Knowl Inf Syst (2011) 28:47–77
DOI 10.1007/s10115-010-0311-2

REGULAR PAPER

The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks

Bin Zhou · Jian Pei

Received: 16 November 2009 / Revised: 28 April 2010 / Accepted: 31 May 2010 /
Published online: 16 June 2010

Abstract Recently, more and more social network data have been published in one way or another. Preserving privacy in publishing social network data becomes an important concern. With some local knowledge about individuals in a social network, an adversary may attack the privacy of some victims easily. Unfortunately, most of the previous studies on privacy preservation data publishing can deal with relational data only, and cannot be applied to social network data. In this paper, we take an initiative toward preserving privacy in social network data. Specifically, we identify an essential type of privacy attacks: neighborhood attacks. If an adversary has some knowledge about the neighbors of a target victim and the relationship among the neighbors, the victim may be re-identified from a social network even if the victim's identity is preserved using the conventional anonymization techniques. To protect privacy against neighborhood attacks, we extend the conventional k-anonymity and l-diversity models from relational data to social network data. We show that the problems of computing optimal k-anonymous and l-diverse social networks are NP-hard. We develop practical solutions to the problems. The empirical study indicates that the anonymized social network data by our methods can still be used to answer aggregate network queries with high accuracy.

Keywords Privacy · Social network · k-Anonymity · l-Diversity

A preliminary version of this paper appears as Zhou and Pei [49]. This research is supported in part by an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplement Grant. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

B. Zhou (✉) · J. Pei
School of Computing Science, Simon Fraser University, 8888 University Drive,
Burnaby, BC V5A 1S6, Canada
e-mail: bzhou@cs.sfu.ca

J. Pei
e-mail: jpei@cs.sfu.ca

1 Introduction

Recently, more and more social network data have been made publicly available in one way or another [1,2,21,22]. Preserving privacy in publishing social network data becomes an important concern. Is it possible that releasing social network data, even with individuals in the network anonymized, still breaches privacy?

1.1 Motivation example

With some local knowledge about individual vertices in a social network, an adversary may attack the privacy of some victims. As a concrete example, consider a synthesized social network of "friends" shown in Fig. 1a. Each vertex in the network represents a person. An edge links two persons who are friends.

Suppose the network is to be published. To preserve privacy, is it sufficient to remove all identities as shown in Fig. 1b? Unfortunately, if an adversary has some knowledge about the neighbors of an individual, the privacy may still be leaked.

If an adversary knows that Ada has two friends who know each other, and has another two friends who do not know each other, that is, the 1-neighborhood graph of Ada as shown in Fig. 1c, then the vertex representing Ada can be identified uniquely in the network since no other vertices have the same 1-neighborhood graph. Similarly, Bob can be identified in Fig. 1b if the adversary knows the 1-neighborhood graph of Bob.

Identifying individuals from released social networks intrudes privacy immediately. In this example, by identifying Ada and Bob, an adversary can even know from the released social network (Fig. 1b) that Ada and Bob are friends, and they share one common friend. Other information regarded as privacy can be further derived such as how well a victim is connected to the rest of the network and the relative position of the victim to the center of the network.

To protect the privacy, one attempt is to guarantee that any individual cannot be identified correctly in the anonymized social network with a probability higher than $\frac{1}{k}$, where k is a user-specified parameter carrying the same spirit in the k-anonymity model [35]. By adding a noise edge linking Harry and Irene, the 1-neighborhood graph of every vertex in Fig. 1d is not

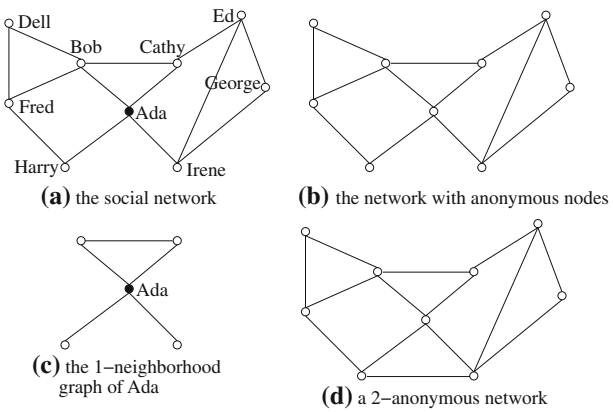


Fig. 1 Neighborhood attacks in a social network

unique. An adversary with the knowledge of 1-neighborhood cannot identify any individual from this anonymous graph with a confidence higher than $\frac{1}{2}$.

1.2 Challenges

Although privacy preservation in data publishing has been studied extensively and several important models such as k-anonymity [35] and l-diversity [27] as well as many efficient algorithms have been proposed, most of the existing studies can deal with relational data only. Those methods cannot be applied to social network data straightforwardly.

As elaborated in Sect. 1.1, privacy may be leaked if a social network is released improperly to public. In practice, we need a systematic method to anonymize social network data before it is released. However, anonymizing social network data is much more challenging than anonymizing relational data due to the following issues.

The first issue is that modeling background knowledge of adversaries and attacks about social network data is much more challenging than that about relational data. For relational data, a set of attributes form a quasi-identifier. The attackers can use the quasi-identifier to identify individuals from multiple tables. However, for social network data, it is much more complicated and much more difficult. The reason is that many pieces of information in a social network can be used to identify individuals, such as labels of vertices and edges, neighborhood graphs, induced subgraphs, and their combinations.

The second issue is that measuring information loss in anonymizing social network data is much more challenging than that in anonymizing relational data. The information loss in relational data can be measured tuple by tuple. Given one tuple in the original table and the corresponding tuple in the anonymized table, the distance between the two tuples is used to measure the information loss at the tuple level. The sum of information loss in individual tuples is used to measure the information loss at the table level. However, for social network data, there can be many different ways to define the measures of information loss and anonymization quality. For example, a social network consists of a set of vertices and a set of edges. Unlike the case of relational data, we cannot compare two social networks by simply comparing the vertices and edges individually, since two social networks may be quite different even if they have the same number of vertices and the same number of edges. Therefore, we need to consider more network-wise properties such as connectivity, betweenness, diameter, and network structure.

The third issue is that devising anonymization methods for social network data is much more challenging than that for relational data. For relational data, anonymizing a group of tuples does not affect other tuples in the table. Thus, divide-and-conquer methods are extensively used to anonymize relational data. However, divide-and-conquer methods may not be useful for social network data, since changing labels of vertices and edges may affect the neighborhoods of other vertices, and removing or adding vertices and edges may affect other vertices and edges as well as the properties of the network. Therefore, specific techniques have to be taken to anonymize social network data.

1.3 Contributions and organization

Privacy preservation in publishing social networks is a new challenging problem that cannot be solved completely by one shot. In this paper, we take the initiative to tackle the problem. We identify an essential type of privacy attacks in social networks: neighborhood attacks, which are illustrated in Sect. 1.1. To protect privacy against neighborhood attacks, we extend the conventional k-anonymity and l-diversity models from relational data to social network

data. We show that the problems of computing optimal k-anonymous and l-diverse social networks are NP-hard. We develop practical solutions to the problems. We conduct an empirical study, which indicates that the anonymized social networks generated by our method can still be used to answer aggregate network queries with satisfactory accuracy.

The rest of the paper is organized as follows. In Sect. 2, we model neighborhood attacks and extend the k-anonymity and l-diversity models to social networks. We review related work in Sect. 3. The practical solutions to k-anonymity and l-diversity are developed in Sects. 4 and 5, respectively. The proposed anonymization methods are examined empirically using both a real dataset and a series of synthetic datasets in Sect. 6. In Sect. 7, we discuss some other related privacy attacks in social networks for future work, and conclude the paper.

2 Problem definition

In this section, we model neighborhood attacks in social network and formulate the k-anonymity and the l-diversity problems of privacy preservation in social networks against neighborhood attacks. We also show that the two problems are NP-hard.

2.1 Preliminaries

In this paper, we model a *social network* as a simple graph $G = (V, E, L, \mathcal{L})$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, L is a set of labels, and a labeling function $\mathcal{L} : V \rightarrow L$ assigns each vertex a label. For a graph G, $V(G)$, $E(G)$, L_G, and $\mathcal{L}_G$ are, respectively, the set of vertices, the set of edges, the set of labels, and the labeling function in G. To keep our discussion simple, we assume that edges do not carry labels. However, our methods can be straightforwardly extended to remove this assumption.

The items in the label set L form a hierarchy. For example, if occupations are used as labels of vertices in a social network, L contains not only specific occupations such as *dentist, general physician, optometrist, high school teacher*, and *primary school teacher*, but also general categories like *medical doctor, teacher*, and *professional*. We assume a meta symbol $* \in L$ which is the most general category generalizing all labels. For two labels $l_1, l_2 \in L$, if l_1 is more general than l_2, we write $l_1 \prec l_2$. For example, *medical doctor* $\prec$ *optometrist*. Moreover, $l_1 \preceq l_2$ if and only if $l_1 \prec l_2$ or $l_1 = l_2$. $\preceq$ is a partial order on L.

For a graph G and a set of vertices $S \subseteq V(G)$, the *induced* subgraph of G on S is $G(S) = (S, E_S, L_G, \mathcal{L}_G)$ where $E_S = \{(u, v) | (u, v) \in E(G) \wedge u, v \in S\}$.

In a social network G, the *neighborhood* of $u \in V(G)$ is the induced subgraph of the neighbors of u, denoted by $Neighbor_G(u) = G(N_u)$ where $N_u = \{v | (u, v) \in E(G)\}$.

Given a graph $H = (V_H, E_H, L, \mathcal{L})$ and a social network $G = (V, E, L, \mathcal{L})$, an *instance* of H in G is a tuple $\left(H', f\right)$ where $H' = (V_{H'}, E_{H'}, L, \mathcal{L})$ is a subgraph in G and $f : V_H \rightarrow V_{H'}$ is a bijection function such that (1) for any $u \in V_H$, $\mathcal{L}(f(u)) \preceq \mathcal{L}(u)$, and (2) $(u, v) \in E_H$ if and only if $(f(u), f(v)) \in E_{H'}$. Literally, the first condition states that the corresponding labels in H' can be more general than those in H.

2.2 Neighbor attacks in social networks

To model privacy attacks and protection in social networks, we need to formulate three issues. First, we need to identify the privacy information under attack. Second, we need to model the background knowledge that an adversary may use to attack the privacy. Last, we need to specify the usage of the published social network data so that an anonymization method can

try to retain the utility of the data as much as possible while the privacy information is fully preserved.

In this paper, we are interested in preserving the privacy of individuals which are represented as vertices in a social network. Specifically, how a small subset of vertices are connected in a social network is considered as the privacy of those vertices.

Particularly, we address re-identification attacks. That is, we want to protect individuals from being re-identified from a published social network by adversaries. In order to attack the privacy of a target individual by analyzing the released anonymization network and re-identify the vertex, an adversary needs some background knowledge. Equipped with different background knowledge, an adversary may conduct different types of attacks against privacy. Therefore, the assumptions of adversaries' background knowledge play a critical role in both modeling privacy attacks on social networks and developing anonymization strategies to protect privacy in social network data.

In this paper, we assume that an adversary may have the background knowledge about the neighborhood of some target individuals. This assumption is realistic in many applications. Among many types of information about a target victim that an adversary may collect to attack the victim's privacy, one essential piece of information easy to be collected is the neighborhood, that is, what the neighbors of the victim are and how those neighbors are connected. For example, several online social networking sites such as `facebook` allow a "lookahead" function [20] such that a user can reach the neighbors of her/his neighbors, and so on. Thus, an attacker may easily collect the neighborhood structure of a target vertex by conducting several lookahead operations.

We assume that an adversary only knows the network topology of the neighborhood and does not have the background knowledge of vertex labels. This assumption is also realistic in some applications. For example, a large number of users do not want to put too much personal information in `facebook` due to privacy concerns. Therefore, it is impossible for the adversary to collect the textual attributes of those users in the network. Please note that the second assumption can be easily removed by extending the techniques developed in this paper straightforwardly.

