Singapore Management University

Institutional Knowledge at Singapore Management University

Dissertations and Theses Collection (Open Access)

Dissertations and Theses

12-2014

Data Preparation for Social Network Mining and Analysis

Yazhe WANG
Singapore Management University, yazhe.wang.2008@phdis.smu.edu.sg

Follow this and additional works at: https://ink.library.smu.edu.sg/etd_coll

Part of the Databases and Information Systems Commons, Numerical Analysis and Scientific Computing Commons, and the Social Media Commons

Citation

WANG, Yazhe. Data Preparation for Social Network Mining and Analysis. (2014). 1-139.
Available at: https://ink.library.smu.edu.sg/etd_coll/101

This PhD Dissertation is brought to you for free and open access by the Dissertations and Theses at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Dissertations and Theses Collection (Open Access) by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Data Preparation for Social Network Mining and Analysis

YAZHE WANG

SINGAPORE MANAGEMENT UNIVERSITY

2014

Data Preparation for Social Network Mining and Analysis

by

Yazhe Wang

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Baihua Zheng (Supervisor/Chair)
Associate Professor of Information Systems
Singapore Management University

HweeHwa Pang
Professor of Information Systems
Singapore Management University

Kyriakos Mouratidis
Associate Professor of Information Systems
Singapore Management University

Gao Cong
Assistant Professor of Computer Engineering
Nanyang Technological University

Singapore Management University
2014

Copyright (2014) Yazhe Wang

Data Preparation for Social Network Mining and Analysis

by

Yazhe Wang

Abstract

This dissertation studies the problem of preparing good-quality social network data for data analysis and mining. Modern online social networks such as Twitter, Facebook, and LinkedIn have rapidly grown in popularity. The consequent availability of a wealth of social network data provides an unprecedented opportunity for data analysis and mining researchers to determine useful and actionable information in a wide variety of fields such as social sciences, marketing, management, and security. However, raw social network data are vast, noisy, distributed, and sensitive in nature, which challenge data mining and analysis tasks in storage, efficiency, accuracy, etc. Many mining algorithms cannot operate or generate accurate results on the vast and messy data. Thus social network data preparation deserves special attention as it processes raw data and transforms them into usable forms for data mining and analysis tasks. Data preparation consists of four main steps, namely data collection, data cleaning, data reduction, and data conversion, each of which deals with different challenges of the raw data. In this dissertation, we consider three important problems related to the data collection and data conversion steps in social network data preparation.

The first problem is the sampling issue for social network data collection. Restricted by processing power and resources, most research that analyzes user-generated content from social networks relies on samples obtained via social network APIs. But the lack of consideration for the quality and potential bias of the samples reduces the effectiveness and validity of the analysis results. To fill this gap, in the first work of the dissertation, we perform an exploratory analysis of data samples obtained from social network stream APIs to understand the representativeness of

the samples to the corresponding complete data and their potential for use in various data mining tasks.

The second problem is the privacy protection issue at the data conversion step. We discover a new type of attacks in which malicious adversaries utilize the connection information of a victim (anonymous) user to some known public users in a social network to re-identify the user and compromise identity privacy. We name this type of attacks connection fingerprint (CFP) attacks. In the second work of the dissertation, we investigate the potential risk of CFP attacks on social networks and propose two efficient k-anonymity-based network conversion algorithms to protect social networks against CFP attacks and preserve the utility of converted networks.

The third problem is the utility issue in privacy preserving data conversion. Existing k-anonymization algorithms convert networks to protect privacy via modifying edges, and they preserve utility by minimizing the number of edges modified. We find this simple utility model cannot reflect real utility changes of networks with complex structure. Thus, existing k-anonymization algorithms designed based on this simple utility model cannot guarantee generating social networks with high utility. To solve this problem, in the third work of this dissertation, we propose a new utility benchmark that directly measures the change on network community structure caused by a network conversion algorithm. We also design a general k-anonymization algorithm framework based on this new utility model. Our algorithm can significantly improve the utility of generated networks compared with existing algorithms.

Our work in this dissertation emphasizes the importance of data preparation for social network analysis and mining tasks. Our study of the sampling issue in social network collection provides guidelines for people to use or not to use sampled social network content data for their research. Our work on privacy preserving social network conversion provides methods to better protect the identity privacy of social network users and maintain the utility of social network data.

Contents

List of Figures

List of Tables

Acknowledgments

It would not have been possible to complete this dissertation without the help and support of many people around me, to only some of whom it is possible to give particular mention here.

Above all, I would like to express the deepest appreciation to my advisor Professor Baihua Zheng. Her good advice, support, and knowledge have been invaluable on both an academic and a personal level, for which I am extremely grateful. Without her guidance and persistent help this dissertation would not have been possible. I would like to thank my committee members, Professor HweeHwa Pang, Professor Kyriakos Mouratidis, and Professor Gao Cong, for reviewing my dissertation and providing insightful comments and feedback.

In addition, I thank Professor Ee-Peng Lim, Professor Steven Miller, and Professor Stephen E. Fienberg who provided me with a great opportunity to study in Carnegie Mellon University for ten months. I am also thankful to Professor Jamie Callan for supervising me in CMU. The work I did with him becomes one important part of this dissertation.

Last but by no means least, I would like to acknowledge the financial, academic, and technical support of Singapore Management University and its staff, in particular, Ong Chew Hong, Seow Pei Huan, Chua Kian Peng, Alenzia Wong Poh Luan, Angela Kwek Renfeng, and Fong Soon Keat.

Chapter 1

Introduction

1.1 Social Network Data Analysis: A Brief Introduction

The proliferation of online social networks has been one of the most remarkable Internet events in this decade. Because of the high penetration of Internet-enabled devices such as personal computers, smart phones, and tablets, online social networks have become easily accessible platforms for users to communicate and share information. Many popular online social networks such as Twitter, Facebook, LinkedIn, and Google Plus have been growing rapidly. According to a market report in 2013, nearly one in four people worldwide uses social networks. The number of social network users around the world rises from 1.47 billion in 2012 to 1.73 billion in 2013, an 18 percent increase. By 2017, the global social network audience will reach 2.55 billion [27]. Another study in April 2013 reveals that social networking has been ranked as the most popular content category in worldwide engagement, accounting for 27 percent of all time spent online [28].

Owing to the popularity of online social networking, the amount of social network data available has been increasing rapidly. For example, every minute of a day, 350,000 tweets are generated on Twitter [1], over 3000 pictures are uploaded to

[1] http://www.internetlivestats.com/twitter-statistics/

Flickr [2], and around 3.3 million items are shared on Facebook [3]. These data provide unprecedented opportunities for data analysis research. The primary objectives of social network analysis are to handle large-scale social network data, extract actionable patterns, and gain insightful knowledge about dynamic and multifaceted social networks. Therefore, social network analysis is of significant value for many application domains such as policy making, advertising, and homeland security. Social network data available for analysis are usually voluminous, structurally complex, heterogeneous, and dynamic in nature, and can be broadly classified as *content* and *linkage* data [2]. Content data contain texts, images, and other multimedia data, which are explicitly generated by social network users. Linkage data are essentially graphs where individual users are represented by vertices, and relationships or interactions between individuals are represented by edges. On each type of social network data, a wide range of analysis tasks can be performed to reveal valuable information.

- **Content-based analysis** studies heterogeneous and unstructured content generated by social network users such as blogs, images, videos, and tags. Some popular analysis practices include opinion mining, trend detection, sentiment analysis, collaborative recommendation, etc. This type of analysis has numerous applications in business, politics, and consumer media research [79].

