinations of mapping all edge labels for all the nodes in this group is $(d!)^{x-1}$.*

Proof Sketch: Let the x nodes in a group be $\{v_1, v_2, \cdots, v_x\}$ and let the annotations be as following.

$$\begin{aligned} A(v_1) &= (A(e_{11}), A(e_{12}), \cdots, A(e_{1d})) \\ A(v_2) &= (A(e_{21}), A(e_{22}), \cdots, A(e_{2d})) \\ &\cdots \\ A(v_x) &= (A(e_{x1}), A(e_{x2}), \cdots, A(e_{xd})) \end{aligned}$$

To get the minimum annotation generalization for $A(v_1), \cdots A(v_x)$ in an exhaustive manner, we need to create mappings between all possible edge labels. For $A(v_1)$'s first edge labels $A(e_{11})$, we can choose one edge from $A(v_2)$ to map with it. The number of edge choices from $A(v_2)$ is $\binom{d}{1}$. Similarly, to choose an edge from $A(v_x)$ to map with $A(e_{11})$ we also have $\binom{d}{1}$ choices. Overall, the total number of choices to map $A(e_{11})$ is $\binom{d}{1}^{x-1} = d^{x-1}$. Then, for $A(v_1)$'s second edge, we have $(d-1)^{x-1}$ edge mapping choices. Similarly, for the last edge label of $A(v_1)$, we have 1^{x-1} choice. In total, the number of edge mapping combinations is $d^{x-1} \cdot (d-1)^{x-1} \cdots 1^{x-1} = (d!)^{x-1}$. ■

The above analysis shows that it is expensive to anonymize the edge labels of nodes in a group. In addition, as we have shown in the beginning of this section, if the edge labels of nodes are anonymized group by group, we need to consider the anonymization back-propagation issue caused by the correlation of edge labels. So, the method of anonymizing edge labels in a similar manner as that in *Group-heuristic* is not applicable.

The critical part to anonymize the edge labels of a graph is to find an annotation A' using which the graph can be anonymized with minimal information loss. A' is a many to one mapping by applying an anonymization rule to every distinct edge label el and generalizing it to another label el'[3]. We use the *source* and *target edge label set* to denote the set of els and the set of $A'(el)$s respectively.

We present a novel approach to anonymize graph edge labels by utilizing a global scheme over all the possible target edge label sets. This approach does

[3]In anonymizing microdata, the cell based anonymization strategy [27] uses a many to many mapping to perform the anonymization. I.e., the same label el may be generalized to multiple other labels. In this work, we do not consider the anonymization solutions with such finer granularities.

not start from the graph's node groups one by one. Instead, it starts from the edge labels' generalization hierarchies and examines all the *target label sets* for the graph.

4.2.1 Anonymizing graph edge labels with one global candidate

An annotation's *target label set* $C_{tgt} = \{l_1, \cdots, l_y\}$ is defined over the values from all the domains' value hierarchies. Every C_{tgt} *implicitly* includes the root values of all the domain hierarchies. Each l_i explicitly existing in C_{tgt} is a non-root value in its domain value hierarchy. Given a set of domain value hierarchies and an annotation target label set C_{tgt} defined on them, the annotation rule A' is derived as follows: every edge label el in graph G is generalized to the label el' in C_{tgt}, which is the lowest ancestor label of el (including itself) in its domain hierarchy. This annotation derivation rule generates a many to one mapping from target label sets to annotations. That is, for each C_{tgt}, there is only one corresponding annotation A'. For simplicity, we call the target label set of an annotation as annotation target. Without confusion, we interchangeably use annotation target and annotation in later descriptions.

Example 4.2 (Target label set and annotations) *Given the two domain hierarchies in Figure 2. Consider the derivation of annotation rules for the target label set* $C_{tgt1} = \{[2006, 2007], c(lassmate)\}$. *Given the edge label el=(2006,c) for the edge between* v_1 *and* v_6 *in Figure 1(a), the annotation rule for el to* C_{tgt1} *is to (1) generalize* 2006 *to* $[2006, 2007]$*; because* $[2006, 2007]$ *is the lowest ancestor for* 2006 *in* C_{tgt1}*, and (2) generalize c to itself; because c is the lowest ancestor for c in* C_{tgt1}.

For one annotation C_{tgt}, the algorithm *AnonyOneCand* (in Figure 5) presents a global-scheme based procedure to anonymize the edge labels of graph G', which is already K-anonymized in degree. If G' satisfies STK-anonymity condition after edge label anonymization using the annotation for C_{tgt}, C_{tgt} is treated as an annotation solution.

This global-scheme based algorithm consists of two major steps. First, it converts *all* the edge labels in graph G' using the candidate $C_{tgt} = \{l_1, \cdots, l_y\}$ (Step 2). Then (Step 3), after the edge label conversion, graph G' is examined for its STK-anonymity. If G' is STK-anonymized, the information loss is calculated.