Generally, we can consider the d-neighbors of the target vertex, that is, the vertices within distance d to the target vertex in the network where d is a positive integer. However, when d is large, collecting information about the d-neighbors of a target vertex may often be impractical for an adversary since the adversary may often have a limited access to a large social network. Moreover, as found in many social networks, the network diameter is often small. In other words, when $d > 1$, an adversary may have to collect information about many vertices. Furthermore, by default privacy settings, those popular online social networking sites (for example, `facebook, myspace, LinkedIn`) usually only allow a user to reach his/her immediate neighbors. A user does not have the privilege to access the information of his/her friends' friends in the network. Thus, a user cannot infer d-neighbors of a target vertex when $d > 1$. Therefore, we confine our discussion in this paper to the essential case where only the immediate neighbors, that is, vertices in $Neighbor_G(u)$, are considered.

It is interesting to notice that a few recent studies on privacy preserving publishing of social network data consider that the attackers have the background knowledge of vertex degrees [16,17,24,45], node pair similarities [46,47], and link types [4,5,8,48]. Those types of background knowledge can be regarded as special cases of the background knowledge of neighborhoods. Under those assumptions, the attackers have weaker capability to attack privacy than the adversaries assumed in this paper. Moreover, some other recent studies also consider neighborhood information as the attackers background knowledge [16,17,45,51]. However, only network structures of neighborhood information are considered, and textual

attributes of neighborhood information, such as vertex labels, are ignored. To the best of our knowledge, assuming neighborhood information of both network structures and textual attributes as the adversary background knowledge is so far the strongest. Consequently, our model can protect privacy better than those taking a weaker assumption.

The models such as k-anonymity and l-diversity make assumptions about attackers' background knowledge. Recently, the *differential privacy* model [10,32] removes such assumptions. Generally speaking, the differential privacy model ensures that the addition or removal of a single database item does not substantially affect the distributional information for data analysis. However, Xiao and Tao [41] showed that verifying differential privacy of the tabular data is NP-hard. Machanavajjhala et al. [28] found that so far no deterministic algorithm can satisfy differential privacy and some randomized algorithms can only with a very small probability return the anonymized data that are totally unrepresentative of the original tabular data. Moreover, the differential privacy model is unable to quantify exactly what sensitive information will be breached by releasing the data [9]. As outlined in Dwork and Smith [11], for tabular data, the differential privacy model uses a more general guarantee that no additional harm will happen if a record is added or deleted. However, social networks contain nodes and edges. It is an open problem to provide clear definitions of the privacy and the utility of social network data under the differential privacy model. Furthermore, most recently, Hay et al. [18] applied the differential privacy model to social network data. However, the technique in Hay et al. [18] can only provide approximated estimates of the degree distributions and it does not release a graph. In some data analysis applications, the structural properties of the network data are of great interest. Thus, releasing an anonymized graph is preferred.

Undoubtedly, the differential privacy model brings new opportunities and also new challenges for privacy preserving social network publishing. It is an interesting direction for future research. In this paper, we focus our discussion on privacy models of k-anonymity and l-diversity, and our goal is to generate an anonymized graph.

An important aspect of anonymizing social network data is how the anonymized networks are expected to be used. Different applications may have different expectations. In some situations, anonymized networks may be used to analyze the global structures. In some other situations, anonymized networks may be used to analyze the micro-structures. Clearly, different usage expectations may lead to different anonymization schemes.

In this paper, we focus on using anonymized social networks to answer aggregate network queries. An *aggregate network query* computes the aggregate on selected paths or subgraphs based on some given conditions. As an example, suppose a user is interested in the average distance from a medical doctor vertex to a nurse vertex in a network. For each doctor vertex, we can find the nearest neighbor vertex that is a nurse. Then, the aggregate network query returns the average distance between a doctor vertex to its nearest nurse neighbor.

Aggregate network queries are useful in many applications, such as customer relationship management. For example, a mobile phone company may want to promote a new phone plan to its targeted customers of medical doctors. The company may also want to know the other potential customers for this new plan, that is, who are closely connected to medical doctors in a mobile communication network. An aggregate network query on the mobile communication network may return the result that medical doctors and nurses are highly connected. Thus the company can promote their new service to its customers of nurses as well. While many types of queries on social networks are interesting, we are particularly interested in aggregate network queries in this paper since typically detailed data are needed to answer such queries accurately. Using aggregate network queries, we can examine the effectiveness of social network anonymization in a meaningful way.

There are many possible ways to anonymize a social network, such as adding and/or deleting edges/vertices. Consider a social network $G = (V, E, L, \mathcal{L})$ and the anonymization $G' = (V', E', L', \mathcal{L}')$ for publishing. We assume that in the anonymization, no fake vertices are added and no vertices in the original graphs are deleted. That is, there is a bijection function $\mathcal{A} : V \to V'$. This assumption is often desirable in applications since all actors are retained in the published social network.

Moreover, we assume that for $(u, v) \in E$, $(\mathcal{A}(u), \mathcal{A}(v)) \in E'$. That is, the connections between vertices in G are retained in G'. During the anonymization, only edges can be added and no edges can be deleted. This assumption is realistic in some application scenarios. For example, in a friendship network for customer-relationship management, we do not want to remove the connections between customers so that a possible customer referring channel may be missed. This assumption is due to the technical concern. In real social networks with vertex degree in power law distribution, the vertices with large degrees are the major subjects of anonymization operations. If edges can be deleted, the degrees of those head vertices may drop substantially. Moreover, it is challenging to define a meaningful quality measure for anonymized graphs with both edges added and deleted.

We believe that a general model of allowing both adding and deleting edges can be highly desirable. It is possible that the neighborhood information of both network structures and textual attributes is used as the attackers' background knowledge. However, under this assumption of the attackers' background knowledge, privacy preservation in social network publishing so far remains an open problem in both the model design and the anonymization algorithm development.

2.3 k-Anonymity in social networks

An adversary may attack the privacy using the neighborhoods. For a social network G, suppose an adversary knows $Neighbor_G(u)$ for a vertex $u \in V(G)$. If $Neighbor_G(u)$ has k instances in G' where G' is an anonymization of G, then u can be re-identified in G' with confidence $\frac{1}{k}$.

Similar to the philosophy in the k-anonymity model [35], to protect the privacy of vertices sufficiently, we want to keep the re-identification confidence lower than a threshold. Let k be a positive integer. For a vertex $u \in V(G)$, u is *k-anonymous* in anonymization G' if there are at least $(k-1)$ other vertices $v_1, \ldots, v_{k-1} \in V(G)$ such that $Neighbor_{G'}(\mathcal{A}(u))$, $Neighbor_{G'}(\mathcal{A}(v_1)), \ldots, Neighbor_{G'}(\mathcal{A}(v_{k-1}))$ are isomorphic. G' is *k-anonymous* if every vertex in G' is k-anonymous.

Analogous to the correctness of k-anonymity model [35] on relational data, we have the following claim.

Property 1 (K-anonymity) Let G be a social network and G' an anonymization of G. If G' is k-anonymous, then with the neighborhood background knowledge, any vertex in G cannot be re-identified in G' with confidence larger than $\frac{1}{k}$.

An adversary knowing the neighborhood of a target vertex is a strong assumption. Provided privacy is preserved under this assumption, privacy is also preserved when an adversary knows only part of the neighborhood (that is, only some neighbors and some connections among neighbors) of a target vertex.

Given a social network G, the *k-anonymity problem* studied in this paper is to compute an anonymization G' such that (1) G' is k-anonymous; (2) each vertex in G is anonymized to a vertex in G' and G' does not contain any fake vertex; (3) every edge in G is retained in G'; and (4) the number of edges to be added is minimized. Heuristically, when the number

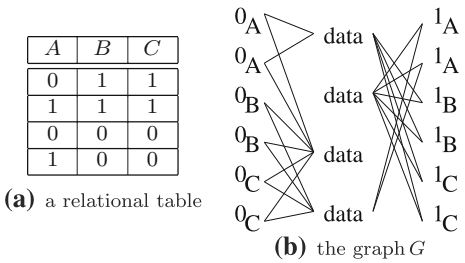


Fig. 2 Transforming a relational table to a graph

of edges added is as small as possible, G' can be used to answer aggregate network queries accurately.

The k-anonymity problem in social networks is challenging. We show that a simplified version of the k-anonymity problem in social networks is NP-hard where all vertices in G carry the same label, or equivalently, G is not labeled.

Theorem 1 (Complexity of k-anonymity in Social Networks) *The following k-anonymity problem in social network is NP-hard.*
Instance: *a social network $G = (V, E, L, \mathcal{L})$, positive integers k and n.*
Question: *is there an anonymized social network $G' = \left(V, E', L, \mathcal{L}\right)$ such that $E \subset E'$, $\left|E' - E\right| \leq n$, and G' is k-anonymous?*

Proof The *k-anonymity problem in relational data* [29] as follows is proved NP-hard by an induction from the *k-Dimensional Perfect Matching* problem [19].
Instance: a table $T = (A_1, \ldots, A_l)$ where each attribute is in domain $\{0, 1\}$, positive integers k and n.
Question: an entry is an attribute value in a tuple. Can we suppress at most n entries in T such that after the suppression the table is k-anonymous (that is, each tuple is identical to $(k-1)$ other tuples in the table after suppression)?

Here, we prove the theorem by reducing the k-anonymity problem in relational data to the k-anonymity problem in social networks.

Consider integer k and a table $T = (A_1, \ldots, A_l)$ where each attribute is in domain $\{0, 1\}$. We transform T into a graph G as follows.

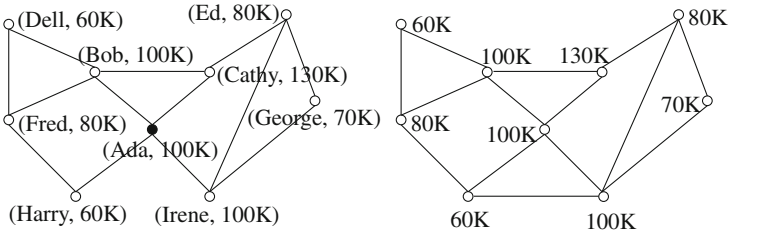
- For each attribute A_i $(1 \leq i \leq l)$, we create k vertices in G with label 0_i and k vertices in G with label 1_i.
- For each tuple t in T, we create a vertex of label `data` in G. That is, all vertices for tuples carry the same label. Moreover, for each attribute A_i, if $t.A_1 = 0$, we add k edges between the vertex of t and the k vertices of label 0_i. Similarly, if $t.A_1 = 1$, we add k edges between the vertex of t and the k vertices of label 1_i.

For example, consider table T in Fig. 2a and $k = 2$. Figure 2b shows the graph transformed from the table.

Apparently, the above transformation is of complexity $O(2l + kl|T|)$, where $|T|$ is the number of tuples in T.

A graph G obtained as such has the following property: vertices 0_i's and 1_i's $(1 \leq i \leq l)$ are k-anonymous. In the rest of this proof, we consider only anonymizations of G which keep the class labels of the vertices unchanged.

Let G' be a k-anonymization of G. We have two important observations. First, if G' contains any edge between vertices of label `data`, removing all of those edges does not affect



(a) a social network **(b)** a 2−anonymous network still leaks privacy

Fig. 3 Neighborhood attacks in a social network with sensitive labels

the k-anonymity of the graph. Second, if a vertex u of label `data` is connected with a vertex v of label 0_i (1_i), u must be connected with all other $k-1$ vertices of label $0_i\,(1_i)$ as well.

If T has a k-anonymization T' where at most n entries are suppressed, then there exists a k-anonymization G' of G where $\left|E\left(G'\right) - E(G)\right| \leq kn$—when an entry is suppressed in T', we add k edges in G' to link the corresponding vertex with label `data` to the k 0_i or 1_i vertices.

On the other hand, let G' be a k-anonymization of G and $\left|E\left(G'\right) - E(G)\right| \leq kn$. Based on the first property, we remove from G' all edges between vertices of label `data` and obtain G''. G'' is also a k-anonymization of G and $\left|E\left(G''\right) - E(G)\right| \leq \left|E\left(G'\right) - E(G)\right| \leq kn$. Based on the second property, we know that $\left|E\left(G''\right) - E(G)\right|$ must be a multiple of k.

We construct a table $T' = (A_1, \ldots, A_l)$ from graph G'' as follows. For each vertex u in G'' of label `data`, we create a tuple t in T'. For each attribute A_i $(1 \leq i \leq l)$, if u is connected with vertices of labels both 0_i and 1_i, the value of $t.A_i$ is suppressed. Otherwise, $t.A_1 = 0(1)$ if u is connected with vertices of label $0_i\,(1_i)$ only. Clearly, T' is a k-anonymization of T where at most n entries are suppressed.