- **Linkage-based analysis** studies linkages among social network users. It is used to reveal the structural properties and evolution patterns of social networks, determine important vertices and social influence, detect communities, predict unobserved or future links, etc. This type of analysis is critical for various application fields such as social psychology, viral marketing, and terrorism defense [13].

It has been observed that content generation and linkage formation are closely related in social networks. Therefore, combining linkage-based analysis with content-

[2]http://blog.flickr.net
[3]http://www.connectsafely.org/tag/social-media/

based analysis can improve the quality of analysis results or discover otherwise unobserved information. For example, utilizing user profile information can greatly improve the quality of clusters detected in social networks [109], and recommendation systems, if considering the underlying social network, can provide users with more personalized content [48].

1.2 Data Preparation: Steps and Challenges

Social network data analysis is of significant importance for academia, industry, government, etc. However, like any other analysis, it is limited by the availability, quantity, and quality of data. Some analysis algorithms cannot run efficiently on huge datasets. Analysis may generate misleading results if underlying datasets are inadequate or biased. Carelessly prepared datasets may contain sensitive information, which raises concerns for privacy and security and then causes analysis projects to fail. Raw social network data are typically huge, mostly informal in nature (e.g., contain a lot of user-generated misspellings and abbreviations), and flooded with sensitive personal information of individual users. Therefore, data preparation is particularly crucial for the success of social network analysis.

The main objectives of data preparation are to process raw datasets, reduce time and space costs, enhance data quality with better interpretability and accuracy, and limit disclosure of sensitive information. The data preparation process can be detailed into four main steps namely data collection, data cleaning, data reduction, and data conversion [77]. In the following subsections, we discuss the challenges and issues in each data preparation step, with a focus on the factors related to social network data.

1.2.1 Data Collection

Researchers in the very early stage of social network research collected social network data from individuals in particular social settings through questionnaires, interviews, and surveys, which are very labor-intensive. The social network data collected were usually limited on small communities of people, which limited the scope of analysis. The development of online social media has brought a significant shift in social network research, leading to the emergence of "computational social science" [53]. It has greatly increased the availability and size of social network data, and thus has broadened the variety of disciplines contributing to the advance of social network research. In this dissertation, our discussion is based on modern online social network data.

Online social network data are collected from social network service providers who possess the overall social network data of their service users. The collection process is normally performed automatically through programs or scripts. Some social media websites such as Facebook and Twitter provide APIs for data crawling. The main challenges of social network data collection are limited processing power and storage space. Online social networks are usually huge as measured by user population size, user-generated content volume, and update velocity. Take Twitter, one of the most popular microblogging social networks, as an example. The number of registered users on Twitter reached one billion in 2013, and collectively, Twitter users now send over 500 million posts every day [86]. These numbers keep growing rapidly. Crawling such huge social networks requires robust systems with high processing power and huge storage capacity (e.g., petabytes). Moreover, social network websites usually set limits on the data access rate. For example, Twitter allows up to 15 requests per 15 minutes sent to its REST API to access the data (https://dev.twitter.com), and Foursquare sets a 500-limit on the number of requests sent per hour (https://developer.foursquare.com). These rate limits further prevent crawling algorithms from obtaining large amounts of social network data efficiently. Because of the issues discussed above, collecting full social network data is often infeasible. Some research utilizes various network sampling methods to collect smaller sample networks of a large social network to perform analysis. Some other research obtains samples of user-generated posts/updates via non-rate-limited stream APIs. The success of the research highly depends on sample quality (e.g., the representativeness of samples to the original full dataset).

1.2.2 Data Cleaning

Social network data contain a lot of informal user-generated content, which is inevitably accompanied by noise, spam, and inconsistency. The purpose of data cleaning is to remove noisy and irrelevant data from useful information. It improves data quality and then improves the accuracy of analysis results. The typical issues that data cleaning deals with are listed as follows:

- **Noise and spam**. Social networks depend heavily on user-generated content, which spreads very fast across social networks. Therefore, social networks are susceptible to various malicious spam and hacker actions. Noisy and harmful information generated by malicious users and automated programs (e.g., web robots) not only leads to personal or business damage but also affects the quality of data analysis results. Spam detection and removal in social networks have been addressed by existing work, e.g., [88, 92].

- **Data inconsistency**. User-generated social network data contain a lot of spelling errors, abbreviations, and synonyms, which raise the issue of data inconsistency. In addition, social network data collected from multiple data sources with different data formats can also cause data inconsistency. Removing inconsistencies in datasets helps to derive complete and integrated records. There is some work dealing with matching entities in social networks [78, 75].

- **Data incompleteness**. Missing information in social network data is caused by various reasons such as no access authorization, communication failure, and imperfect data acquisition processes. Incomplete data can cause analysis to fail or produce incorrect results. Various algorithms have been designed to treat missing user profiles or links [69, 29].

1.2.3 Data Reduction

Data collected from online social networks can be very large. It becomes very time consuming for analysis algorithms to run on these data. The goals of data reduction are to represent data in a reduced form (with or without information loss) which has much smaller volume, and to make sure that analysis on the reduced data produces the same or almost the same outputs as on the original data. The classic technologies of social network data reduction include feature extraction and data compression.

- **Feature extraction**. Social network data are very rich in nature. For some particular analysis requests, thousands of relevant features can be identified, which are suspected to be notoriously redundant. Input data containing all these features are very high-dimensional and large. Feature extraction technologies transform these data into a reduced set of features that contain relevant information from the input data and can be used to perform the desired tasks to replace the full-size input data. Feature extraction techniques are usually task and data specific. For example, Lee et al. [54] construct a bag-of-words feature to represent documents for trending topic classification tasks. Sengstock et al. [85] extract some latent geographic features from social media data for spatial information-related analysis. Generally, feature extraction can reduce dataset size, enhance understanding of datasets, and improve performance of analysis.

- **Data compression**. Large social network data can also be compressed by transforming the data into a compact data structure with smaller size. Analysis can be performed on the compressed data efficiently without decompression. Various methods have been proposed to compress a large network allowing efficient neighborhood queries, one of the most essential operations of graph mining, over its compressed representation [19, 63].

1.2.4 Data Conversion

With or without awareness, users are sharing a large amount of personal information on social media such as their identities, daily activity patterns, and financial and health statuses. Gathering and releasing social network data for analysis raise genuine concerns about privacy disclosure. Therefore, social network data need to be carefully converted to remove sensitive information (e.g., personal identifiers) before being released for analysis. Preliminary data conversion methods based only on removing sensitive information are not sufficient to prevent privacy breaches. Malicious adversaries can link social network data with some external data or background knowledge to infer sensitive information. For example, Backstrom et al. [7] demonstrate that the mere knowledge about a user's neighborhood structure can reveal the identity of the user. To prevent privacy disclosure, more sophisticated data conversion approaches have been proposed [7, 10, 59, 99, 107, 110]. The main challenge for these privacy protection approaches is maintaining a balance between privacy and data utility.

1.3 Contributions

Social network data preparation is not a simple issue; rather it is a compositional issue with various challenging problems. Each problem alone needs dedicated study. In this dissertation, we focus on three crucial problems related to the data collection and data conversion steps. One is the sampling issue in social network data collection, and the other two are the privacy and utility issues respectively at the data conversion step. Figure 1.1 provides a flow chart which summarizes the social network data preparation steps and highlights the problems studied in this dissertation. Our contributions are summarized as follows.