In this algorithm, all the nodes are anonymized using a global annotation candidate C_{tgt}. Once C_{tgt} is fixed, there is only one way to anonymize the graph. Thus, with this method, we can avoid anonymization propagation.

Example 4.3 *We use an example to illustrate this process. Given the graph in Figure 1(b), and assume that its edge labels come from two domains with the value generalization hierarchies in Figure 2. From the domain value generalization hierarchies, we can get one anonymization candidate* $C = \{[2006, 2007], 2008, c(lassmate)\}$. *Implicitly, this candidate represents* $C_{imp} = ([2006, 2007], 2008, [2006, 2011], c, f(riend))$ *where* $[2006, 2011]$ *and* $f(riend)$ *do not explicitly appear in C because they are the root values*

AnonyOneCand (G, G', K, C_{tgt})

1. $\phi(G, G') = 0$;
2. Convert all the edge labels in graph G' to labels in annotation target C_{tgt}, the conversion rules are put into A'
3. **For each** group g in G' whose nodes are already K-anonymized in degree
   (a) *anonymized* = examine the edge label anonymity of nodes in g
   (b) if *anonymized* = $false$, **return (-1)**
   (c) Else
      i. Calculate the information loss of each node v in this group $\phi((A(v), A'(v))$
      ii. $\phi(G, G')=\phi(G, G')+\sum_{v \in g} \phi(A(v), A'(v)))$
4. **return** $\phi(G, G')$;

Figure 5: Edge label anonymization using one candidate target C_{tgt}

of two hierarchies. When this candidate is applied to the graph in Figure 1(b), we need to convert the edge labels:

- *The edge label* $(2006, c)$ *between* v_1 *and* v_6 *is converted to* $([2006, 2007], c)$ *because* $[2006, 2007]$ *is the lowest ancestor of* 2006 *in* C *and* c *is the lowest ancestor of itself.*
- *The edge label* $(2008, l(eisure))$ *between* v_2 *and* v_3 *is converted to* $(2008, f)$ *because* 2008 *is the lowest ancestor of itself and* f *is the lowest ancestor of* l*.*
- *In a similar way, all the edge labels of Figure 1 (b) can be converted and we can get the graph in Figure 1(c).*

To check the anonymity of a group of nodes in Step 3a, the algorithm randomly chooses one node in the group (say v_1) to probe other v_is by checking whether $A'(v_1)$ and $A'(v_i)$ are compatible. If $A'(v_1)$ is compatible with every $A'(v_i)$, then this group of nodes are edge label anonymized.

Time complexity: The worst case time complexity of this step is $O(n^2)$ where n is the number of nodes in G'. In particular, the major computation comes from two parts. The first part is the edge label conversion using the global candidate C_{tgt}. Obviously, this calculation is in $O(|E'|)$ where $|E'|$ is the number of edges in G'. In the worst case when G' is a complete graph (i.e., $|E'|=n(n-1)$), this complexity is $O(n^2)$. The second part of the calculation is to check whether each group of nodes is anonymized. Let a group g contain $|g.V|$ nodes ($K \leq |g.V| < 2K$) and d be the degree of each node in this group. The edge anonymity checking of each group then has complexity $O(dK)$. Since there are at most $\frac{n}{K}$ groups, this part of calculation would be $O(dn)$. In the worst case, i.e., d is $n-1$, which is very rare, this part has $O(n^2)$ complexity. Thus, the overall worst case computation is in $O(n^2)$.

4.2.2 Get optimal edge label anonymization

The above described component anonymizes a graph's edge labels with one annotation target. With this component, we can examine all the possible annotation targets to calculate the best anonymization solution (i.e., new annotations). In this section, we present our approach to traverse all the possible annotation targets. The annotation targets are enumerations of all the value combinations from all the domains. To efficiently manage this large space of annotation targets, we utilize the technique of set-enumeration tree, which was shown to be very good in representing the set of combined domain values [16,30]. Each node in a set-enumeration tree represents the target label set of one annotation candidate C_{tgt}. With this target label set, we can create an annotation A' to convert the original edge labels of a graph to labels represented by this node C_{tgt} using the procedure described in the previous subsection. All the possible target label sets of annotation candidates are enumerated through the set-enumeration tree.

The set-enumeration tree for the annotation targets can be constructed by applying a similar technique as that in [16]. To start with, all the domains are ordered sequentially and each domain is assigned with a series of sequential numbers. In particular, let D_i ($i \geq 1$) be the i-th domain with m_i partitions or m_i hierarchical nodes. The sequential numbers assigned to one domain D_i are $\sum_{l=1}^{i-1} m_l + 1, \cdots, \sum_{l=1}^{i} m_l$. Inside one domain, the sequential numbers are assigned to its values according to a total ordering of these values. When the values of a domain do not form any hierarchy, they are ordered in their partition values (e.g., alphabetically). When these values form a hierarchy, the hierarchy is traversed either breath-first or depth-first to acquire a total ordering.