We show that the k-anonymity problem in relational data can be reduced to the k-anonymity problem in social networks. □

In Sect. 4, we will develop a practical solution to the k-anonymity problem, which can obtain k-anonymous social networks with low information loss in practice.

2.4 l-Diversity in social networks

As shown in Machanavajjhala et al. [27], a k-anonymized relational table may not preserve privacy sufficiently if it lacks diversity in sensitive attributes. Similarly, k-anonymized social network data still may leak privacy. If an adversary can link a victim to a group of vertices anonymized together all associated with a sensitive attribute value, then the adversary still can link the victim to the sensitive attribute value.

As a concrete example, consider the social network in Fig. 3a. Each vertex in the social network carries two labels: the name and a sensitive attribute value `Salary`. Figure 3b is a 2-anonymous network of Fig. 3a. Does Fig. 3b preserve the privacy on the sensitive salary information sufficiently?

If an adversary is equipped with the background knowledge of the 1-neighborhood of Ada, due to the 2-anonymity, the adversary cannot identify the vertex of Ada in Fig. 3b. However, since Ada, Bob and Irene have the isomorphic 1-neighborhood in Fig. 3b, and no one else has the same 1-neighborhood, the adversary is sure that Ada must be one of the three vertices. Importantly, since Ada, Bob and Irene all have salary $100K$, the adversary can accurately determine the salary of Ada.

The above example clearly demonstrates that a k-anonymized social network may still disclose sensitive information due to the lack of diversity. We model the attack illustrated in the example as follows.

In a social network G, a set S of m vertices $\{v_1, v_2, \ldots, v_m\} \subseteq V(G)$ is said to form an *equivalence group* if their 1-neighborhood structures $Neighbor_G(v_1)$, $Neighbor_G(v_2)$, $\ldots$, $Neighbor_G(v_m)$ are isomorphic.

Following the philosophy of the l-diversity principle [27] in relational data, in order to protect the privacy of the vertices sufficiently in an equivalence group, we have to make sure that the distribution of the sensitive values in each equivalence group are sufficiently diverse.

Technically, let G be a social network and G' be an anonymization of G. G' is said to be l-diverse if in every equivalence group of vertices, at most $\frac{1}{l}$ of the vertices are associated with the most frequent sensitive label.[1]

As a result, an adversary with the background knowledge of 1-neighborhood structure only can infer the sensitive label for a target victim with the probability not large than $\frac{1}{l}$. The larger the value of l, the better privacy is protected.

For simplicity, suppose each vertex in a social network carries only one sensitive label.[2] Obviously, an l-diverse social network is also l-anonymous. The complexity of the l-diversity problem in social networks can be shown to be NP-hard.

Theorem 2 (Complexity of l-diversity in Social Networks) *The following l-diversity problem in social networks is NP-hard.*
Instance: *a social network $G = (V, E, L, \mathcal{L})$, positive integers l and n.*
Question: *is there an anonymized social network $G' = \left(V, E', L, \mathcal{L}\right)$ such that $E \subset E'$, $\left|E' - E\right| \leq n$, and G' is l-diverse?*

Proof We reduce the problem of k-anonymity in social networks to k-diversity in social networks. For a given social network G that we want to compute its k-anonymization by adding at most n edges, we construct a new social network G' by assigning each vertex in G a unique sensitive label. Clearly, there is a k-anonymization of G_1 such that $|E(G_1) - E(G)| \leq n$ if and only if there is a k-diverse anonymization G_2 of G' such that $\left|E(G_2) - E\left(G'\right)\right| \leq n$. □

3 Related work

Privacy becomes a more and more serious concern in many applications. The development of privacy preserving data processing techniques has become a fruitful direction for database and data mining research.

One of the privacy concerned problems is publishing microdata for public use [33], which has been extensively studied recently. The major goal of privacy preserving research is how to hide sensitive knowledge [15]. A large category of privacy attacks is to re-identify individuals by joining the published table with some external tables modeling the background knowledge of users. To battle this type of attacks, the mechanism of k-anonymity was proposed in Samarati and Sweeney [34], Sweeney [35]. Specifically, a data set is said to be k-anonymous ($k \geq 1$) if, on the quasi-identifier attributes (that is, the minimal set of attributes in the table

[1] The original definition of l-diversity in Machanavajjhala et al. [27] is more general. For simplicity, we adopt the frequency-based l-diversity, as most of the previous studies [23,38,40] did.

[2] This assumption can be easily removed by expanding the cardinality of the sensitive label set, that is, considering a combination of multiple sensitive labels as a new sensitive label.

that can be joined with external information to re-identify individual records), each record is indistinguishable from at least $k-1$ other records within the same data set. The larger the value of k, the better the privacy is protected.

Machanavajjhala et al. [27] showed that a k-anonymized dataset has some subtle but severe privacy problems due to the lack of diversity in the sensitive attributes. In particular, they showed that, the degree of privacy protection does not really depend on the size of the quasi-identifier attribute set. Instead, it is determined by the number of distinct sensitive values associated with each quasi-identifier attribute set. The observation leads to the notion of l-diversity [27]. Xiao and Tao [39] proved that l-diversity always guarantees stronger privacy preservation than k-anonymity.

While the privacy models ensure that the anonymized data can protect privacy, the utility of anonymized data also plays an important role. Many previous studies have been conducted on anonymized data for specific data mining tasks, such as Luo et al. [26], Qiu et al. [31].

In this paper, we focus on k-anonymity and l-diversity since they are the most essential and most applicable privacy models. Particularly, k-anonymity can be used even when sensitive attributes are not defined. A few governmental privacy regulations including HIPAA and European Union Data Directive adopted k-anonymity.

Beyond microdata, some other data sources such as social network data also have privacy concerns when they are published for public use. In online social networking sites, privacy breaching may happen in many scenarios. Muhlestein and Lim [30] considered the privacy issue when people in the online community try to share common interests. A thorough survey on the recent work of anonymization techniques for privacy preserving publishing of social network data is provided in Liu et al. [25], Zhou et al. [50]. In the following, we briefly review some representative studies and point out the differences between those studies and ours.

Typically, social network data can be represented as a graph, in which vertices correspond to people or other social entities, and edges correspond to social links between them [37]. As a first step to hide information about social entities while preserving the global network properties, the released social network data have to go through the anonymization procedure which replaces social entity names with meaningless unique identifiers [3]. Although this kind of anonymization can exactly preserve the unannotated structure of the social network, it may still leak a lot of privacy information of individuals.

Attacks in social network data can be regarded as one kind of link mining [14]. Specifically, as a pioneer work about privacy in social network data, Backstrom et al. [3] described a family of attacks based on random graph theory. For example, an attacker may plant some well-constructed sub-structures associated with the target entities in advance. Once the social network data are collected and published, the attacker can first try to identify the planted structures and thus peek the linkage between the target vertices. However, there is no practical solution proposed in Backstrom et al. [3] to counter those attacks.

The attacks proposed in Backstrom et al. [3] are different from the neighborhood attacks addressed in this paper. The attacks in Backstrom et al. [3] need to plant a set of deliberative structures before the social network data are anonymized, which is a task hard to achieve in some situations. As shown before, even without planting deliberative structures, the released social network data are still in danger, as neighborhood attacks are still possible.

Wang et al. [36] adopted description logic as the underlying knowledge representation formalism and proposed some metrics of anonymity for assessing the risk of breaching confidentiality by disclosing social network data. However, they did not give any anonymization algorithms for social network data.

Recently, Zheleva and Getoor [48] proposed a model different from ours. They focused on social networks where nodes are not labeled but edges are labeled. Some types of edges are sensitive and should be hidden. They provided the edge anonymization methods based on edge clustering and removal to prevent link re-identification.

Very recently (after the preliminary version of this paper [49] was published), Liu and Terzi [24] studied the graph k-degree anonymous problem. Specifically, they address the possible re-identification attacks against individuals by an adversary using the prior knowledge of the degree of a victim vertex. A graph is said to be k-degree anonymous if for every node v, there exist at least $k-1$ other nodes in the graph with the same degree as v. Generally, their privacy requirement is weaker than ours. They only consider the node degrees, but ignore the link structures among those 1-neighbor nodes. The 1-neighborhood attacks considered in this paper are more general.

For example, the network in Fig. 1 is 2-degree anonymous. However, as shown in Sect. 1.1, it is not 2-anonymous. Ada in the network still can be re-identified by her 1-neighborhood.

Liu and Terzi [24] proposed to modify the graph by adding and deleting edges from the graph to achieve k-degree anonymity. As pointed out in Liu and Terzi [24], due to the difference in problem formulation, the approaches in Liu and Terzi [24] cannot be used to tackle the problem studied in this paper.

Simultaneous to our study, Hay et al. [16,17] presented a framework for assessing the privacy risk of sharing anonymized network data. They modeled the adversaries' background knowledge as vertex requirement structural queries and subgraph knowledge structural queries, and proposed a privacy requirement k-candidate anonymity that is similar to k-anonymity in tabular data. They developed a random graph perturbation method by randomly deleting or adding edges to anonymize a social network. Their model assumes that the nodes and the edges in a social network are not labeled.

Zou et al. [51] proposed a k-automorphism framework to anonymize social networks. Given an original network, the algorithm in Zou et al. [51] converts the network into a k-automorphic network, which is then published. The networks considered in Zou et al. [51] are unlabeled. However, we consider the case that each vertex in the network carries sensitive labels and nonsensitive labels.

In some applications, entities and their relationships can be modeled as a bipartite graph, such as customers and medical products used. The edges in such a bipartite graph may be considered as privacy. Cormode et al. [8] focused on the problem of anonymizing bipartite graphs. The method proposed in Cormode et al. [8] cannot be extended to social networks. Later on, Bhagat et al. [4] studied a much richer class of graphs called interaction graphs that can handle many different types of entities and edge types. Bhagat et al. [4] represented the interaction graphs as bipartite graphs over the sets of entities and interactions. However, we consider more general social networks in which edges do not carry type information. The grouping method proposed in Bhagat et al. [4] cannot be used to solve our problem.

4 A k-anonymity method

In this section, we introduce a practical method to anonymize a social network to satisfy the k-anonymity requirement. The method is in two steps.

First, we extract the neighborhoods of all vertices in the network. To facilitate the comparisons among neighborhoods of different vertices including the isomorphism tests, which will be conducted frequently in anonymization, we propose a simple yet effective *neighborhood component coding technique* to represent the neighborhoods in a concise way.

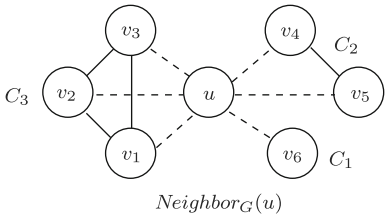


Fig. 4 Neighborhood and neighborhood components (the dashed edges are just for illustration and are not in the neighborhood subgraph)

In the second step, we greedily organize vertices into groups and anonymize the neighborhoods of vertices in the same group. Due to the well-recognized power law distribution of the degrees of vertices in large social networks, we start with those vertices of high degrees.

4.1 Neighborhood extraction and coding

In order to meet the k-anonymity requirement, we need to put vertices into groups and anonymize the neighborhoods of vertices in a group. Ideally, vertices having similar neighborhoods should be grouped together. As the first step, we extract neighborhoods of vertices and represent them in a concise way.

Extracting the neighborhood of a vertex is straightforward. The challenge is how we can represent the neighborhood information to facilitate the later operations in anonymization. Since we need to anonymize all neighborhoods of the vertices in one group to the same, isomorphism tests are frequently conducted.

The general graph isomorphism problem that determines whether two graphs are isomorphic is NP (which is neither known to be solvable in polynomial time nor NP-complete) [13]. Here, we propose a coding technique for neighborhood subgraphs so that whether two neighborhoods are isomorphic can be determined by the corresponding coding.

In a social network G, a subgraph C of G is a *neighborhood component* of $u \in V(G)$ if C is a maximal connected subgraph in $Neighbor_G(u)$.