Firstly, we study the sampling quality issue in collecting user-generated content from social networks. As mentioned in Section 1.1, social network data consist of user-generated content data and linkage data. Various sampling methods on link-

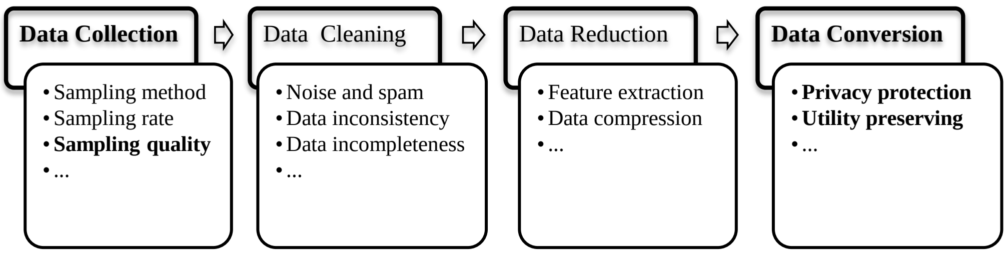


Figure 1.1: Social network data preparation.

age data have been proposed, and their advantages and bias have been thoroughly discussed [62]. However, relatively less attention has been paid to user-generated content data on this issue. Restricted by processing power and storage space, most research that analyzes user-generated content from social networks relies on samples obtained via stream APIs. However, the lack of consideration on the quality and potential bias of the samples reduces the effectiveness and validity of the analysis results. To fill this gap, in the first part of the dissertation, we perform a comparative analysis of data samples obtained from social network stream APIs to understand the representativeness of the samples to the corresponding complete data and their potential for use in various data mining tasks.

Secondly, we study sophisticated data conversion techniques to protect user privacy in social networks. We address two main desiderata of a good privacy protection scheme namely *privacy* and *utility*. In the second part of the dissertation, we discover a new type of attacks on social networks called connection fingerprint (CFP) attacks, in which attackers utilize the connection information of an anonymous user to some known public users in a social network to re-identify the user and compromise identity privacy. We formally analyze the risk of CFP attacks on real-world social networks and propose two efficient k-anonymity-based network conversion algorithms to protect social networks against CFP attacks and preserve the utility of converted networks. One algorithm is based on adding dummy vertices. It can resist powerful attackers with the connection information of a user with the public users within n hops ($n \geq 1$) and can preserve the centrality utility of public users. The other algorithm is based on edge modifications. It is only able to resist attackers with the connection information of a user with the public users within 1 hop but preserve a rich spectrum of network utility.

Finally, in the third part of the dissertation, we identify the disadvantage of existing k-anonymity-based privacy protection methods for preserving social network data utility. Most, if not all, of the existing k-anonymization algorithms convert a network to protect privacy via deleting and adding edges, and at meanwhile they control utility loss by minimizing the number of edges modified. We find that sometimes this simple utility model cannot reflect real utility changes of networks with complex structure. Therefore, existing k-anonymization algorithms designed based on this simple but not efficient utility model cannot guarantee generating social networks with high utility. To solve this problem, we propose a new utility benchmark which directly measures the changes caused by a network conversion algorithm on network community structure. We also design a general k-anonymization algorithm framework based on this new utility model. Our algorithm can significantly improve the utility of generated networks compared with existing algorithms.

1.4 Organization of the Dissertation

The work in this dissertation is an aggregation of several research papers we have published or submitted. We organize them as follows. In Chapter 2, we review existing work on social network data sampling and privacy protection. In Chapter 3, we analyze samples of user-generated content obtained via social network stream APIs and highlight their potential for use in various data mining tasks. In Chapter 4, we investigate connection fingerprint attacks on social network data privacy and propose corresponding protection algorithms. In Chapter 5, we consider the utility issue when converting a social network to protect privacy and propose a novel utility measure to enhance the utility performance of network conversion algorithms. Finally, in Chapter 6, we conclude this dissertation and discuss promising directions for future work.

Chapter 2

Related Work

In this chapter, we further elaborate on the background of our study in this dissertation with a comprehensive literature review. We first discuss prior work on social data sampling and identify the blank spot in this area of research to which our study can contribute. We then review the current development of social network privacy protection techniques and discuss the weaknesses of existing approaches which our work can improve.

2.1 Social Network Data Sampling

With the development of Internet and online media, the size of online social networks is increasing exponentially. Collecting and handling huge social network data challenge the social network analysis community. Social network data mainly consist of linkage data and content data, which can be collected separately.

2.1.1 Sampling Linkage Data

Linkage data are essentially a graph representing social network entities (e.g., users) and their relationships (e.g., friendships). In order to reduce the time cost and resource consumption of collecting a very large social graph, people usually collect a much smaller sample of the complete graph and use the sample for analysis. Various graph sampling techniques have been studied. The focus of these techniques includes how to generate small enough samples of a huge graph efficiently, and whether these imperfect samples can preserve certain graph properties or be used to perform certain analysis tasks and produce similar results as if the tasks were performed on the full graph.

There are three main types of graph sampling strategies, including vertex selection, edge selection, and sampling by exploration. Vertex selection strategies sample a subset of vertices in a graph either randomly or with probability proportional to some known properties of the vertices such as degree and PageRank values [55], and then include associated edges. Edge selection strategies first select a subset of edges by a certain principle, e.g., randomly, and then include associated vertices. We can also combine vertex selection strategies with edge selection strategies to perform sampling. For example, the random vertex-edge sampling method [50] uniformly selects a vertex and then uniformly selects an edge associated with it, and a hybrid sampling method [50] performs random edge sampling with probability p and then performs random vertex-edge sampling with probability $1 - p$. Sampling by exploration strategies start from including a set of seed vertices to the sample and then choose next-hop vertices to add to the sample from the neighbors of the current vertices in the sample. Various sampling algorithms are proposed based on different methods of choosing next-hop vertices such as depth-first/breadth-first algorithms [50], random walk-based algorithms [103], the forest fire algorithm [55], the sample edge count algorithm [62], and the expansion sampling algorithm [61].

Some prior work compares the structural properties of samples obtained using different methods with those of the original graph and discusses the sampling bias of different methods. For example, Leskovec et al. [55] study sampling methods of the three strategies and find some sampling by exploration methods (i.e., random walk and forest fire) outperform vertex selection and edge selection methods in accurately representing both the static and evolutionary patterns of the original graph, Gjoka et al. [35] find the Metropolis-Hashing random walk algorithm and a re-weighted random walk sampling method can produce approximately uniform user samples on Facebook, and Maiya et al. [62] investigate the bias of different sampling strategies and show that certain types of bias are beneficial for many applications as they "push" the sampling process towards inclusion of specific properties of interest (e.g., high-expansion or high-degree vertices).

2.1.2 Sampling User-Generated Content

The main stream of social network data sampling research focuses on sampling linkage data. However, social networks contain not only linkage data but also a large volume of user-generated content like blogs, status updates, and shared pictures and videos. Similar to linkage data, user-generated content also contains lots of valuable information. Many social network analysis tasks are developed based on this information such as opinion mining, event detection, and sentiment analysis. The volume of user-generated content increases much faster than that of pure linkage data because users in social networks continuously generate new content. An active user can generate hundreds of new posts per day. Given the huge social network user population, it is even more impractical to collect and maintain a complete record of user activities. Therefore, most research and some commercial applications that analyze user-generated content also rely on sampling. However, unlike linkage data, relatively little attention has been paid to the quality of samples, even though it fundamentally affects the effectiveness and validity of the analysis results. To fill this gap, in the first work of this dissertation, we perform an exploratory study of social media content samples and emphasize how well the samples represent the underlying complete data. We will discuss more specific prior work on this subject in Chapter 3.