Example 4.4 *Given the two domains in Figure 2, the first domain is assigned with sequential numbers* $1, \cdots, 11$*, and the second domain is assigned with sequential numbers* $12, \cdots, 18$*. For each domain, each value's sequential number, which is denoted in the figure, follows the post-order traversal of the hierarchy.*

EdgeAnony(G, G', K, $D_1, \cdots, D_{|D|}$)
/* G' is K-degree anonymized */

1. T = set-enumeration tree built from domains $D_1, \cdots, D_{|D|}$
2. Solution $G_{opt} = \varnothing$ with cost $\phi(G, G_{opt}) = +\infty$
3. **For each** node $vt \in T$ (pre-order traversal)
   (a) $\phi(G, G')$ = **AnonyOneCand** (G, G', K, $L(vt)$)
   (b) If $\phi(G, G') < 0$, **continue**; /* not a solution*/
   (c) If $\phi(G, G') < \phi(G, G_{opt})$, set $G_{opt} = G'$
4. Return G_{opt} and vt corresponding to G_{opt}

Figure 6: Framework for edge label anonymization

Figure 6 shows the algorithm to perform edge label anonymization by utilizing the set-enumeration tree. This algorithm first constructs the set-enumeration tree for all the candidate targets. Then, it traverses the set-enumeration tree and anonymizes the edge labels using each tree node vt,

whose corresponding label set is $L(vt)$. In particular, for each node in the set-enumeration tree, the algorithm *AnonyOneCand* in Figure 5 is applied to calculate the information loss and the annotation of anonymizing the edge labels with this annotation target vt. Among all the anonymization solution (with non-negative information loss), the one with the minimum cost is the final solution.

Optimality analysis. The above described approach returns the optimal solution of STK-anonymizing a graph g which is degree-anonymized. Let G' be an optimal solution, and let the set of its distinct edge labels be $EL(G')$. Since the set-enumeration tree enumerates *all the possible value combinations* over the symbols in the hierarchical trees $\Sigma = \Sigma_{D1} \cup \cdots \cup \Sigma_{D_{|D|}}$, the labels in $EL(G')$ must belong to one node in the set-enumeration tree. The above described procedure examines every possible target set. Thus, it returns the optimal edge label anonymization solution.

Time Complexity: Let $|T|$ be the number of nodes in the set-enumeration tree, the algorithm *EdgeAnony* needs to examine all these T nodes. Let m_i be the number of elements (or values) for domain D_i and m be the total number of elements in all the domains for a graph (i.e., $m = \sum m_i$). The nodes in the set-enumeration tree enumerates all the value combinations from all the domains. Then the number of nodes $|T| = 2^m$. As we discussed in the previous section, to examine the STK anonymity of one node, the worst case anonymization cost is $O(n^2)$. So, the overall worst case cost of the edge degree anonymization is $O(2^m n^2)$.

4.3 Pruning strategies

The number of nodes in a set-enumeration tree is 2^m where m is the number of domain labels. So, it is very expensive (thus impractical) to run this optimal edge label anonymization by examining each annotation target. In this section, we present two properties of the set-enumeration tree. Based on these properties, we introduce three pruning strategies to improve the running efficiency.

Property 4.1 (Downward closure property) *Given a set-enumeration tree, if a node vt in this tree does not generate an anonymization solution, all the nodes in vt's subtree cannot generate any solution either.*

This property is directly derived from the construction of the set-enumeration tree. For a node vt in a set-enumeration tree, its child node has one more label than it. Recall that $L(vt)$ is the set of labels corresponding to the set-enumeration tree node vt. Let vt_c refer to a child node of vt, then $L(vt_c) = L(vt) \cup l_i$, where l_i is a new label that does not exist in $L(vt)$. If the lowest ancestor of l_i in $L(vt_c)$ is l_j, then any edge label in the subtree of l_i (in its domain hierarchical tree) is converted to l_i but not the more general l_j. Thus, vt's child node vt_c is more specific than node vt. This more specific conversion rule ensures that there exists no anonymized solution with the use of the subtrees.

Pruning strategy 1: Property 4.1 shows that when a list of labels (representing a target label set) does not generate an anonymous solution, then another label list containing more labels than the current list cannot generate any solution. This property can be directly applied to prune the subtree of a node vt if vt does not generate a solution. So, when traversing the set-enumeration tree, if a node vt cannot generate an anonymization solution, all its children nodes do not need to be examined and can be safely pruned.