Example 1 (Neighborhood component) Figure 4 shows $Neighbor_G(u)$, the neighborhood of a vertex u. $Neighbor_G(u)$ contains three neighborhood components, C_1, C_2, and C_3 as shown in the figure.

Clearly, the neighborhood of a vertex can be divided into neighborhood components. To code the whole neighborhood, we need to first code each component.

The *depth-first search tree* (*DFS-tree* for short) [7] is popularly used for navigating connected graphs. Thus, it is natural to encode the edges and vertices in a graph based on its DFS-tree. All the vertices in G can be encoded in the pre-order of T. However, the DFS-tree is generally not unique for a graph. That is, there can be multiple DFS-trees corresponding to a given graph.

For example, Fig. 5b and c show two DFS-trees of the graph G in Fig. 5a. The thick edges in Fig. 5b and c are those in the DFS-trees, and are called the *forward edges*, while the thin edges are those not in the DFS-trees, and are called the *backward edges*. The vertices in the graph are encoded v_0 to v_3 according to the pre-order of the corresponding DFS-trees.

To solve the uniqueness problem, a *minimum DFS code* notation is proposed in Yan and Han [43]. For any connected graph G, let T be a DFS-tree of G. Then, an edge is always listed as (v_i, v_j) such that $i < j$. A linear order $\prec$ on the edges in G can be defined as follows. Given edges $e = (v_i, v_j)$ and $e' = (v_{i'}, v_{j'})$. $e \prec e'$ if (1) when both e and e' are *forward edges* (that is, in DFS-tree T), $j < j'$ or $(i > i' \wedge j = j')$; (2) when both e and e'

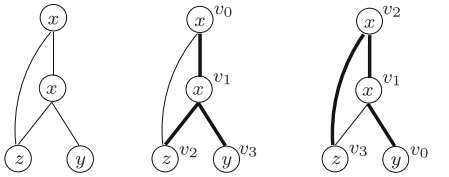


Fig. 5 DFS codes, starting from different vertices

(a) Graph G **(b)** DFS-tree T_1 **(c)** DFS-tree T_2

are *backward edges* (that is, edges not in DFS-tree T), $i < i'$ or $\left(i = i' \wedge j < j'\right)$; (3) when e is a forward edge and e' is a backward edge, $j \leq i'$; or (4) when e is a backward edge and e' is a forward edge, $i < j'$.

For a graph G and a DFS-tree T, a list of all edges in $E(G)$ in order $\prec$ is called the *DFS code* of G with respect to T, denoted by $code(G, T)$. For example, the DFS code with respect to the DFS-tree T_1 in Fig. 5b is $code(G, T_1) = \langle (v_0, v_1, x, x)\text{-}(v_1, v_2, x, z)\text{-}(v_2, v_0, z, x)\text{-}(v_1, v_3, x, y)\rangle$, where an edge $\left(v_i, v_j\right)$ is written as $\left(v_i, v_j, \mathcal{L}(v_i), \mathcal{L}\left(v_j\right)\right)$, that is, the labels are included. Similarly, the DFS code with respect to the DFS-tree T_2 in Fig. 5c is $code(G, T_2) = \langle (v_0, v_1, y, x)\text{-}(v_1, v_2, x, x)\text{-}(v_2, v_3, x, z)\text{-}(v_3, v_1, z, x)\rangle$.

Suppose there is a linear order over the label set L. Then, for DFS-trees T_1 and T_2 on the same graph G, their DFS codes can be compared lexically according to the vertex pairs as labels of edges. For example, we have $code(G, T_1) < code(G, T_2)$ in Fig. 5b and c.

The lexically *minimum DFS code* is selected as the representation of the graph, denoted by $DFS(G)$. In our example in Fig. 5, $DFS(G) = code(G, T_1)$.

Minimum DFS code has a nice property [43]: two graphs G and G' are isomorphic if and only if $DFS(G) = DFS\left(G'\right)$. Using minimum DFS code, we can code every component of the neighborhood of a vertex. Now, the problem becomes how we can combine the minimum DFS codes of all components in a neighborhood into one code.

For two neighborhood components C_i and C_j in $Neighbor_G(u)$, we define $C_i \prec C_j$ if (1) $|V(C_i)| < \left|V\left(C_j\right)\right|$; or (2) $|V(C_i)| = \left|V\left(C_j\right)\right|$ and $|E(C_i) < \left|E\left(C_j\right)\right|$; or (3) $|V(C_i)| = \left|V\left(C_j\right)\right|, |E(C_i) = \left|E\left(C_j\right)\right|$, and $DFS(C_i)$ is smaller than $DFS\left(C_j\right)$.

Based on the neighborhood component order, we can assign a canonical label for each neighborhood. In a social network G, for vertex $u \in V(G)$, the *neighborhood component code* of $Neighbor_G(u)$ is a vector $NCC(u) = (DFS(C_1), \ldots, DFS(C_m))$ where $C_1, \ldots, C_m$ are the neighborhood components of $Neighbor_G(u)$, that is, $Neighbor_G(u) = \cup_{i=1}^{m} C_i$, $C_i \preceq C_j$ for $1 \leq i < j \leq m$.

Example 2 (Neighborhood component code) In Fig. 4, the neighborhood component code of $Neighbor_G(u)$ is $NCC(u) = (DFS(C_1), DFS(C_2), DFS(C_3))$.

Using neighborhood component code, we can easily identify isomorphic neighborhoods.

Theorem 3 (Neighborhood component code) *For two vertices $u, v \in V(G)$ where G is a social network, $Neighbor_G(u)$ and $Neighbor_G(v)$ are isomorphic if and only if $NCC(u) = NCC(v)$.*

Proof Yan and Han [43] showed that two graphs G and G' are isomorphic if and only if their minimal DFS codes are identical. In neighborhood component codes, each component is represented using its minimum DFS code. Consider vertices u and v. The isomorphism test of the two neighborhoods can be replaced by examining whether the corresponding neighborhood component codes are identical. □

As pointed out in Yan and Han [43], a graph may have multiple DFS codes. Two graphs are isomorphic if and only if their minimum DFS codes are identical. As the general graph isomorphism problem is NP, that is, it is neither known to be solvable in polynomial time nor NP-Complete, computing the minimum DFS code for a graph in general carries the same complexity nature as the general graph isomorphism problem. The time complexity of the general graph isomorphic problem is not improved by minimum DFS code.

However, using neighborhood component code to label and index neighborhoods has some advantages. First, it is easy to test whether a group of neighborhoods are isomorphic once their neighborhood component codes are pre-computed. Without any coding techniques, we need to conduct graph isomorphic tests for many pairs of neighborhood graphs. With the help of neighborhood component codes, we can easily identify which set of neighborhoods are isomorphic by examining whether their corresponding indices are identical. Second, since each neighborhood is decomposed into several neighborhood components, it is easy to calculate the structure similarity between two neighborhoods by calculating the similarity between different components. Third, we can find similar components between two neighborhoods. Due to the specific characteristics of minimum DFS code, the problem of finding isomorphic connected subgraphs from two neighborhoods is equivalent to finding equivalent minimum DFS codes from the corresponding neighborhood component codes. As a result, anonymizing similar components can lead to low information loss and high similarity between the anonymization and the original social network.

In summary, in the first step, we extract the neighborhood for each vertex, and compute its neighborhood component code.

4.2 Social network anonymization

One major challenge in anonymizing a social network is that changing labels of vertices and adding edges may affect the neighborhoods of some other vertices as well as the properties of the network. It has been well recognized that the following two properties often hold in practical social networks. The properties help us in designing anonymization methods.

Property 2 (vertex degree in power law distribution) The degrees of vertices in a large social network often follow the power law distribution [12]. Such degree distributions have been identified in various social networks including Internet, biological networks, and co-authorship networks.

Property 3 (the "small-world phenomenon" [37]) It is also popularly known as "six degrees of separation", which states that large social networks in practice often have surprisingly small average diameters.

Our social network anonymization method processes vertices in the degree descending order and utilizes the above two properties of large social networks in practice.

The k-anonymity requires that each vertex $u \in V(G)$ is grouped with at least $(k-1)$ other vertices such that their anonymized neighborhoods are isomorphic. For a group S of vertices having the isomorphic anonymized neighborhoods, all vertices in S have the same degree. Since the degrees of vertices in a large social network follow a power law distribution, only a small number of vertices have a high degree. Processing those vertices of high degrees first can keep the information loss about those vertices low. There are often many vertices of a low degree. It is relatively easy to anonymize those low degree vertices and retain high quality. Moreover, as will be shown soon, low degree vertices can be used to anonymize those high degree vertices and do not affect the diameters of the network too much.

4.2.1 Anonymization quality measure

In our social network anonymization model, there are two ways to anonymize the neighborhoods of vertices: *generalizing vertex labels* and *adding edges*. Each of the two methods leads to some information loss.

The information loss due to generalization of vertex labels can be measured by the normalized certainty penalty [42]. Consider a vertex u of label l_1, where l_1 is at the leaf level of the label hierarchy, that is, l_1 does not have any descendant. Suppose l_1 is generalized to l_2 for u where $l_2 \prec l_1$. Let $size(l_2)$ be the number of descendants of l_2 that are leafs in the label hierarchy, and $size(*)$ be the total number of leafs in the label hierarchy. Then, the *normalized certainty penalty* of l_2 is $NCP(l_2) = \frac{size(l_2)}{size(*)}$.

The information loss due to adding edges can be measured by the total number of edges added and the number of vertices that are not in the neighborhood of the target vertex and are linked to the anonymized neighborhood for the purpose of anonymization.

Consider two vertices $u_1, u_2 \in V(G)$ where G is a social network. Suppose $Neighbor_G(u_1)$ and $Neighbor_G(u_2)$ are generalized to $Neighbor_{G'}(\mathcal{A}(u_1))$ and $Neighbor_{G'}(\mathcal{A}(u_2))$ such that $Neighbor_{G'}(\mathcal{A}(u_1))$ and $Neighbor_{G'}(\mathcal{A}(u_2))$ are isomorphic. Let $H = Neighbor_G(u_1) \cup Neighbor_G(u_2)$ and $H' = Neighbor_{G'}(\mathcal{A}(u_1)) \cup Neighbor_{G'}(\mathcal{A}(u_2))$. The *anonymization cost* is

$$\begin{aligned} Cost(u, v) = \alpha \cdot & \sum_{v' \in H'} NCP\left(v'\right) \\ & + \beta \cdot \left|\left\{(v_1, v_2) | (v_1, v_2) \notin E(H), (\mathcal{A}(v_1), \mathcal{A}(v_2)) \in E\left(H'\right)\right\}\right| \\ & + \gamma \cdot \left(\left|V\left(H'\right)\right| - |V(H)|\right) \end{aligned}$$

where α, β and γ are the weights specified by users. Literally, the cost consists of three parts. The first part is the normalized certainty penalty measuring the information loss of generalizing labels of vertices. The second part measures the information loss due to adding edges. The last part counts the number of vertices that are linked to the anonymized neighborhoods to achieve k-anonymity.

The anonymization cost of two vertices u and v measures the similarity between $Neighbor_G(u)$ and $Neighbor_G(v)$. The smaller the anonymization cost, the more similar the two neighborhoods.

4.2.2 Anonymizing two neighborhoods

Now, let us consider a greedy method to anonymize two neighborhoods $Neighbor_G(u)$ and $Neighbor_G(v)$.

We first find all perfect matches of neighborhood components in $Neighbor_G(u)$ and $Neighbor_G(v)$. Two components perfectly match each other if they have the same minimum DFS code. Those perfect matches are marked as "matched" and pass over for further consideration.

For example, consider two vertices u and v whose neighborhoods are shown in Fig. 6. Each vertex is shown in the form of $(id, label)$. The neighborhood component $C_2(u) \in Neighbor_G(u)$ perfectly matches $C_3(v) \in Neighbor_G(v)$.