2.2 Social Network Privacy Protection

The importance of privacy protection has been well recognized in data mining and data management research [3]. Prior work in this area studies privacy protection on tabular data [57, 80], transactional data [33], and survey rating data [89]. A comprehensive survey is provided in [31]. With the increasing popularity of social network analysis research, the privacy problem on social network data starts to gain a lot of attention [58].

2.2.1 Identity Privacy Protection

The identity disclosure problem is one of the mostly investigated social network privacy issues. Revealing the true identities of social network users can lead to further disclosure of other sensitive information. Therefore, a social network is usually released for research and analysis with the personal identifiers (e.g., names, social security numbers) of all users removed. We call this released social network a naively anonymized network. However, an adversary can use various background knowledge about some victim users to re-identify vertices from the naively anonymized social network. The background knowledge that the adversary utilizes is usually some structural features of the victims in the network. In other words, the adversary associates an anonymized vertex in the network with a victim user if they have the same structural features. This type of attacks is known as *structural re-identification attacks*. Backstrom et al. [7] prove that many structural features which are maliciously planted by attackers and naturally exist in social networks can be used to perform re-identification attacks. The typical structural features used in re-identification attacks include vertex degree, vertex neighborhood, subgraph, distances to hub vertices, etc [38, 59, 107]. Other than structural features, some non-structural features (e.g., community membership [97]) may also be used to re-identify vertices in social networks.

To prevent structural re-identification attacks, various network conversion methods have been proposed [10, 37, 38]. For example, the random permutation approach [37] protects privacy by randomly deleting and inserting m edges. This method is simple, but it does not provide any quantitative guarantee on privacy. Later, Bonchi et al. [13] propose to quantify the anonymity level of a randomly perturbed network based on entropy. The network generalization-based approach [38] partitions the vertices of a social network into small blocks. Then, it converts the social network into a super graph in which a super node represents a block of vertices, and a super edge between two super nodes represents the connections between vertices across the corresponding blocks. The super graph also contains statistical information in the super nodes and super edges (e.g., the number of vertices and edges covered by a super node, and the number of edges represented by a super edge). To guarantee privacy, this method requires partitions of size at least k. However, the structural uncertainty introduced by the generalization significantly degrades the utility of the social network.

The notion of k-anonymity [90] is widely adopted to prevent re-identification attacks on social networks. The general idea is to convert a social network to a k-anonymized network, in which every vertex is indistinguishable from at least $k-1$ other vertices. Thus, an attacker cannot associate a victim user with a vertex in the network with probability larger than $1/k$. Various k-anonymity schemes have been proposed as well as the corresponding k-anonymization algorithms targeting on attackers with different background knowledge. For example, Liu et al. [59] propose a k-degree anonymity scheme against attackers with degree information of some target users. It requires that for every vertex v in the converted social network, there are at least $k-1$ other vertices having the same degree as v. Zhou et al. [107] consider attackers with neighborhood information and propose a k-neighborhood anonymity scheme. It requires that for every vertex v, there are at least $k-1$ other vertices having isomorphic neighborhoods. Zou et al. [110] propose a k-automorphism scheme against attackers with unpredictable background knowledge. It modifies a network such that for each vertex, there are at least $k-1$ other structurally equivalent vertices.

Recently, Cheng et al. [18] propose a k-security model to enhance the k-anonymity notion with the ability to protect sensitive links. After applying this model to a social network, an attacker should not be able to associate any vertex to a user with probability larger than $1/k$, nor to determine two users linked by a path of certain length with probability higher than $1/k$. Zhou et al. [108] provide a comprehensive survey of these k-anonymity schemes. The k-anonymization algorithms corresponding to those schemes all share the same high-level logic. They convert a social network to satisfy their corresponding k-anonymity requirements by adding or deleting edges/vertices and require the number of edges/vertices added or deleted to be minimum in order to preserve the utility of the social network.

In this dissertation, we investigate the identity privacy protection problem in the network conversion stage. We discover a new type of re-identification attacks, namely connection fingerprint (CFP) attacks, on social networks where motivated attackers utilize the connection information of a victim to some known public users to re-identify the anonymous victim in a social network. This type of attacks is novel because attackers utilize background knowledge which combines both non-structural and structural features of the victim (i.e., the identities of some public users and the connection patterns of the victim user to the public users) to perform re-identification attacks. These attacks have not been studied by prior work. In the second work of this dissertation, we formally define CFP attacks on social networks and propose corresponding k-anonymization algorithms to protect a social network against CFP attacks. Because of the different assumption on attacks, the k-anonymization algorithms that we proposed are very different from existing ones.

Besides privacy protection, in this dissertation, we also discuss the other important component of a good privacy protection scheme: utility. Utility is important because it directly affects the quality of released social networks, which is crucial to the success of subsequent analysis tasks. Although the utility issue in releasing tabular data has been well addressed [47, 57, 80], it has not been well studied on social network data. The utility preserving approaches adopted by existing k-

anonymization algorithms on social networks are fairly naive. Most, if not all, of these algorithms try to minimize the number of edges/vertices changed to reduce utility loss. We find this utility model is *not* effective because sometimes even a small number of edges changed can cause great utility loss in a social network. In the third work of this dissertation, we design a more effective utility measure based on social community structure with a general k-anonymization algorithm framework to generate anonymized social networks with high utility.

2.2.2 Link Privacy Protection

Although our work in this dissertation mainly focuses on the identity privacy problem, link privacy is also an important privacy issue concerned while a social network is released for analysis. Protecting link privacy requires limiting the ability of attackers to infer the presence or certain properties (e.g., weights) of some sensitive edges in a social network. Random edge perturbation [37] is a commonly used technique to prevent disclosure of edge existence. It applies m random edge deletions followed by m random edge insertions to a social network. Ying et al. [102] study the impact of the edge perturbation technique on network eigenvalues and propose a new edge randomization approach to preserve spectral characteristics of networks. Xiao et al. [101] design a randomization scheme to obfuscate edge existence based on a hierarchical graph model to preserve critical statistical properties of networks. Milani Fard et al. [66] propose a neighborhood randomization approach which probabilistically randomizes the endpoints of an edge within a local neighborhood to reduce the distortion to network structure. Liu et al. [60] present edge weight perturbing techniques to protect private link weight information in social networks. The techniques designed for link privacy protection are not directly applicable to protecting identity privacy.

2.2.3 Differential Privacy

Unlike the work introduced above which releases a whole, possibly converted, network for data mining and analysis, some other work considers releasing some statistics of a network. To ensure privacy, random noise is added to the output statistics, and the notion of *differential privacy* [26] is utilized to control the level of noise injected to the output. An algorithm implementing differential privacy calculates desired statistics of an input network and injects sufficient noise to the output so that it is indistinguishable from the output on any "neighboring" network. A neighboring network is the network that is different at a single edge or vertex (with its adjacent edges) from the original network. In this way, attackers with arbitrarily high levels of background knowledge cannot infer the presence of any edge or vertex in the original network from the output, and correspondingly the algorithm is considered to guarantee *edge-differential privacy* or *vertex-differential privacy*. The success of implementing differential privacy relies on the precondition that the maximum possible change to the statistics resulted from the change of one edge or vertex should be small and bounded. Such maximum possible change is called the sensitivity of the statistics, which determines the minimum magnitude of noise that has to be added to the output. The higher the sensitivity, the more the noise added to the output.