Pruning strategy 2: This pruning strategy is also based on the downward closure property. Let us call a node vt a solution node when it is able to generate an anonymization solution, which may not be optimal. Given solution node vt, all the other nodes $\{vt' | L(vt') \subset L(vt)\}$ should be solution nodes as well. However, such solution nodes will not improve vt's solution (i.e., anonymize the graph with less information loss). Because of this, when traversing the set-enumeration tree, we can prune a node if its label set is a subset of a solution node's label set.

Property 4.2 (Encompassment property) *Given the set-enumeration tree nodes vt and vt' with $L(vt) = \{\cdots, l_1, \cdots, l_x l, \cdots\}$ and $L(vt') = \{\cdots, l_1, \cdots, l_x, \cdots\}$. If l satisfies the condition that $subtree(l) = \cup_{i=1}^{x} subtree(l_i)$, then vt is redundant with respect to vt'. Here, $subtree(l_i)$ represents all the values in l_i's subtree (including l_i in its domain hierarchy.*

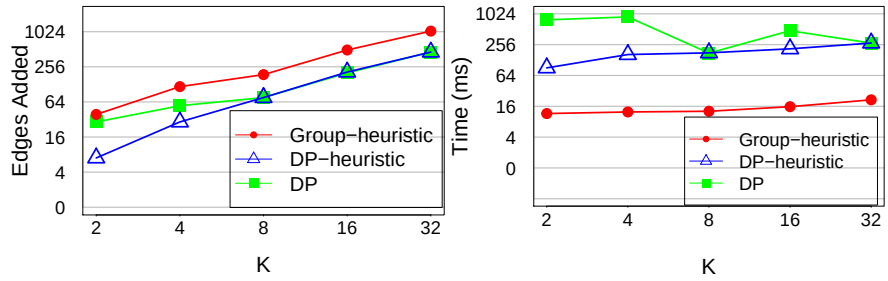
The condition $subtree(l) = \cup_{i=1}^{x} subtree(l_i)$ means that a label l and its descendant labels in its domain hierarchy appear together in the label list of the tree node vt. In addition, l's descendant labels can cover all the leaf nodes of l's subtree. During the process of the algorithm, each original graph edge label is converted to only one general label, which is its lowest common ancestor. Because l's descendant labels can cover all the leaf nodes of l's subtree, a leaf value in l's subtree is converted to l_i instead of l. This makes the annotation for $L(vt)$ the same to the annotation for $L(vt')$. Thus, vt is redundant.

Pruning strategy 3: This pruning strategy is based on Property 4.2. When examining the set-enumeration tree nodes, we first check the relationship of the labels represented by the tree nodes. When a tree node satisfies the condition in this property, this node and all its subtree nodes can be pruned. The added computation for this strategy is the checking of edge label relationships. To improve the efficiency of this computation, we can apply strategies for encoding tree nodes in the implementation.

5 Experiments

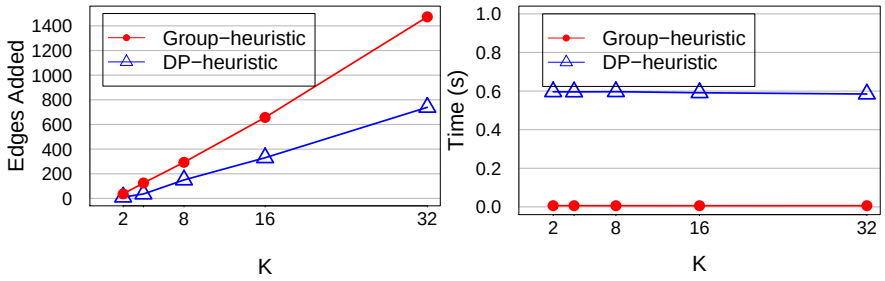
This section presents the experimental results on the performance of our proposed techniques. The approaches are implemented using C/C++. All the experiments were run on a Linux server (Ubuntu 11.04) with an Intel Xeon(R) CPU E5645 (@2.40GHz) and 4G RAM.

Both real datasets (Enron, arXiv, DBLP) and synthetic datasets are used in the experiments. The **Enron** dataset is from the email communications in a