For those unmatched components, the anonymization algorithm tries to pair similar components and anonymize them. The similarity between two components is based on the anonymization cost. To calculate the similarity between two components, we try to match similar

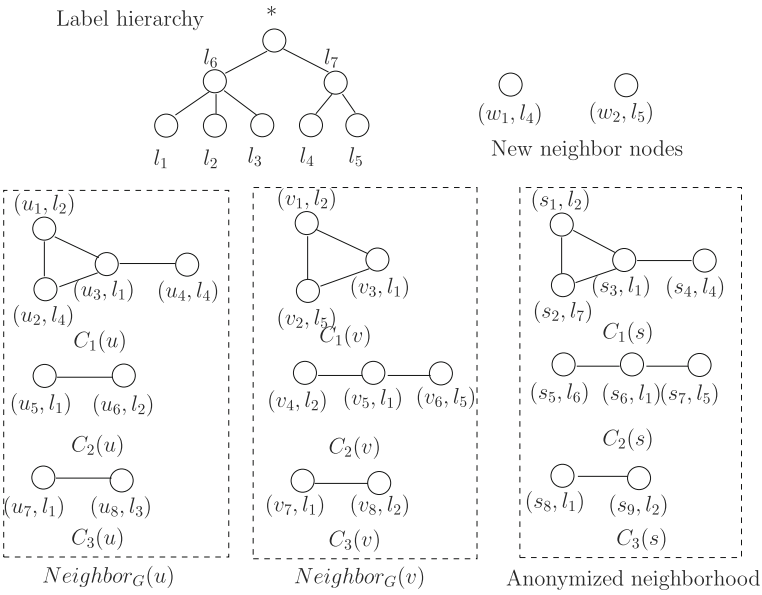


Fig. 6 Anonymizing two neighborhoods

vertices in the two components as much as possible. This is a traditional substructure similarity search problem, which has been proved NP-hard [44]. Instead of computing the optimal matching, we conduct a greedy match.

To start with, we first try to find two vertices with the same degree and the same label in the two components to be matched. If there are multiple matching vertex pairs, the pair with the highest vertex degree is chosen. If there is no such a pair of matching vertices, we relax the matching requirement (vertex degree and label), calculate the difference of degrees and the normalized certainty penalty of generalizing the labels in the label hierarchy, and choose the one with the minimum anonymization cost. Then, we conduct a breadth-first search to match vertices one by one, until all possible vertex matchings are found. The anonymization cost is calculated according to the matching, and is used to measure the similarity of the two components.

Consider components $C_1(u)$ and $C_1(v)$ in Fig. 6. Vertices u_1 and v_1 match. We start from these two vertices and conduct a breadth-first search. Vertex v_2 partially matches vertex u_2. Vertex v_3 partially matches vertex u_3. The vertex matching stops since all possible vertex matchings are found. However, vertex u_4 does not find any vertex matching in $C_1(v)$. Thus, we have to find a vertex $w_1 \in V(G)$ that is neither in $C_1(v)$ nor in $C_1(u)$, and add it into $C_1(v)$, so that $C_1(u)$ and $C_1(v)$ can be anonymized to the same.

When a vertex has to be introduced into the neighborhood for the sake of anonymization, the following rules are used: we first consider those vertices in $V(G)$ that are unanonymized. The vertex with smallest degree has the highest priority. If there are more than one candidate with the same smallest degree, we choose the one having the closest label in terms of normalized certainty penalty. If we cannot find any other vertex that is unanonymized, we select one anonymized vertex w with the smallest degree and satisfying the label requirement, and mark w and its $(k-1)$ other vertices anonymized in the same group as "unanonymized".

In our example, suppose we can find an unanonymized vertex (w_1, l_4) to be added to $C_1(u)$, the anonymization cost of $C_1(u)$ and $C_1(v)$ is $\alpha\cdot$

$$\sum_{v' \in V(C_1(u)) \cup V(C_1(v))} NCP(\mathcal{L}(\mathcal{A}(v')) + \beta \cdot 1 + \gamma \cdot 1 = \alpha \cdot \frac{4}{5} + \beta + \gamma.$$

Based on the component similarity, we can pair similar components. We start with the component with the largest number of vertices. This component is paired with the most similar component in the other neighborhood. The two paired components are anonymized to the same, marked "matched", and removed from consideration. The matching continues until all components in one neighborhood are marked "matched".

If there are some components left in the other neighborhood say $Neighbor_G(u)$, we use some other vertices in $V(G)$ that are not in $Neighbor_G(u)$ to construct a component and add it to $Neighbor_G(u)$ to construct the matching and anonymization. The vertices are selected using the same criteria as selecting vertices to match two components.

We anonymize each pair of matched neighborhood components to the same. The two neighborhoods then are anonymized. For example, in Fig. 6, the algorithm matches components $C_1(u)$ and $C_1(v)$, and $C_2(v)$ and $C_3(u)$ in turn. As a result, two vertices w_1 and w_2 from $V(G)$ have to be added into components $C_1(v)$ and $C_3(u)$, respectively.

Once two neighborhoods are anonymized, the neighborhoods of some vertices may be changed. We need to update the neighborhood component codes for those vertices accordingly. To reduce the frequency of updates and to reduce the cost, we use the following heuristic. We try to link a vertex to some vertices of the smallest degrees. This heuristic helps to reduce the number of calculations of neighborhood component codes for updates, since those linked vertices are not likely to have many neighbors.

4.2.3 Anonymizing a social network

We propose a greedy method to anonymize a social network as shown in Fig. 1.

First, we mark all vertices in the network as "unanonymized". We maintain a list *VertexList* of "unanonymized" vertices in the neighborhood size descending order: for vertices $u, v \in V(G)$, if $|V(Neighbor_G(u))| < |V(Neighbor_G(v))|$, or $|V(Neighbor_G(u))| = |V(Neighbor_G(v))|$ and $|E(Neighbor_G(u))| < |E(Neighbor_G(v))|$, then v precedes u in the list. If their neighborhoods have the same numbers of vertices and edges, they can be ordered arbitrarily.

Iteratively, we pick the first vertex *SeedVertex* in the list *VertexList*. The anonymization cost of *SeedVertex* and any other vertices in *VertexList* is calculated using the anonymization method for two vertices discussed in Sect. 4.2.2. If the number of unanonymized vertices in *VertexList* is at least $2k-1$, we select a set *CandidateSet* of the top $k-1$ vertices in *VertexList* with the smallest anonymization cost. We can easily give a lower bound of the anonymization cost based on the number of vertices and the number of edges in two neighborhoods. Since all vertices in *VertexList* have a neighborhood size smaller than or equal to that of *SeedVertex*, we scan *VertexList* in the neighborhood size descending order, and stop once the lower bound of the anonymization cost exceeds the cost of the current $(k-1)$-th most similar vertex.

The *SeedVertex* and the vertices in $CandidateSet = \{u_1, \ldots, u_m\}$ are anonymized in turn using the anonymization method for two vertices discussed in Sect. 4.2.2. The anonymization of *SeedVertex* and u_1 is straightforward. After these two vertices are anonymized, their neighborhoods are identical. When we anonymize them with respect to u_2, any change (for example, adding an edge or a neighbor node) to the neighborhood of *SeedVertex* will be

Algorithm 1 Anonymizing a social network to achieve k-anonymity.

Input: a social network $G = (V, E)$, the anonymization requirement parameter k, the cost function parameters α, β and γ;
Output: an anonymized graph G';
Initialization: initialize $G' = G$, and mark $v_i \in V(G)$ as "unanonymized";
1: sort $v_i \in V(G)$ as *VertexList* in neighborhood size descending order;
2: **while** *VertexList* $\neq \emptyset$ **do**
3: let *SeedVertex* = *VertexList.head*() and remove it from *VertexList*;
4: **for** each $v_i \in$ *VertexList* **do**
5: calculate *Cost*(*SeedVertex*, v_i) using the anonymization method for two vertices;
6: **end for**
7: **if** *VertexList.size*() $\geq 2k - 1$ **then**
8: let *CandidateSet* contain the top $k - 1$ vertices with the smallest *Cost*;
9: **else**
10: let *CandidateSet* contain the remaining unanonymized vertices;
11: **end if**
12: suppose *CandidateSet*= $\{u_1, \ldots, u_m\}$, anonymize *Neighbor*(*SeedVertex*) and *Neighbor*(u_1) as discussed in Section 4.2.2;
13: **for** $j = 2$ to m **do**
14: anonymize *Neighbor*(u_j) and {*Neighbor*(*SeedVertex*), *Neighbor*(u_1), ..., *Neighbor*(u_{j-1})} as discussed in Section 4.2.2, mark them as "anonymized";
15: update *VertexList*;
16: **end for**
17: **end while**

applied to u_1 as well, so that the neighborhoods of *SeedVertex*, u_1 and u_2 are anonymized to the same. The process continues until the neighborhoods of *SeedVertex* and $u_1, \ldots, u_m$ are anonymized.

During the anonymization of a group of vertices, some changes may occur to some other vertices v that have been marked as "anonymized" in another group (for example, adding edges between an anonymized vertex and a vertex being anonymized based on vertex matching). In order to maintain the k-anonymity for those vertices, we apply the same changes to every other $k - 1$ vertices having the isomorphic neighborhoods as v. Once those k vertices are changed, they are marked as "unanonymized" and inserted into the *VertexList* again.

When the number of unanonymized vertices in *VertexList* is less than $2k$, to satisfy the k-anonymity, the remaining vertices in *VertexList* have to be considered together in anonymization. They are added to *CandidateSet* in a batch.

The social network anonymization algorithm continues until all the vertices in the graph are marked as "anonymized".

Theorem 4 (Termination) *The algorithm in Fig. 1 terminates for a finite social network of at least k vertices.*

Proof Clearly, a clique of n vertices where each vertex is labeled $*$ is k-anonymous provided $n \geq k$. In each iteration such that *VertexList* is not shortened, the algorithm either adds some edges into the network, or generalizes the labels of some vertices toward $*$. In the worst case, the network will be anonymized to a clique. □

Surprisingly, as shown in our experiments, the algorithm can stop very quickly in practice, and the anonymization cost is relatively small. This is due to the two important properties of real social networks (vertex degree in power law distribution and small-world phenomenon). In our algorithm, when the number of unanonymized vertices in *VertexList* is less than $2k$, those vertices have to be considered together in anonymization. We tested the algorithm

using different synthetic data sets, and found that in most of the time, we even do not need to conduct any anonymization for those remaining vertices. The reason is that those vertices have the same labels, very low degree (1 in many sparse social networks), and isomorphic neighborhoods.

Our algorithm involves graph isomorphism testing. As shown in Theorem 4, up to $\frac{n(n-1)}{2}$ edges may be added, where n is the number of vertices in the graph. Adding an edge takes $O(f)$ time where f is the largest degree in the graph. Without loss of generality, we can assume $f \ll n$. Thus, the anonymization algorithm has the same time complexity as the graph isomorphism problem.

Although the general framework of Algorithm 1 may be used to tackle the case where edge additions and deletions are simultaneously allowed, there are several important concerns when edge deletions are allowed. When only edge additions are allowed, we anonymized vertices in degree descending order. Each time to construct an equivalence group of vertices (that is, finding a set of vertices to be anonymized to the isomorphic neighborhoods), we only need to examine the vertices with smaller degrees which have not been anonymized yet. However, when edge deletions are also allowed, we need to examine the vertices of larger degrees as well. Those vertices have been anonymized in the previous steps, and thus may have to be re-anonymized. Moreover, by allowing edge additions only, the termination of Algorithm 1 in a finite number of steps is guaranteed as proved in Theorem 4. However, such a theoretical guarantee disappears if edge additions and deletions are simultaneously allowed.

5 An l-diversity method

In this section, we extend the k-anonymity method developed in Sect. 4 to tackle the l-diversity problem.

In order to achieve l-diversity in social networks, vertices have to be partitioned into equivalence groups, such that in every equivalence group of vertices, at most $\frac{1}{l}$ of the vertices are associated with the most frequent sensitive label. Following the philosophy in Machanavajjhala et al. [27], we introduce the notion of l-diverse partition in social networks.

Given a social network $G = (V, E)$ with n vertices and each vertex is associated with a non-sensitive label and a sensitive label, an *l-diverse partition* divides the vertices in V into m equivalence groups of vertices, such that $\frac{freq(c)}{|EG|} \leq \frac{1}{l}$, where $freq(c)$ is the number of vertices which carry the most frequent sensitive label c in group EG, and $|EG|$ is the number of vertices in the corresponding equivalence group.

We have the following result for an l-diverse partition, which is similar to the one in Xiao and Tao [38].