A lot of differential privacy algorithms on social networks implement edge-differential privacy. This is because under this definition, many network statistics have low and bounded sensitivity, and thus the algorithms can generate relatively accurate outputs with low levels of noise added. For example, Mir et al. [67] propose a method to generate differentially private Kronecker graph models of an input network, based on which synthetic networks that mimic important properties of the input network can be generated. Some work proposes edge differential privacy algorithms to accurately estimate the degree distribution [36, 45], degree correlation [83, 104], subgraph counting [44], and spectral properties [94, 5] of a social network. Recently, Xiao et al. [100] propose a method to generate differentially private statistical hierarchical random graph models, which have theoretically proved lower sensitivity and can preserve essential network properties such as degree distribution, shortest path length distribution, and influential vertices.

Vertex-differential privacy is a strictly stronger privacy guarantee. However, for many network statistics, e.g., the number of edges and the frequency of a particular subgraph, it may be infeasible to design algorithms that achieve vertex-differential privacy while getting accurate results in the worst case. The problem is that vertex-differential privacy algorithms must be robust to the insertion of a new vertex in a network, but the statistics of the network can be altered dramatically by the insertion of a vertex with many adjacent edges. In other words, the sensitivity of those statistics under vertex-differential privacy is very high so that we need to add much noise to the results, which makes the results useless. Some recent work considers applying vertex-differential privacy to network data subject to certain constraints which can bound the sensitivity of some network statistics under vertex-differential privacy thus guarantee the accuracy of the results. For example, Kasiviswanathan et al. [46] provide vertex-differential privacy algorithms on bounded-degree networks for accurately releasing the edge number, subgraph counting, and degree distribution information of the networks. Chen et al. [17] design vertex-differential privacy algorithms for subgraph-counting statistics with a relaxed definition of sensitivity which measures only the maximum possible change of the statistics when a participant is removed from the dataset. This sensitivity is always bounded and is often small.

Even though differential privacy is often believed to provide a stronger privacy guarantee and thus is preferred over some other syntactic privacy notions, e.g., k-anonymity, it cannot completely replace syntactic privacy. As discussed in [23], differential privacy has many genuine problems such as calculating sensitivity, utility loss due to the inherent uncertainty, and the independence assumption of the data, and it is more suitable for privacy preserving data mining scenarios, where the statistics of the dataset to be analyzed are known prior to applying the privacy preserving process. While syntactic privacy is more suitable for privacy preserving data publishing scenarios, where no assumptions are made about the types of analysis that can be executed on the dataset. In this dissertation, we focus on privacy preserving release of social network data. Therefore, we apply the k-anonymity privacy notion but not differential privacy.

2.2.4 Social Network Privacy in Other Scenarios

In this dissertation, our focus is on the social network privacy protection problem in a scenario that data owners release a static social network to public for research and analysis. However, we are aware there is much excellent work that discusses social network privacy issues in different scenarios. For example, some work discusses privacy protection in sequential release of dynamic networks [11, 25, 91], some work discusses privacy issues in online social network settings [106, 42], and some work considers protecting the privacy of outsourced networks in untrusted servers [32]. Although these problems are very interesting, they are orthogonal to the work in this dissertation thus are not discussed in detail.

Chapter 3

Sample Representativeness Analysis for Social Network Data Collection

In this chapter, we study the sample representativeness problem of user-generated content. Our study focuses on Twitter, a typical and fast-growing microblogging service. We study Twitter because it is one of the mostly used data sources for social media analysis research and applications. Twitter is a popular data source because of its variegated uses including daily chatter, conversation, information sharing, news reporting [43] and diverse topic coverage such as arts, family and life, business, travel, sci-tech, health, education, style, world, and sports [105]. Many researchers have analyzed Twitter content and made interesting observations with real business value. For example, Sakaki et al. [82] utilize Twitter to detect earthquakes, Bakshy et al. [8] study various methods of identifying influential Twitter users, which may be useful for online marketing and targeted advertising, and Johan et al. [12] analyze Twitter user sentiment to predict the stock market.

Despite the usefulness of Twitter data, the large number of Twitter users and the even larger volume of tweets often make it impractical to collect and maintain a complete record of activity. Therefore, most research and some commercial software applications rely on samples, often relatively small samples, of Twitter data. In most cases, the sample size is based on availability and practical considerations.

Relatively little attention has been paid to how well the samples represent the underlying stream of Twitter data. In this chapter, compare the samples obtained from two of Twitter's streaming APIs with a complete Twitter dataset to gain an in-depth understanding of the nature of Twitter data samples and their potential for use in various data mining tasks.

3.1 Motivation and Problem Description

Microblogging is an increasingly popular form of lightweight communication on the Web. Twitter as a typical and quickly emerging microblogging service has attracted much attention. Millions of Twitter users around the world form a massive online information network by initiating one-way "following" relationships to others. Twitter users post brief text updates, which are commonly known as tweets, with at most 140-characters. The tweets posted by a user are immediately available to his direct followers and can be quickly disseminated through the network via retweeting. Different from traditional blog platforms, where users write long articles with low update frequency, Twitter generates a large volume of short and real-time messages daily.

One obstacle to using Twitter data is their huge size, as measured by the size of the user base, the volume of tweets, and the velocity of updates. The number of registered user profiles on Twitter reached near a billion in 2013, and, collectively, Twitter users now send over 500 million tweets every day [86]. These numbers keep growing rapidly. It is challenging for third-party researchers and developers to collect and manage such a huge amount of data.

Twitter provides API functions to facilitate third-party users to access the data (https://dev.twitter.com/docs/). There are two main types of Twitter APIs: the REST API and the stream API. The REST API supports queries to Twitter user accounts and tweets, and it usually has very strict limits on the query rate (e.g., 15 requests per 15 minutes). Although the REST API provides flexible access to Twitter data from almost every angle, the rate limits make it not suitable for collecting large amounts of Twitter data and monitoring updates. On the other hand, the stream API provides almost real-time access to Twitter's global stream of public tweets. Once the connection is built, tweet data are pushed to the client without any of the overheads incurred by pulling data through the REST API. The stream API produces near real-time samples of Twitter's public tweets. The convenience and immediacy of the stream API make it a common source of Twitter data for a variety of applications and mining tasks, for example, topic modeling [40, 76], disease outbreak surveillance [87], and popular trend detection [64]. However, prior research has not addressed the issue of how well the sample data provided by the stream API represent the original data and, if do not, towards which properties the sample data might be biased.

In this work, we focus on characterizing the sample data from the Twitter stream API, studying possible sampling bias, if any, and understanding the implications of the findings to related applications. The Twitter stream API has different access priorities. According to Twitter, the default *Spritzer* access provides a 1% sample of the complete public tweets, while *Gardenhose* access provides a larger 10% sample. However, Twitter does not reveal how the samples are generated and even does not guarantee that the sampling ratios are stable. These make it difficult to perform theoretical analysis of the sample data. Therefore, in this work, we conduct a study to experimentally analyze the properties of Twitter data samples and compare them with a baseline complete dataset.