(a) # of edges added (b) Time

Figure 7: Degree anonymization: Enron dataset

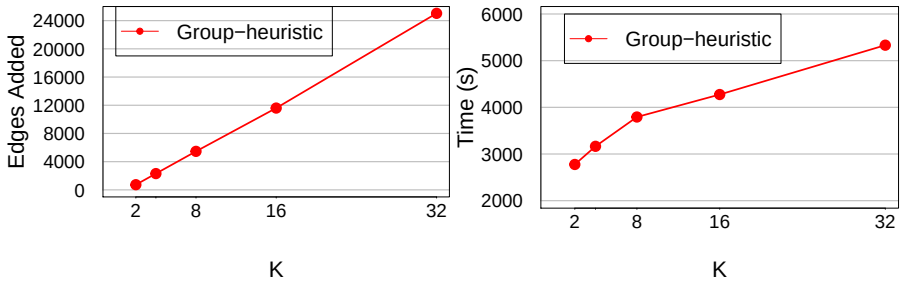


(a) # of edges added (b)Time

Figure 8: Degree anonymization: arXiv dataset

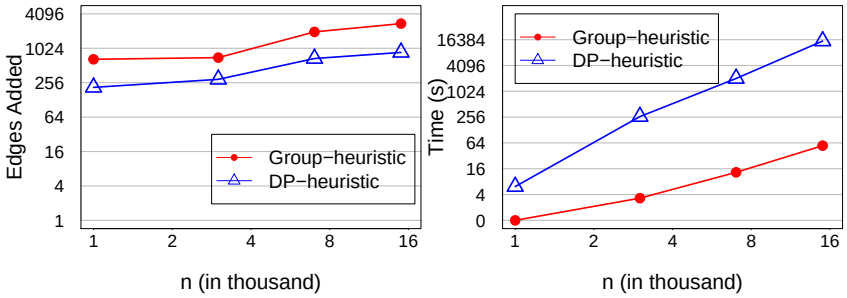
company (`www.cs.cmu.edu/~enron/`). It is a representative small dataset used in several other social network analysis and anonymization works [15, 20]. This dataset consists of 150 nodes (representing users) and 1180 edges (representing user communications) abstracted from the original $\sim 17K$ message exchanges. The **arXiv** dataset (`http://snap.stanford.edu/data/ca-GrQc.html`) is a collaboration network on general relativity and quantum cosmology from Stanford. This dataset initially contained 5242 nodes and 28980 edges. It was cleaned by merging its directed edges to indirected edges and by getting rid of self-loops. The final cleaned graph consists of 5230 nodes and 14470 edges. The **DBLP** dataset consists of the co-authorship relationships, extracted from the XML file generated on Sep. 7, 2012, at the DBLP website (`http://www.informatik.uni-trier.de/~ley/db/`). Our experiment uses the largest connected component in this graph, which contains $\sim 970K$ nodes and $\sim 4M$ edges. The **synthetic** datasets are generated using the R-MAT graph model [5]. This model creates graphs with two major properties of real social networks: the graphs show small-world characteristics and their vertex degrees follow power law distribution.

Edge labels are generated for these graphs. The Enron dataset has two real labels denoting the time of the first email communication and the total number of message exchanges. Besides these two real edge labels, the graphs are also associated with other different types (temporal, numerical, and categorical) of synthetic edge labels. Each type of edge labels comes from a domain with label



(a) # of edges added (b) Time

Figure 9: Degree anonymization: DBLP dataset



(a) # of edges added (b) Time

Figure 10: Degree anonymization: Synthetic dataset

generalization information organized in a hierarchical tree.

We test our techniques on degree anonymization and edge label anonymization separately due to the two-stage nature of our solution framework.

Degree anonymization. The set of experiments to test the degree anonymization techniques are shown in Figures 7-11. In these figures, *DP* represents the approach in [20] with recursive probing, *DP-heuristic* refers to our approach in Figure 3 which uses only dynamic programing for the initial step, and *Group-heuristic* denotes our heuristic approach in Figure 4 based on node grouping. We compare the different approaches using Enron, arXiv, DBLP, and synthetic datasets.

Figure 7 compares these three techniques using the Enron dataset by varying K. Comparing *DP* and *DP-heuristic*, we observe that the number of edges added through *DP-heuristic* is generally smaller than that through *DP*. Indeed, when K is smaller (e.g., 2, 4, 8) *DP-heuristic* adds much less edges than *DP*. This is because DP approach needs to invoke the probing many times and adds significant random noise. However, our *DP-heuristic* approach avoids adding noise to random nodes. With larger K, the numbers of added edges are similar (Figure 7(a)) because the degree sequence generated by the first activation of dynamic programing is very close to be realizable. Thus, the algorithms do not need any further iteration (for $K = 8, K = 32$) or need less number of iterations (for $K = 16$) to make the degree sequence realizable. The running

time (Figure 7(b)) of *DP* is dominated by the number of iterations, which are 24, 30, 0, 16, 0 for K values 2, 4, 8, 16, and 32 respectively. Thus, its running time fluctuates with different K values. The number of iterations does not play such an important role in *DP-heuristic*. DP-heuristic is more stable and generally faster than DP, for which we cannot control the number of probing times. Due to this, our later tests do not include *DP* for comparison. Among all these three approaches, *Group-heuristic* is the fastest approach, but it sacrifices the utility (with more added edges). More testing results on arXiv dataset (Figure 8) show the similar effect of K to different degree anonymization techniques.