Theorem 5 **(l-diverse partition)** *In a social network $G = (V, E)$ where $|V| = n$ and each vertex carries a sensitive label, there exists an l-diverse partition of the n vertices if and only if there are at most $\frac{n}{l}$ vertices associated with a sensitive label.*

Proof **(Necessity)** Since G has a valid l-diverse partition, we denote the m equivalence groups of vertices by $EG_1, EG_2, \ldots, EG_m$, respectively. Consider a sensitive label x which appears in groups $EG_{i_1}, \ldots, EG_{i_w} (1 \leq i_1 < \cdots < i_w \leq m)$, according to the definition of l-diverse partition, label x appears at most $\frac{1}{l} \times \left|EG_{i_j}\right|$ times in group $EG_{i_j} (1 \leq j \leq w)$. As a result, the total number of occurrences of x is at most $\sum_{j=1}^{w} \frac{1}{l} \times \left|EG_{i_j}\right| = \frac{1}{l} \sum_{j=1}^{w} \left|EG_{i_j}\right| \leq \frac{n}{l}$.

(Sufficiency) We construct a valid l-diverse partition as follows. First, we put all vertices into a candidate set $Cand$. In each iteration, from $Cand$ we find the top-l most frequent

Algorithm 2 Anonymizing a social network to achieve l-diversity.

Input: a social network $G = (V, E)$, the anonymization requirement parameter l, the cost function parameters α, β and γ;
Output: an anonymized graph G';
Initialization: initialize $G' = G$, and mark $v_i \in V(G)$ as "unanonymized";
1: sort $v_i \in V(G)$ as *VertexList* in neighborhood size descending order
2: **while** *VertexList* $\neq \emptyset$ **do**
3: let *SeedVertex* = *VertexList.head()* and remove it from *VertexList*;
4: **for** each $v_i \in$ *VertexList* **do**
5: calculate *Cost(SeedVertex, v_i)* using the anonymization method for two vertices;
6: **end for**
7: **if** *VertexList.size()* $\geq 2l - 1$ **then**
8: let *CandidateSet* = **Partition**(*VertexList*);
9: **else**
10: let *CandidateSet* contain the remaining unanonymized vertices;
11: **end if**
12: suppose *CandidateSet* $= \{u_1, \ldots, u_m\}$, anonymize *Neighbor(SeedVertex)* and *Neighbor(v_{j-1})* as discussed in Section 4.2.2;
13: **for** $j = 2$ to m **do**
14: anonymize *Neighbor*(u_j) and {*Neighbor*(*SeedVertex*), *Neighbor*(u_1), ..., *Neighbor*(u_{j-1})} as discussed in Section 4.2.2, mark them as "anonymized";
15: update *VertexList*;
16: **end for**
17: **end while**

Function: Partition(*VertexList*)
18: initialize $S = \emptyset$;
19: place $v_i \in$ *VertexList* into chunks such that vertices in each chunk carry the same sensitive label;
20: let S contain top $l - 1$ vertices from $l - 1$ different chunks (exclude the chunk whose sensitive label is same to the one for *SeedVertex*) with smallest *Cost* and remove them from *VertexList*;
21: **while** not exist a valid l-diverse partition for *VertexList* **do**
22: find z with smallest *Cost* from the chunk of the most frequent sensitive label in *VertexList*;
23: find z' in S with largest *Cost* and different sensitive label of z, replace z' with z, and place z' back into *VertexList*;
24: **end while**
25: **return** S;

labels, and pick a vertex for each of such a label. Those l vertices form an equivalence group and are removed from $Cand$. We continue until there are less than $2l$ vertices left in $Cand$. Those remaining vertices form the last equivalence group.

We show by induction that in each iteration except for the last one there must be at least l different sensitive labels in $Cand$. The claim holds in the first iteration, otherwise, the most frequent label must appear more than $\frac{n}{l}$ times according to the pigeon hole theorem. After the first iteration, the most frequent label must appear at most $\frac{n}{l} - 1$ times. After iteration i except for the last one, there are $n - l \cdot i$ vertices left in $Cand$. The most frequent label can appear at most $\frac{n}{l} - i$ times. Therefore, there must be at least $\frac{n-l \cdot i}{\frac{n}{l}-i} = l$ different labels left in $Cand$.

The last equivalence group contains $n - \lfloor \frac{n}{l} \rfloor l$ vertices. The most frequent label can appear at most $\frac{n}{l} - \lfloor \frac{n}{l} \rfloor l \leq 1$ times. Thus, the last equivalence group is an l-diverse group. □

Theorem 5 provides an efficient way to determine whether an l-diverse partition exists for a given social network. We only need to examine the most frequent sensitive label(s). By incorporating the method for k-anonymity in Algorithm 1, we can derive a method for l-diversity in social network data, as shown in Algorithm 2.

Algorithm 2 follows a framework similar to that in Algorithm 1. In order to achieve l-diversity, Line 8 is changed. Rather than greedily picking the top $l-1$ vertices with the smallest *Cost* into *CandidateSet*, we enforce the constraint that the vertices in *CandidateSet* and *SeedVertex* should contain at least l different sensitive labels. For a selected *SeedVertex*, the algorithm identifies $l-1$ vertices with the smallest *Cost* to *SeedVertex* such that the $l-1$ vertices carry unique sensitive labels different from that of *SeedVertex*. Those vertices form the *CandidateSet*.

Once a *CandidateSet* is formed, the algorithm examines if there exists a valid l-diverse partition for the remaining vertices in *VertexList* using Theorem 5. If so, the *CandidateSet* is valid and the anonymization can continue on the vertices in *CandidateSet*. Otherwise, we need to update *CandidateSet*. Theorem 5 indicates that we only need to consider the most frequent sensitive labels in *VertexList*. The algorithm finds the most frequent sensitive label in *VertexList*, and picks the vertex z of the label with the smallest *Cost*. In *CandidateSet*, the vertex z' with the largest *Cost* and a sensitive label different from z is replaced by z. The algorithm iterates until a valid *CandidateSet* is found, guaranteed by Theorem 5.

6 Empirical evaluation

In this section, we report a systematic empirical study to evaluate our anonymization method using both real datasets and synthetic datasets. All the experiments were conducted on a PC running the Microsoft Windows XP SP2 Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk. The program was implemented in C/C++ and was compiled using Microsoft Visual Studio .NET 2005.

Our experiments were conducted in four steps. First, we will show that the neighborhood attacks exist in real data sets, which lead to a serious privacy problem. Second, we will examine the anonymization performance of our method, including the number of dummy edges, the label certainty penalty and the running time. Third, we will evaluate the query performance for aggregate social queries using our anonymized social network data. Fourth, we will examine the diversity issues in social network data.

6.1 Neighborhood attacks in real data

We used a real co-authorship data set from KDD Cup 2003 to examine whether neighborhood attacks may happen in practice. The data set was from the e-print arXiv (http://arXiv.org)and contains a subset of papers in the high-energy physics section of the arXiv. The LaTeX sources of all papers are provided. We extracted author names from the data sources and constructed a co-authorship graph. Each vertex in the graph represents an author, and two vertices are linked by an edge if the two corresponding authors co-authored at least one paper in the data set. There are 57,448 vertices and 120,64 edges in the co-authorship graph and the average number of vertex degrees is about 4.

We tested two ways to preserve the privacy of authors by generalizing labels. In the first method, we removed all labels, that is, author names. In the second method, we use the author's affiliation as the label, that is, all authors from the same institution have the same label. After generalization, for each vertex, we extracted its neighborhood and counted the number of other vertices in the graph that have the isomorphic neighborhoods. Table 1 shows the percentages of vertices whose neighborhoods violate the k-anonymity.

Table 1 The percentages of vertices violating k-anonymity in the co-authorship data

k	Removing labels (%)	Generalizing to affiliations (%)
5	1.3	12.7
10	3.9	16.1
15	7.1	19.4
20	12.0	23.2

Table 1 clearly shows that the neighborhood attacks are a real issue for social network data publishing. When the value of k increases, the number of vertices violating k-anonymity increases. Moreover, the more specific are the vertex labels, the more vertices fail the k-anonymity. Anonymizing labels only cannot prevent neighborhood attacks effectively.

6.2 Anonymization performance

We use the R-MAT graph model [6] to generate synthetic data sets. R-MAT can generate graphs with power law vertex degree distribution and small-world characteristic, which are the two most important properties for many real-world social networks.

R-MAT takes four probability parameters, namely a, b, c and d. Consider the adjacency matrix representation A of the graph. Assume that each element a_{ij} in A is non-zero if there exists an edge between vertices i and j. Given the number of vertices n and the number of edges m, R-MAT starts with an empty $n \times n$ adjacency matrix A. It recursively divides the adjacency matrix into four equal-size partitions, and distributes edges within these partitions with a set of unequal probabilities. Each edge chooses one of the four partitions with probabilities a, b, c and d, respectively, such that $a+b+c+d=1$. The parameters a, b, c and d can be determined based on the required community structure and vertex degree distribution, as detailed in Chakrabarti et al. [6]. Chakrabarti et al. [6] conjectured that the ratios a/b and a/c are approximately $3/1$ in many real-world graphs, and $a \geq d$.

We used the default values of 0.45, 0.15, 0.15, and 0.25 for those four parameters, respectively. We generated a series of synthetic data sets by varying the number of vertices from 5,000 to 25,000 and the average vertex degree from 3 to 7. Hereafter, we refer to a synthetic data set as $D(n, d)$, where n and d are the number of vertices and the average vertex degree, respectively. The edge weight was set in the range [0, 100] by default. We assigned each vertex a label based on its average edge weight. A simple two level hierarchy structure for those labels was generated.

We first examined the effect of parameters α, β and γ in the anonymization quality measure. β was set to 1 as the base. We changed the values of α and γ, and measured the number of edges added and the normalized certainty penalty incurred in the anonymized graphs. The results for data set $D(15K, 5)$ and $k = 10$ are shown in Fig. 7. The results show that we can trade off between adding edges and generalizing labels by tuning the three parameters. Often adding less edges is more desirable in anonymizing a social network since the network structures can be preserved better. We observed that, on the synthetic data sets, when $\alpha = 100$ and $\gamma = 1.1$, the number of edges added is small and the normalized certainty penalty is moderate. Hereafter, we use 100, 1, and 1.1 as the default values for α, β and γ.

Figure 8 reports the anonymization quality on various synthetic data sets with respect to different k values, and shows the anonymization cost in both the number of edges added and the normalized certainty penalty. First, when the number of vertices increases, the anonymization cost increases. However, the increase of the number of edges added is sub-linear, since

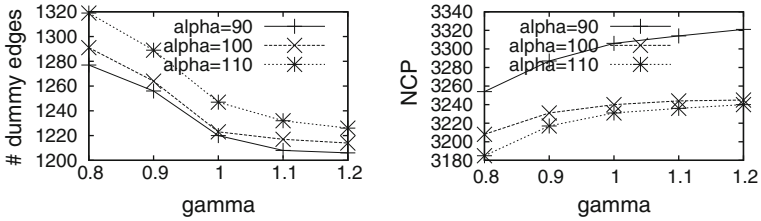


Fig. 7 The effect of parameters in anonymization quality measure

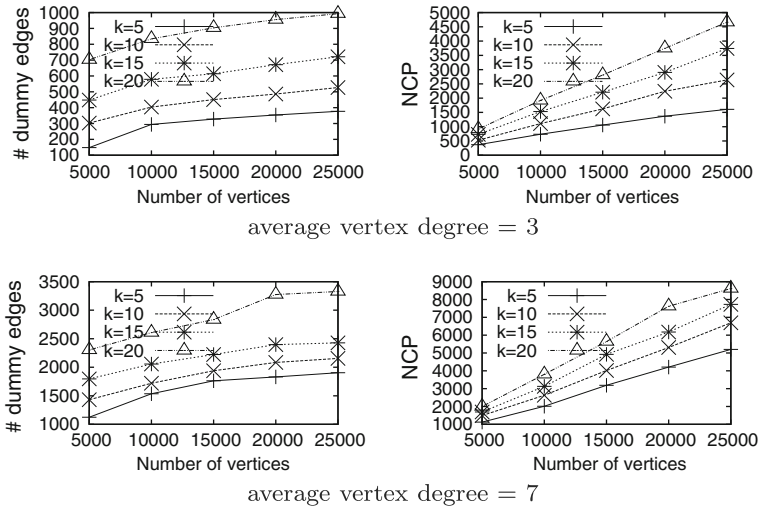


Fig. 8 Anonymization cost on various synthetic data sets

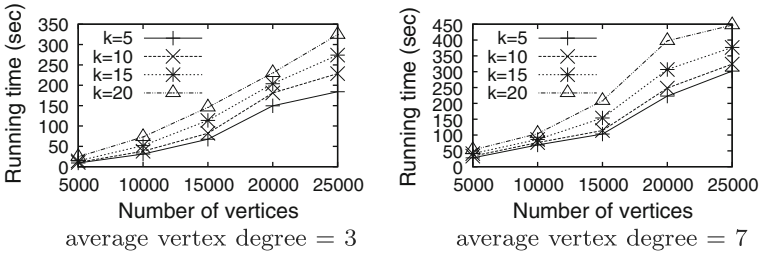


Fig. 9 The runtime on various synthetic data sets

in a larger network it is more likely to find similar neighborhoods. Second, when k increases, the anonymization cost also increases, because more neighborhoods need to be anonymized in a group. Last, when the average number of vertex degree increases, the anonymization cost increases, too. In a denser network, the neighborhoods are more diverse and more edges are needed to anonymize different neighborhoods.