Limited by storage capacity and the API access rate restrictions, we could not afford to collect the complete set of tweets generated by all Twitter users (over 500 million tweets per day). Instead, we use the Twitter REST API to identify a relatively small subset of Twitter users (i.e., the Singapore Twitter users) via a snowball crawling process starting with a set of manually selected seed Singapore users. And then, we obtain all the tweets generated by these Singapore Twitter users during May of 2012 via crawling the Twitter REST API and use these tweets as the *complete dataset*. Meanwhile, we gather the tweets of these users returned by the Twitter stream API in the same period with two different access priorities as the *sample datasets*. We perform a comparative analysis of the sample datasets with the complete dataset in terms of basic tweet statistics, content representativeness, user coverage, and user interactions. We find that the actual sampling ratios of the Spritzer sample and the Gardenhose sample are around 0.96% and 9.6% respectively. The sampled Twitter data represent the general user activity patterns and tweet contents of the complete dataset well even with a sampling ratio as small as 0.96%. However, the samples are biased towards active users and miss lots of user interaction information. Extending the sampling period and increasing the sampling rate both help to improve the coverage of the user base and the accuracy of the interaction based user popularity estimation.

3.2 Related Work

The huge volume of user-generated content in modern online social networks presents challenges to researchers for collecting and analyzing these data. A common practice to deal with this problem is to generate and analyze a representative sample of the complete data. There are several main issues in generating the sample: what is a good sampling strategy, what is a good sampling ratio, and does the sample have good quality? In the case of the Twitter stream API, the sample data are generated by some unknown strategies designed by Twitter with approximately fixed sampling ratios. Therefore, the focus of this work is on the unresolved question of whether the sample data generated by the Twitter stream API are good enough for various mining and analysis tasks.

A very recent work by Morstatter et al. [70] studies the same problem, however there are important differences between their work and ours. The main difference is that they use a sample dataset collected from the Twitter stream API that focuses on a particular event: the Syria conflict from December 2011 to January 2012. We analyze a dataset that is *not* event-specific to provide more general observations.

In addition, their work does not address the issue of the sampling ratio, whereas we study two different sampling ratios and discuss their effects on the quality of the data obtained. In terms of methodology, they measure the daily sampling ratio, whereas we also study the retweet ratio, and the user tweet frequency distribution to provide a more comprehensive analysis. In studying tweet contents, they analyze the correlation of the ranks of the top hashtags and compare the topic distribution of the sample data with that of the complete data. We do not compare topic distributions because we consider topic alignment across unlabeled datasets to be difficult, subjective, and unreliable. Instead, we study a rich set of terms in tweets including text words, hashtags, urls, and url domains, and discuss the similarity of the sample data using these terms to the complete data based on vocabulary coverage and frequency correlations. In order to study user relationships, their work focuses on the user retweet network, whereas we study not only retweet relationships but also mention relationships. Finally, their work analyzes the geolocation distribution of the tweets. However, since our dataset is based on Singapore Twitter users, the tweets are mainly located in Singapore, the geolocation distribution adds no new information. To sum up, our study on Twitter data samples is more general and comprehensive than the study in [70].

The rising popularity of Twitter has inspired research into its characteristics. Krishnamurthy et al. [49] perform a descriptive analysis of the Twitter user base and compare the results of two datasets crawled by different techniques. The first dataset is collected by snowball crawling of the Twitter network using the Twitter REST API. It starts with a small set of seed users and expands the user set by adding partial lists of the users being followed by the current users. The second dataset is obtained by the Twitter public timeline API, which provides continuously 20 most recent tweet updates. The users associated with these tweets are extracted. They find that the analysis results on the two datasets are similar in terms of the user class, daily activity pattern, source interface usage, and geographic distribution. Our work also analyzes Twitter datasets collected in different ways. However, it differs from the above work in three aspects. Firstly, the datasets analyzed have different properties. Our work analyzes three datasets based on the same set of Twitter users: i) a complete Singapore user tweet dataset collected by crawling the Twitter REST API, and ii) two sample datasets obtained via the Twitter stream API with different access priorities; the sample datasets are proper subsets of the complete dataset. In contrast, Krishnamurthy et al. study two datasets that may cover different sets of Twitter users. Secondly, the two studies have different purposes. In this work, our focus is not on characterizing the Twitter user base, but on characterizing the Twitter stream API and understanding how well the data collected from the stream API represents the complete Twitter data space. Finally, because of the different study objectives, we analyze different aspects of the datasets, not only including users, but also tweets and user interactions.

Kwak et al. [52] conduct an exploratory analysis of the entire Twittersphere to study the topological characteristics of the Twitter network and information diffusion on it. Their results show a remarkable deviation from known characteristics of human social networks. They find that the Twitter network has a non-power-law degree distribution, short effective diameter, and low reciprocity, which establish Twitter's role of a new medium of information sharing. This study collects the entire Twitter network snapshot in its early stage (i.e., 2009). With the rapid growth of the Twitter population, it becomes more and more difficult to handle the whole Twitter network, not to mention tracking its frequent information update. Therefore, much research has been performed on incomplete Twitter data. Java et al. [43] analyze a Twitter subset with 76,000 users and 1 million tweets and categorize the users based on their intentions on Twitter. Their dataset is collected by periodically retrieving the most recent public tweets using an old version of the Twitter stream API that is no longer supported by Twitter. Naaman et al. [72] study Twitter user activity based on a small set of sampled non-organizational users and classify them as "Meformers" and "Informers" according to whether they like to post tweets that are self-related or general informational. Zhao et al. [105] characterize Twitter with topic modeling based on tweets collected from the Twitter stream API. They classify tweets into different topic categories and study the size distribution of these categories. Huberman et al. [41] study the user activity and interactions in Twitter and reveal that the use of Twitter is driven by a hidden network of connections underlying the "declared" friend and follower relationships. The dataset they use consists of over 300,000 Twitter users and their tweets, however the method of collection is not described. These studies use Twitter datasets collected in several different ways, but none of them discusses the strengths and limitations of the data collection methods used and the representativeness of their datasets.

Besides Twitter, several other popular social networks have been studied. [16] and [34] study user-generated content on YouTube [16, 34]. Kumar et al. [51] analyze the structural properties of the Flickr and Yahoo!360 networks including path lengths, changes over time, and component structure. Mislove et al. [68] verify the power-law, small-world, and scale-free properties of many popular online social networks including Flickr, YouTube, LiveJournal, and Orkut. Benevenuto et al. [9] characterize the behavior of a set of 37,000 collected users in online social networks such as Orkut, MySpace, Hi5, and LinkedIn. None of these studies investigates the relationships between their analysis results and the data collection methods. Ahn et al. [6] compare the topological characteristics and growth patterns of three large-scale online social networks: Cyworld, MySpace, and Orkut. They evaluate the validity of the snowball sampling method, which they use to crawl the networks. Their results reveal that with a sampling ratio above a certain threshold, snowball sampling captures the scaling property of the vertex degree distribution correctly, but it cannot estimate other metrics such as the clustering coefficient distribution and degree correlation. Our work is different from this work because the sample datasets that we study are not obtained by snowball sampling of the Twitter network, but by an unknown sampling method developed by Twitter on the public tweet stream.

3.3 Data Collection

In order to study sampling bias, we need the complete Twitter dataset to serve as a baseline, with which the sample datasets can be compared. However, collecting the complete Twitter stream is not practical for our study because of its expensive cost. Instead of considering the full set of more than 1 billion Twitter users, we focus on the complete set of Singapore Twitter users, which is a smaller group. We used all the tweets posted by the Singapore Twitter users within a one-month period as the complete dataset. We also gathered all the tweets generated by these Singapore users that appeared in the Spritzer and Gardenhose streams during the same timespan to create two sample datasets.