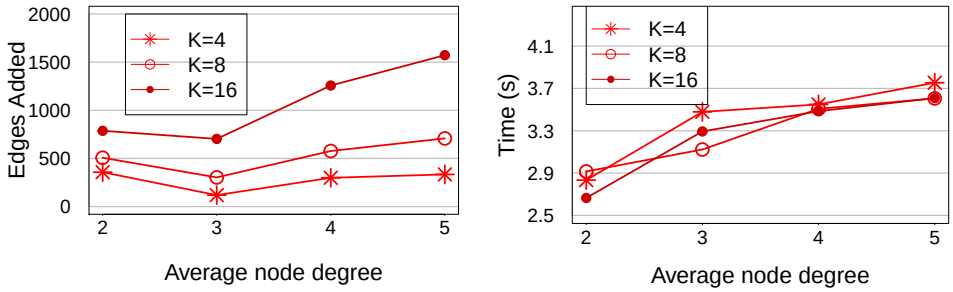
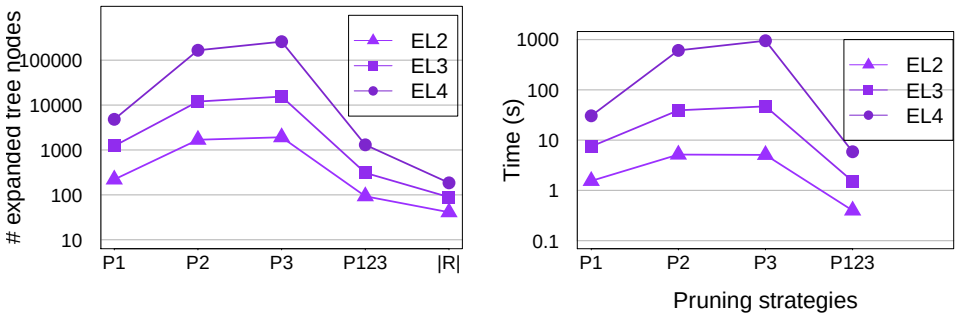


Figure 11: Degree anonymization: *Group-heuristic*, Synthetic dataset, fix $n = 5K$

We further evaluate the effect of *graph size* over the heuristic approaches by using synthetic data and DBLP data. Figure 10 shows the results of comparing *DP-heuristic* and *Group-heuristic* for synthetic data. For this test, we fix $K = 16$ and vary the graph size from ~250 nodes to ~15K nodes. These figures show that both these heuristic approaches grow linearly in the number of nodes (n) in the graph, which is much better than the worst case that we derived in Section 4.1.1. The test on the DBLP dataset (Figure 9) also shows a similar trend for *Group-heuristic* when K is varied. However, *DP-heuristic* approach is not even able to finish for this dataset and runs out of memory due to its quadratic space usage. This demonstrates the non-scalability of the dynamic-programming based approach.

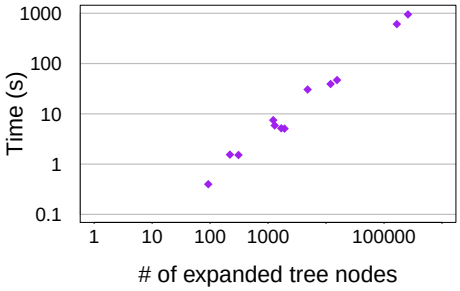
We also show the results of changing *the average node degrees* to anonymize the synthetic data, which are generated with the fixed number of nodes (5K). The results are reported in Figure 11. The number of added edges and the running time grow almost linear in the average node degrees. This experiment shows that, practically, we can achieve better performance than the worst case scenario.

Edge label anonymization. Our second set of experiments analyzed the effectiveness of the edge label anonymization techniques. First, we performed experiments to compare the effect of different pruning strategies using the Enron datasets. The results for different edge label settings are plotted in Figure 12. In these figures, i in ELi denotes the number of labels on each edge in the graph. $P1$, $P2$, and $P3$ represent the edge label anonymization approach by applying pruning strategy 1, 2, and 3 respectively. $P123$ applies all three pruning strate-



(a) $|R|$ is the number of tree nodes that generate a solution

(b) Different strategies



(c)

Figure 12: Edge label anonymization: Enron dataset, compare different pruning strategies: EL denotes the number of edge labels on each edge

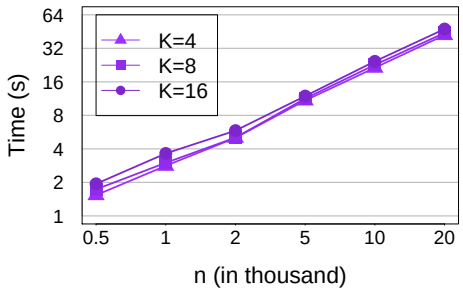


Figure 13: Edge label anonymization: Synthetic dataset ($EL2$)