Figure 9 shows the runtime on various synthetic data sets with respect to different k values. The runtime increases when the average vertex degree increases, since the network becomes denser. Moreover, the larger the k, the longer the runtime since more neighborhoods in a group need to be anonymized.

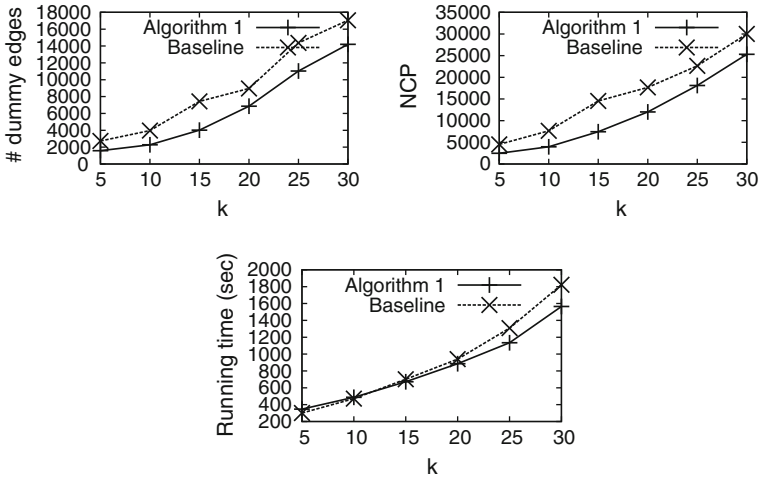


Fig. 10 Anonymizing the KDD cup 2003 co-authorship data set

The roughly linear scalability is due to the following reasons. First, the average cost in time to compute the neighborhood component code for a vertex does not change substantially when the number of vertices increases since the graphs are scale free. Second, the number of times that neighborhood component codes need to be computed is approximately linear to the number of vertices. Last, once the neighborhood component codes are computed, the graph isomorphism tests are linear to the number of vertices in the neighborhoods, which is often much smaller than the number of vertices in the graph and can be practically bounded by a small integer.

6.3 Anonymizing the co-authorship data set

We built a three-level label hierarchy for the KDD cup 2003 co-authorship data set. The leaves are the labels of author affiliations. The middle level contains the countries of the authors' affiliations. The root is the most general label $*$. We anonymized the co-authorship data set using our method. The number of edges added, the normalized certainty penalty and the runtime with different values of k (varied from 5 to 30) are shown in Fig. 10. Comparing to the total number of edges in the data set (120, 640), the number of edges added is less than 12% even when $k = 30$. Moreover, the runtime is scalable with respect to k.

We compare our method to the following baseline algorithm which conducts a greedy degree-based matching. When anonymizing two neighborhoods, we first try to match vertices in the two neighborhoods such that each vertex in one neighborhood is uniquely matched to one vertex in the other neighborhood. The vertex matching takes a degree-first and label-later approach. To find a vertex in neighborhood H' to match vertex u in neighborhood H, we first consider all vertices in H' that have the same degree as u. If there are multiple such vertices, we pick the one whose label is the closest to the label of u. We break any tie by a random selection. If there are no vertices in H' that have the same degree as u, we pick a vertex v in H' which has the largest degree smaller than the degree of u. Again, if there are multiple available vertices, we make selection based on the labels. Since v has a smaller degree than u, we link v to vertices of small degrees in *VertexList* to increase the degree of v to the same as u. The vertex matching procedure is conducted in the degree descending

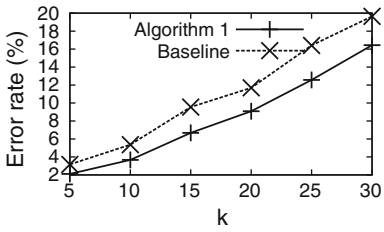


Fig. 11 Query answering on the KDD Cup 2003 co-authorship data set

order of vertices in H. Once all the vertices are matched, we add necessary edges to the corresponding neighborhoods to make the two neighborhoods isomorphic.

Figure 10 compares our method and the baseline method. Our method outperforms the baseline method in anonymization quality, since our method takes into account more neighborhood information during matching. The running time of the baseline method is slightly better than our method when k is small. When k increases, the running time of the baseline method increases and performs worse than our method. The baseline method often has to change the neighborhood structures of some vertices which have been anonymized in the previous steps, thus the algorithm needs to conduct more rounds than our method. The experimental results indicate that the heuristics considered in the paper improve the performance of anonymization.

6.4 Aggregate query answering and more

To test the utility of the anonymized social networks, we conducted aggregate network queries on the KDD Cup 2003 co-authorship data set, and the anonymized networks. For two labels l_1 and l_2 in the data set, we calculated the average distance from a vertex with label l_1 to its nearest vertex with label l_2. Since labels are organized in a hierarchy structure, when we calculated the average distance, we also considered distance from vertices with labels l_1' to vertices with labels l_2' such that $l_1 \preceq l_1'$ and $l_2 \preceq l_2'$. The error rate is $\frac{d-d'}{d}$, where d and d' are the average distances in the original network and in the anonymized network, respectively. We randomly picked ten label pairs from the label hierarchy, and calculated the average error rate of them. The results are shown in Fig. 11. After the anonymization, with some edges added, the average distance decreases. Therefore, the error rate is always positive. However, the error rate is small even when k is up to 20, since the number of edges added is small, as shown in Fig. 10.

As a comparison, in Fig. 11, we also plot the query answering performance using the anonymized data generated from the baseline method discussed in Sect. 6.3. Our method achieves a lower error rate.

To further test the utility of the anonymized data, we measure several graph properties using the original graph data and the anonymized data, including the vertex degree distribution and the clustering co-efficient (CC). Figure 12 shows the results. In the anonymized data, the clustering co-efficient decreases slightly when k increases. The new edges inserted during anonymization makes the graph shrink. However, the clustering co-efficient of the anonymized graph is still quite close to the original value. Even when $k = 30$, the difference is only 0.03. We also plot the degree distribution of the original graph and the anonymized graphs when k is set to 10 and 20. The degree distributions are very similar.

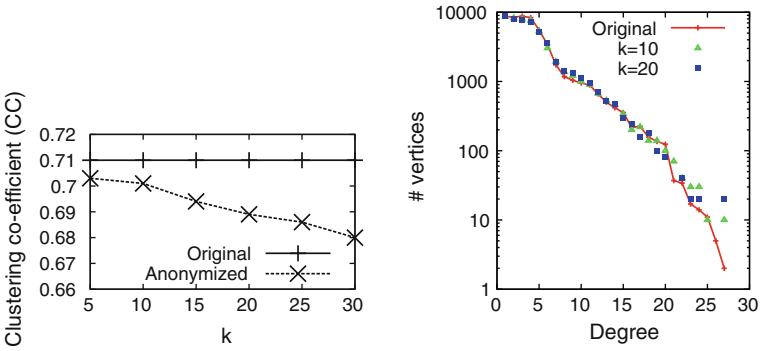


Fig. 12 The data utility in anonymized co-authorship data

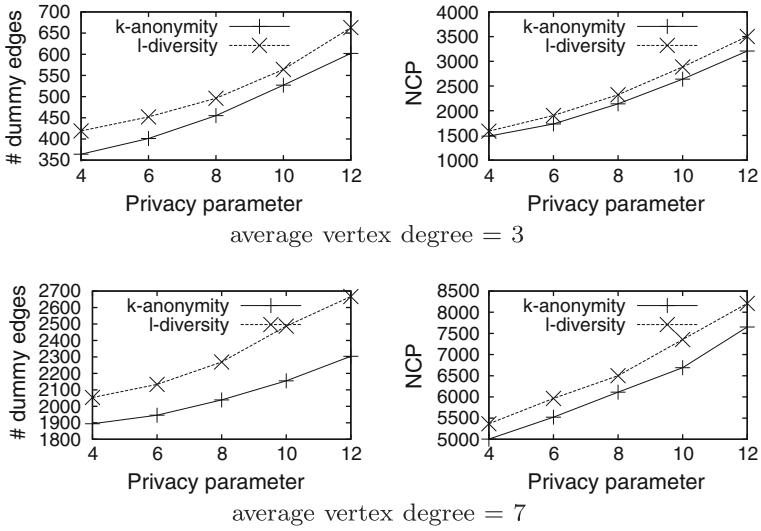


Fig. 13 Comparisons of anonymization cost using k-anonymity and l-diversity on synthetic data sets

6.5 l-Diversity in social network data

In order to examine the effectiveness of l-diversity in social network data, we used the synthetic data set generated from the R-MAT graph generator in Sect. 6.2. We treated the original labels in the graph as non-sensitive labels. In order to assign a sensitive label to each vertex in the graph, we adopted a random number generator. A uniformly distributed random number in the range [0, 100] was assigned to each vertex as a sensitive label. We used 100, 1, and 1.1 as the default values for α, β and γ in the information loss function.

We first compared the information loss of k-anonymity and l-diversity. The two data sets we used both contain 25,000 vertices, and the average vertex degree is 3 and 7, respectively. We varied the number of k and l from 4 to 12, and examined the number of fake edges added, as well as the normalized certainty penalty. The results are shown in Fig. 13.

Figure 13 reports the anonymization quality on various synthetic data sets with respect to different k and l values, and shows the anonymization cost in both the number of edges

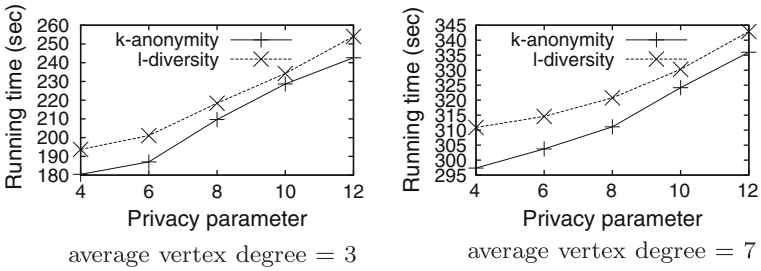


Fig. 14 The comparison of runtime using k-anonymity and l-diversity on synthetic data sets

added and the normalized certainty penalty. When the value of k and l increases, the anonymization cost also increases, because more neighborhoods need to be anonymized in a group. Generally, the cost for achieving l-diversity is slightly larger than that for achieving k-anonymity. This is because during the l-diversity anonymization, in order to make sure that each equivalence group of vertices contains at least l different sensitive labels, some vertices may have large anonymization cost with respect to the seed vertex. Moreover, when the average number of vertex degree increases, the anonymization cost increases, too. In a denser network, the neighborhoods are more diverse and more edges are needed to anonymize different neighborhoods.

Figure 14 shows the runtime on the same synthetic data sets with respect to different k and l values. The runtime increases when the average vertex degree increases, since the network becomes denser. Moreover, the larger the k and l, the longer the runtime since more neighborhoods in a group need to be anonymized. Furthermore, the runtime for l-diversity is 10% larger than that for k-anonymity. There are two reasons. First, achieving l-diversity needs additional operations for l-diverse partition. Second, according to the information loss in Fig. 13, the cost for achieving l-diversity is larger than that for achieving k-anonymity.