The complete dataset was collected with the help of the social network mining research group of Singapore Management University [1]. In order to locate the Singapore Twitter users, a set of 58 popular Singapore Twitter users were manually selected as seeds. Initially, the user set only contained these seed users. The user set was then expanded by exploring the follower and friend lists of the users in the set. A follower or friend of a current user was added to the user set if either he specified his location to be "Singapore" or he followed at least three of the known Singapore users. In this way, a set of 151,041 Singapore Twitter users in 2012 was identified, which covered the majority of the Singapore Twitter users. After the set of users was constructed, the Twitter REST API was invoked to crawl the tweets generated by these users for a one-month period beginning on May 1, 2012 and ending on May 31, 2012. The collected tweets formed the complete dataset referred to as $Complete$.

We collected two sample datasets at the same period via the Twitter stream API using the Spritzer and Gardenhose access priorities respectively. The Spritzer and Gardenhose streams output samples of the entire public tweet stream with different sampling ratios. According to Twitter, Spritzer provides an approximately 1% sample of the complete public tweets, while Gardenhose generates a larger sample with a sampling ratio of around 10%. Twitter does not provide any description of

[1]https://sites.google.com/site/socnetmine/

Table 3.1: Description of datasets.

Datasets	$Complete$	$Sample_{Spritzer}$	$Sample_{Gardenhose}$
API used for collection	REST	Stream (Spritzer)	Stream (Gardenhose)
Timespan	May 2012		
Num. of users	151,041		
Num. of Tweets	13,468,661	128,647	1,297,304

the algorithms that generate the samples nor does it guarantee the sampling ratios to be stable. From the sampled tweets, we extracted the subsets that were posted by the identified Singapore users. In this way, two samples of the complete dataset were obtained, referred to as $Sample_{Spritzer}$ and $Sample_{Gardenhose}$ respectively. Table 3.1 provides some basic information of the datasets.

3.4 Analysis of Results

In this section, we perform a detailed comparative analysis of the collected sample and complete datasets. Specifically, we compare them in terms of tweet statistics, content representativeness, user coverage, and user interactions. Through the comparison, we try to understand the nature of the sample datasets, for which properties the sample datasets are representative of the complete dataset, and for which properties the sample datasets are not representative. We also discuss the implications of our findings for certain mining tasks.

3.4.1 Tweet Statistics

We first study the sampling ratio and basic tweet statistics in this section. We perform analysis on the datasets collected over the one-month period and also present results on a daily basis.

We begin the analysis by examining the actual sampling ratios of the two sample datasets from the Twitter stream API and present the average daily sampling ratios and standard deviations in Table 3.2. As shown in Table 3.2a, the Singapore users generate around a half million tweets a day on average. The Spritzer and Garden-

Table 3.2: Average daily sampling ratios.

(a) Daily sampling ratios for tweets.

Daily statistic	$Complete$	$Sample_{Gardenhose}$		$Sample_{Spritzer}$	
	tweet#	tweet#	sampling ratio	tweet#	sampling ratio
Daily avg.	481,024	46,332	9.62%	4,634	0.96%
Std. dev.	67,446	6,637	0.15%	664	0.014%

(b) Daily sampling ratios for users.

Daily statistic	$Complete$	$Sample_{Gardenhose}$		$Sample_{Spritzer}$	
	user#	user#	sampling ratio	user#	sampling ratio
Daily avg.	35,316	15,769	44.55%	3,625	10.22%
Std. dev.	2,407	1,601	1.88%	484	0.85%

Table 3.3: Daily tweets and retweets ratios.

Daily statistic	$Complete$		$Sample_{Gardenhose}$		$Sample_{Spritzer}$	
	tweet%	retweet%	tweet%	retweet%	tweet%	retweet%
Daily avg.	84.41%	15.59%	84.24%	15.76%	84.21%	15.79%
Std. dev.	0.56%	0.56%	0.54%	0.54%	0.76%	0.76%

hose samples return around 0.96% and 9.6% of them respectively. The actual tweet sampling ratios both are slightly lower than what Twitter announced (i.e., 1% and 10%). Table 3.2b shows the sampling ratios for users each day. We find on average there are around 35,000 Singapore users who generate tweets in each day, and the Spritzer and Gardenhose samples capture around 10% and 45% of them respectively. The sampling ratio for users is much higher than it is for tweets, which is not surprising; each tweet appears just once in the complete dataset, whereas a user may appear many times, thus increasing the likelihood that he will appear in a sample.

Next, we study whether the sample datasets preserve the general tweeting patterns of the Twitter users. Table 3.3 lists the average proportions of original tweets and retweets generated by the Twitter users each day. As observed from the table, among all the tweets published daily, about 85% are original tweets, and 15% are retweets. The same ratio between original tweets and retweets is captured by both sample datasets. Figure 3.1 further illustrates the average hourly tweet counts of the three datasets for all the 31 studied days. We observe that the Singapore users tend to be more active at the nighttime. The tweeting frequency increases rapidly

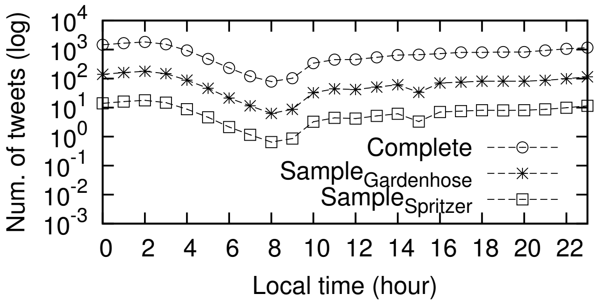
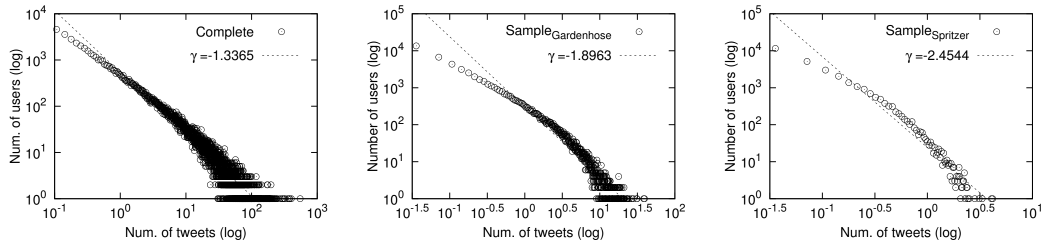


Figure 3.1: Average hourly tweet counts of Singapore Twitter users.



(a) $Complete$ $SE_\gamma = 0.008$, $R^2 = 0.91$

(b) $Sample_{Gardenhose}$ $SE_\gamma = 0.024$, $R^2 = 0.93$

(c) $Sample_{Spritzer}$ $SE_\gamma = 0.072$, $R^2 = 0.94$

Figure 3.2: Average daily tweeting frequency distributions of Singapore Twitter users. γ is estimated power-law exponent. SE_γ is the standard error of γ. R^2 is the square error of the power-law fitting.

after 17:00 and peaks at 22:00. Then it drops quickly through midnight and hits the bottom at 4:00. Thereafter, as a new day starts, the users gradually regain activity, and the tweeting frequency rises slowly through the day. The sample datasets both reflect the same hourly tweeting frequency pattern of the users. The results above indicate that both the small Spritzer sample and the larger Gardenhose sample obtained via the Twitter stream API reflect the general user tweeting patterns of the complete dataset accurately.