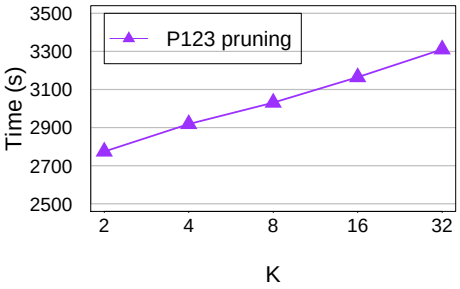


Figure 14: Edge label anonymization: DBLP dataset ($EL2$)

gies together. $|R|$ refers to the number of set enumeration tree nodes which can be the target label set of an STK-solution. Figure 12(a) shows that the pruning techniques ($P1, P2, P3, P123$) can dramatically reduce the number of nodes that need to be examined in comparison to the original (non-pruned) enumeration trees, which have 2^{11}, 2^{14}, and 2^{18} nodes respectively. Accordingly, these strategies reduce the running time as well (Figure 12(b)). The linear relationship between the number of expanded nodes and the running time is confirmed

in Figure 12(c).

We next test the effect of the *graph size* and K by using synthetic datasets and the DBLP dataset. We only plot the results for $P123$ and for graph with two edge labels on each edge (i.e., $EL2$). Figure 13 and 14 show that the running time increases linearly with n (for every K) and K (for fixed n with DBLP dataset). This is because the search space (i.e., the number of set-enumeration tree nodes) is fixed once the edge label domains are fixed and the execution time is mainly affected by the graph size.

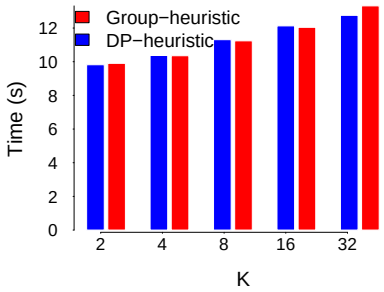


Figure 15: Edge label anonymization: arXiv dataset

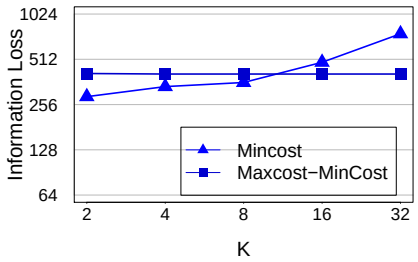


Figure 16: Information loss: Enron dataset

The *last set of tests* compare the running time of edge label anonymization after applying different degree anonymization approaches (*DP-heuristic* and *Group-heuristic*) on a graph. This test is performed on the arXiv dataset with two edge labels on each edge. Also, the $P123$ pruning strategy is applied. Figure 15 shows that the edge label anonymization time is almost the same for graphs which achieve K-anonymity on degrees by using *DP-heuristic* and *Group-heuristic*. This is consistent with the situation that edge label anonymization just needs to convert the edge labels and examine the textual anonymity. This time is the same when the search space is fixed. For larger K, the label anonymization of graphs after degree anonymization with *Group-heuristic* uses slightly more time because the *Group-heuristic* approach adds more edges (compared with *DP-heuristic*) to the graph to achieve degree anonymity.

Information loss. Besides the running time, we also measure the information loss of incurred in anonymization. Due to space limitation, we only report the information loss in anonymizing the Enron dataset for different K values. The *Mincost* line in Figure 16 represents the information loss. We also report the information-loss difference between a completely anonymized graph (i.e., all the edge labels are anonymized to their root label) and our solution graph. Such difference is shown through the *Maxcost-Mincost* line. This figure shows that the difference remains almost constant although the *Mincost* grows linearly in K. This is because the majority of the anonymization cost is introduced by edge addition, but not edge label generalization.

6 Conclusions

In this paper, we study the problem of achieving K-anonymity of social networks containing rich information to prevent attacks utilizing both structural (node degrees) and textual (edge labels) information. This problem is formulated as STK-anonymization. The major challenge in solving this problem is because of the correlation of edge labels. To address this challenge, we present a two-phase solution framework, which anonymizes edge degrees and edge labels in two phases. The first phase, for degree anonymization, leverages the edge correlation. In this phase, we introduce two heuristics, one based on a dynamic programming scheme and the other based on node grouping heuristic. In the second phase of anonymizing edge labels, we introduce a global scheme to avoid the calculation of $O((d!)^K)$ edge label mapping combinations, where d is the average node degree in a group of K nodes. This global scheme utilizes a set-enumeration tree structure to enumerate the global candidates. Furthermore, we introduce three pruning strategies to improve the efficiency without affecting the anonymization utility. We theoretical analyze the optimality and running time complexity for the techniques in different stages of this framework. We also experimentally show the applicability of the approach by using real and synthetic data.