7 Discussion and conclusions

In this paper, we tackled the novel and important problem of preserving privacy in social network data, and took an initiative to combat neighborhood attacks. We modeled the problem systematically and developed a practically feasible approach.We identified the diversity issue in social network data anonymization, and proposed an efficient solution. An extensive empirical study using both a real data set and a series of synthetic data sets strongly indicated that neighborhood attacks are real in practice, and our method is highly feasible. Moreover, anonymized social networks can still be used to answer aggregate queries accurately.

As social network data is much more complicated than relational data, privacy preserving in social networks is much more challenging and needs many serious efforts in the future. Particularly, modeling adversarial attacks and developing privacy preservation strategies are critical. Privacy preservation in social networks is a relatively new research direction. There is much future work needed to be done. For example, in this paper, we modeled the complete 1-neighborhood structure as the background knowledge; in practice, partial background knowledge of the 1-neighborhood structures is more realistic. This introduces new challenges for background knowledge modeling, meanwhile opens new opportunities to improve the utility of the anonymized social network data. Moreover, negative 1-neighborhood background knowledge is also popular in practice (for example, an adversary knows that the number of

friends of a target victim is not larger than ten, or there doesn't exist three friends of the target victim that they know each other, etc.). Considering different kinds of background knowledge, the privacy preservation model and methods in social network data can be completely different. Furthermore, there may be various kinds of other privacy attacks in the social network data, thus effective and efficient anonymization methods with respect to different attacks are quite interesting.

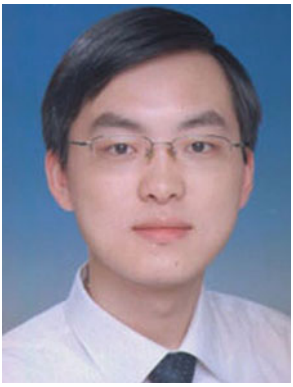
References

1. Adamic L, Adar E (2005) How to search a social network. Soc Netw 27(3):187–203
2. Backstrom L, Huttenlocher D, Kleinberg J, Lan X (2006) Group formation in large social networks: membership, growth, and evolution. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'06), ACM Press, New York, pp 44–54
3. Backstrom L, Dwork C, Kleinberg J (2007) Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In: Proceedings of the 16th international conference on World Wide Web (WWW'07), ACM Press, New York, pp 181–190
4. Bhagat S, Cormode G, Krishnamurthy B, Srivastava D (2009) Class-based graph anonymization for social network data. PVLDB 2(1):766–777
5. Campan A, Truta TM (2008) A clustering approach for data and structural anonymity in social networks. In: Proceedings of the 2nd ACM SIGKDD international workshop on privacy, security, and trust in KDD (PinKDD'08), in conjunction with KDD'08, Las Vegas, Nevada
6. Chakrabarti D, Zhan Y, Faloutsos C (2004) R-mat: a recursive model for graph mining. In: Proceedings of the 2004 SIAM international conference on data mining (SDM'04), SIAM, Philadelphia
7. Cormen TH, Leiserson CE, Rivest RL, Stein C (2002) Introduction to algorithms, 2nd edn. MIT Press and McGraw-Hill, Cambridge
8. Cormode G, Srivastava D, Yu T, Zhang Q (2008) Anonymizing bipartite graph data using safe groupings. PVLDB 1(1):833–844
9. Coull SE, Monrose F, Reiter MK, Bailey M (2009) The challenges of effectively anonymizing network data. In: Proceedings of the 2009 cybersecurity applications & technology conference for homeland security (CATCH'09), IEEE Computer Society, Washington, DC, pp 230–236
10. Dwork C (2008) Differential privacy: a survey of results. In: Proceedings of the 5th international conference on theory and applications of models of computation. Lecture notes in computer science, vol 4978. Springer, pp 1–19
11. Dwork C, Smith A (2008) Differential privacy for statistics: what we know and what we want to learn. In: Proceedings of NCHS/CDC data confidentiality workshop
12. Faloutsos M, Faloutsos P, Faloutsos C (1999) On power law relationships of the internet topology. In: Proceedings of the conference on applications, technologies, architectures, and protocols for computer communication (SIGCOMM'99), ACM Press, New York, pp 251–262
13. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York
14. Getoor L, Diehl CP (2005) Link mining: a survey. ACM SIGKDD Explor Newsl 7(2):3–12
15. Gkoulalas-Divanis A, Verykios VS (2009) Hiding sensitive knowledge without side effects. Knowl Inf Syst 20(3):263–299
16. Hay M, Miklau G, Jensen D, Weis P, Srivastava S (2007) Anonymizing social networks. Tech. Rep. 07-19, University of Massachusetts Amherst
17. Hay M, Miklau G, Jensen D, Towsley D (2008) Resisting structural identification in anonymized social networks. PVLDB 1(1):102–114
18. Hay M, Li C, Miklau G, Jensen D (2009) Accurate estimation of the degree distribution of private networks. In: Proceedings of the 2009 ninth IEEE international conference on data mining (ICDM'09), IEEE Computer Society, Washington, DC, pp 169–178
19. Hazan E, Safra S, Schwartz O (2003) On the complexity of approximating k-dimensional matching. In: Proceedings of the 6th international workshop on approximation algorithms for combinatorial optimization problems and of the 7th international workshop on randomization and computation techniques in computer science (RANDOM-APPROX'03), LNCS, vol 2764. Springer, Berlin, pp 83–97
20. Korolova A, Motwani R, Nabar SU, Xu Y (2008) Link privacy in social networks. In: Proceedings of the 24th international conference on data engineering (ICDE'08), IEEE, pp 1355–1357
21. Kossinets G, Watts DJ (2006) Empirical analysis of an evolving social network. Science 311(5757):88–90

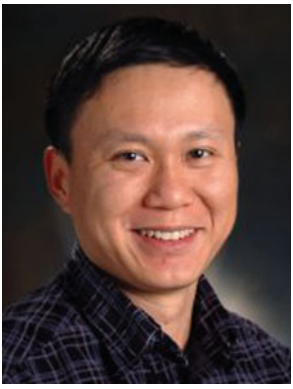
22. Kumar R, Novak J, Tomkins A (2006) Structure and evolution of online social networks. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06), ACM Press, New York, pp 611–617
23. Li N, Li T, Venkatasubramanian S (2007) t-Closeness: privacy beyond k-anonymity and l-diversity. In: Proceedings of the 23rd international conference on data engineering (ICDE'07), IEEE, pp 106–115
24. Liu K, Terzi E (2008) Towards identity anonymization on graphs. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data (SIGMOD'08), ACM Press, New York, pp 93–106
25. Liu K, Das K, Grandison T, Kargupta H (2008) Privacy-preserving data analysis on graphs and social networks. In: Kargupta H, Han J, Yu P, Motwani R, Kumar V (eds) Next generation data mining. CRC Press, Boca Raton
26. Luo H, Fan J, Lin X, Zhou A, Bertino E (2009) A distributed approach to enabling privacy-preserving model-based classifier training. Knowl Inf Syst 20(2):157–185
27. Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) L-diversity: privacy beyond k-anonymity. In: Proceedings of the 22nd IEEE international conference on data engineering (ICDE'06), IEEE Computer Society, Washington, DC
28. Machanavajjhala A, Kifer D, Abowd JM, Gehrke J, Vilhuber L (2008) Privacy: theory meets practice on the map. In: Proceedings of the 24th international conference on data engineering (ICDE'08), pp 277–286
29. Meyerson A, Williams R (2004) On the complexity of optimal k-anonymity. In: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems (PODS'04), ACM, New York, pp 223–228
30. Muhlestein D, Lim S (2010) Online learning with social computing based interest sharing. Knowl Inf Syst. doi:10.1007/s10115-009-0265-4
31. Qiu L, Li Y, Wu X (2008) Protecting business intelligence and customer privacy while outsourcing data mining tasks. Knowl Inf Syst 17(2):99–120
32. Rastogi V, Hay M, Miklau G, Suciu D (2009) Relationship privacy: output perturbation for queries with joins. In: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems (PODS'09), ACM, New York, pp 107–116
33. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng (TKDE) 13(6):1010–1027
34. Samarati P, Sweeney L (1998) Generalizing data to provide anonymity when disclosing information. In: Proceedings of the 7th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems (PODS'98), ACM Press, New York, p 188
35. Sweeney L (2002) K-anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl Based Syst 10(5):557–570
36. Wang DW, Liau CJ, Hsu TS (2006) Privacy protection in social network data disclosure based on granular computing. In: Proceedings of the 2006 IEEE international conference on fuzzy systems, Vancouver, BC, pp 997–1003
37. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, New York
38. Xiao X, Tao Y (2006) Anatomy: simple and effective privacy preservation. In: Dayal U, Whang KY, Lomet DB, Alonso G, Lohman GM, Kersten ML, Cha SK, Kim YK (eds) Proceedings of the 32nd international conference on very large data bases (VLDB'06), ACM, pp 139–150
39. Xiao X, Tao Y (2006b) Personalized privacy preservation. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD'06), ACM Press, New York, pp 229–240
40. Xiao X, Tao Y (2007) M-invariance: towards privacy preserving re-publication of dynamic datasets. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data (SIGMOD'07), ACM, New York, pp 689–700
41. Xiao X, Tao Y (2008) Output perturbation with query relaxation. PVLDB 1(1):857–869
42. Xu J, Wang W, Pei J, Wang X, Shi B, Fu AWC (2006) Utility-based anonymization using local recoding. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'06), ACM Press, New York, pp 785–790
43. Yan X, Han J (2002) gspan: graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE international conference on data mining (ICDM'02), IEEE Computer Society, Washington, DC, p 721
44. Yan X, Yu PS, Han J (2004) Graph indexing: a frequent structure-based approach. In: Proceedings of the 2004 ACM SIGMOD international conference on management of data (SIGMOD'04), ACM Press, New York, pp 335–346
45. Ying X, Wu X (2008) Randomizing social networks: a spectrum preserving approach. In: Proceedings of the 2008 SIAM international conference on data mining (SDM'08), SIAM, pp 739–750

46. Ying X, Wu X (2009a) On link privacy in randomizing social networks. In: Proceedings of the 13th Pacific-Asia conference on advances in knowledge discovery and data mining, Springer, pp 28–39
47. Ying X, Wu X (2009b) On randomness measures for social networks. In: Proceedings of the 2009 SIAM international conference on data mining, SIAM, pp 709–720
48. Zheleva E, Getoor L (2007) Preserving the privacy of sensitive relationships in graph data. In: Proceedings of the 1st ACM SIGKDD workshop on privacy, security, and trust in KDD (PinKDD'07)
49. Zhou B, Pei J (2008) Preserving privacy in social networks against neighborhood attacks. In: Proceedings of the 24th IEEE international conference on data engineering (ICDE'08), IEEE Computer Society, Cancun, pp 506–515
50. Zhou B, Pei J, Luk WS (2008) A brief survey on anonymization techniques for privacy preserving publishing of social network data. SIGKDD Explor 10(2):12–22
51. Zou L, Chen L, Özsu MT (2009) K-automorphism: a general framework for privacy preserving network publication. PVLDB 2(1):946–957

Author Biographies



Bin Zhou received his B.Sc. degree in Computer Science from Fudan University, China, in 2005 and his M.Sc. degree in Computing Science from Simon Fraser University, Canada, in 2007. He is currently a Ph.D. candidate in School of Computing Science at Simon Fraser University, Canada. His research interests lie in graph analysis, graph mining, data privacy, and their relations to Web-scale data management and mining, as well as their applications in Web search engines.



Jian Pei is an Associate Professor at the School of Computing Science at Simon Fraser University, Canada. His research interests can be summarized as developing effective and efficient data analysis techniques for novel data intensive applications. He is currently interested in various techniques of data mining, Web search, information retrieval, data warehousing, online analytical processing, and database systems, as well as their applications in social networks, health-informatics, business and bioinformatics. His research has been supported in part by government funding agencies and industry partners. He has published prolifically and served regularly for the leading academic journals and conferences in his fields. He is an associate editor of ACM Transactions on Knowledge Discovery from Data (TKDD) and IEEE Transactions of Knowledge and Data Engineering (TKDE). He is a senior member of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). He is the recipient of several prestigious awards.