References

[1] C. C. Aggarwal, A. Khan, and X. Yan. On flow authority discovery in social networks. In *SDM*, pages 522–533. SIAM/Omnipress, 2011.

[2] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190, 2007.

[3] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD*, pages 44–54, 2006.

[4] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. In *ICDE*, pages 924–935, 2011.

[5] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *SDM*, 2004.

[6] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu. Graph olap: Towards online analytical processing on graphs. In *ICDM*, pages 103–112. IEEE Computer Society, 2008.

[7] J. Cheng, A. W.-C. Fu, and J. Liu. K-isomorphism: privacy preserving network publication against structural attacks. In A. K. Elmagarmid and D. Agrawal, editors, *SIGMOD Conference*, pages 459–470. ACM, 2010.

[8] S. Chester, B. M. Kapron, G. Srivastava, and S. Venkatesh. Complexity of social network anonymization. *SOCIAL NETWORK ANALYSIS AND MINING, DOI: 10.1007/s13278-012-0059-7*, 2012.

[9] G. Cormode, D. Srivastava, S. Bhagat, and B. Krishnamurthy. Class-based graph anonymization for social network data. *PVLDB*, 2(1):766–777, 2009.

[10] S. Das, Ö. Egecioglu, and A. E. Abbadi. Anonymizing weighted social network graphs. In F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, editors, *ICDE*, pages 904–907. IEEE, 2010.

[11] A. M. Fard, K. Wang, and P. S. Yu. Limiting link disclosure in social network analysis through subgraph-wise perturbation. In *EDBT*, pages 109–119, 2012.

[12] J. Han, X. Yan, and P. S. Yu. Scalable olap and mining of information networks. In *EDBT*, page 1159, 2009.

[13] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.

[14] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and C. Li. Resisting structural re-identification in anonymized social networks. *VLDB J.*, 19(6):797–823, 2010.

[15] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *PVLDB*, 1(1):102–114, 2008.

[16] R. J. B. Jr. and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE*, pages 217–228, 2005.

[17] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *KDD*, pages 611–617, 2006.

[18] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *SIGMOD Conference*, pages 49–60, 2005.

[19] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

[20] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD Conference*, pages 93–106, 2008.

[21] L. Liu, J. Wang, J. Liu, and J. Zhang. Privacy preservation in social networks with sensitive edge weights. In *SDM*, pages 954–965, 2009.

[22] X. Liu and X. Yang. A generalization based approach for anonymizing weighted social network graphs. In *WAIM*, pages 118–130, 2011.

[23] X. Lu, Y. Song, and S. Bressan. Fast identity anonymization on graphs. In *DEXA (1)*, pages 281–295, 2012.

[24] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[25] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *IJCAI*, pages 786–791, 2005.

[26] N. Medforth and K. Wang. Privacy risk in graph stream publishing for social network data. In *ICDM*, pages 437–446, 2011.

[27] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS*, pages 223–228, 2004.

[28] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, pages 173–187, 2009.

[29] H. Park and K. Shim. Approximate algorithms for k-anonymity. In *SIGMOD Conference*, pages 67–78, 2007.

[30] R. Rymon. Search through systematic set enumeration. In *KR*, pages 539–550, 1992.

[31] A. J. Seary and W. D. Richards. Spectral methods for analyzing and visualizing networks: an introduction. In *Workshop Summary and Papers*, pages 209–228, 2000.

[32] Y. Song, P. Karras, Q. Xiao, and S. Bressan. Sensitive label privacy protection on social network data. In *SSDBM*, pages 562–571, 2012.

[33] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[34] C.-H. Tai, P. S. Yu, D.-N. Yang, and M.-S. Chen. Privacy-preserving social network publication against friendship attacks. In *KDD*, pages 1262–1270, 2011.

[35] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang. K-symmetry model for identity anonymization in social networks. In *EDBT*, pages 111–122, 2010.

[36] X. Ying, K. Pan, X. Wu, and L. Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In *SNAKDD*, page 10, 2009.

[37] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, pages 739–750, 2008.

[38] M. Yuan and L. Chen. Node protection in weighted social networks. In *DASFAA (1)*, pages 123–137, 2011.

[39] M. Yuan, L. Chen, and P. S. Yu. Personalized privacy protection in social networks. *PVLDB*, 4(2):141–150, 2010.

[40] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, pages 153–171, 2007.

[41] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515, 2008.

[42] B. Zhou and J. Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowl. Inf. Syst.*, 28(1):47–77, 2011.

[43] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. *PVLDB*, 2(1):946–957, 2009.