In addition, we analyze tweet patterns based on individual users. We plot the distribution of user average daily tweeting frequency in Figure 3.2. The average daily tweeting frequency distribution of the Singapore users approximates a power-law distribution with an exponent of -1.3365. However, the user average daily tweeting frequency distributions captured by the Spritzer and Gardenhose samples fit the power-law distributions with different exponents of -2.4544 and -1.8963 respectively. Therefore, the sample data preserve the scaling pattern of the user tweeting

Table 3.4: Symbols for a vocabulary.

Symbol	Description
V	The vocabulary for all the text words/hashtags/urls/url domains appearing in the set of tweets of the studied timespan (e.g., one day or one month).
$\lvert V\rvert$	The size of the vocabulary V; it is the number of terms that exist in the vocabulary.
t	A term in a vocabulary representing a text word/hashtag/url/url domain.
$t.f^V$, $\overline{f^V}$	$t.f^V$ is the frequency of t in vocabulary V; it is the number of times that t appears in the tweet set on which the vocabulary is built. $\overline{f^V}$ is the average frequency of all the terms in V.
$t.r^V$, $\overline{r^V}$	$t.r^V$ is the rank of t in vocabulary V; it is the rank of t in the vocabulary based on its frequency. $\overline{r^V}$ is the average value of all the term ranks in V.

frequency distribution (i.e., power-law), but tend to overestimate the proportion of users with low tweeting frequency, and the overestimation is more serious in the sample with a smaller sampling ratio, i.e., Spritzer.

3.4.2 Content Representativeness

Twitter data are also widely used for performing mining tasks such as event detection, sentiment analysis, content summarization, and topic modeling. As many of these tasks are built upon analyzing tweet contents, it is important to understand if the tweet contents in the sample datasets from the Twitter stream API accurately represent those in the complete dataset.

For each dataset, we extracted the vocabularies of four common types of text representation: text terms, hashtags, urls, and url domains. We performed lightweight preprocessing of the text terms by eliminating stopwords [2], punctuation, and non-English terms. For each dataset and each method of representation (e.g., Spritzer urls), we record the frequency of each vocabulary item and its rank each day and for the entire one-month timespan, as described in Table 3.4. We analyze the correspondence of the vocabularies of the complete dataset and the sample datasets using four metrics as described in Table 3.5 and display the results in Table 3.6.

We measure how well a vocabulary of a sample dataset covers the corresponding

[2] We use a stopword dictionary with 429 distinct terms (http://www.lextek.com/manuals/onix/stopwords1.html).

Table 3.5: Comparison metrics for vocabularies.

Metric	Description
S_{size}	The size ratio of a vocabulary of the sample tweet set (V_S) and the corresponding vocabulary of the complete tweet set (V_C). $S_{size} = \frac{\lvert V_S\rvert}{\lvert V_C\rvert}$
S_{ctf}	The collection term frequency (ctf) ratio of a vocabulary of the sample tweet set (V_S) and the corresponding vocabulary of the complete tweet set (V_C). $S_{ctf} = \frac{\sum_{t\in V_S} t.f^{V_C}}{\sum_{t\in V_C} t.f^{V_C}}$
S_{pcc}	The Pearson product-moment correlation coefficient of the term frequency values of a vocabulary of the sample tweet set (V_S) and the corresponding vocabulary of the complete tweet set (V_C). $S_{pcc} = \frac{\sum_{t\in V_S\cap V_C}(t.f^{V_S}-\overline{f^{V_S}})(t.f^{V_C}-\overline{f^{V_C}})}{\sqrt{\sum_{t\in V_S\cap V_C}(t.f^{V_S}-\overline{f^{V_S}})^2}\sqrt{\sum_{t\in V_S\cap V_C}(t.f^{V_C}-\overline{f^{V_C}})^2}}$
S_{scc}	The Spearman's rank correlation coefficient of the term rank values of a vocabulary of the sample tweet set (V_S) and the corresponding vocabulary of the complete tweet set (V_C). $S_{scc} = \frac{\sum_{t\in V_S\cap V_C}(t.r^{V_S}-\overline{r^{V_S}})(t.r^{V_C}-\overline{r^{V_C}})}{\sqrt{\sum_{t\in V_S\cap V_C}(t.r^{V_S}-\overline{r^{V_S}})^2}\sqrt{\sum_{t\in V_S\cap V_C}(t.r^{V_C}-\overline{r^{V_C}})^2}}$

vocabulary of the complete dataset using two metrics: the *size ratio* and the *collection term frequency (ctf) ratio*. Each metric provides a different perspective on how well the sample vocabulary covers the complete vocabulary.

The *size ratio* metric calculates the proportion of the unique terms in a vocabulary of the complete dataset that are captured by a sample dataset. As observed from Table 3.6a, the Spritzer sample only covers around 6% of the text vocabulary, 2.5% of the hashtag vocabulary, 1% of the url vocabulary, and 3.7% of the url domain vocabulary in each day. The size ratios of the vocabularies of the Gardenhose sample are much higher due to the higher sampling ratio (i.e., the Gardenhose sample covers around 26% of the text vocabulary, 16% of the hashtag vocabulary, 9% of the url vocabulary, and 18% of the url domain vocabulary in each day). In addition, we find that the size ratio of the url vocabulary almost equals the tweet sampling ratio, while the size ratios of text terms, hashtags, and url domains are much higher than the tweet sampling ratio. This result is easily explained. Many of the url terms occur only once in the complete dataset, thus the odds of seeing them in a sample depend strongly on the sample size. In contrast, many individual text terms, hashtags, and url domains have higher occurrence frequency, thus samples tend to cover more of

Table 3.6: Content representativeness based on four types of vocabularies daily and for all the studied days (i.e., one month).

(a) Size ratio (S_{size})

Daily statistic	$Sample_{Gardenhose}$				$Sample_{Spritzer}$			
	text	hashtag	url	url domain	text	hashtag	url	url domain
Daily avg.	0.257	0.160	0.090	0.182	0.064	0.025	0.010	0.037
Std. dev.	0.021	0.019	0.013	0.023	0.002	0.002	0.001	0.003
All days	0.237	0.185	0.092	0.184	0.062	0.032	0.010	0.034

(b) ctf ratio (S_{ctf})

Daily statistic	$Sample_{Gardenhose}$				$Sample_{Spritzer}$			
	text	hashtag	url	url domain	text	hashtag	url	url domain
Daily avg.	0.915	0.622	0.121	0.915	0.750	0.371	0.034	0.837
Std. dev.	0.013	0.044	0.018	0.013	0.022	0.038	0.006	0.020
All days	0.977	0.791	0.144	0.960	0.939	0.603	0.054	0.926

(c) Pearson product-moment correlation coefficient (S_{pcc})

Daily statistic	$Sample_{Gardenhose}$				$Sample_{Spritzer}$			
	text	hashtag	url	url domain	text	hashtag	url	url domain
Daily avg.	0.997	0.9715	0.911	0.987	0.975	0.856	0.655	0.974
Std. dev.	0.002	0.021	0.045	0.005	0.005	0.040	0.195	0.013
All days	0.100	0.993	0.990	0.988	0.999	0.979	0.973	0.985

(d) Spearman's rank correlation coefficient (S_{scc})

Daily statistic	$Sample_{Gardenhose}$				$Sample_{Spritzer}$			
	text	hashtag	url	url domain	text	hashtag	url	url domain
Daily avg.	0.812	0.641	0.433	0.736	0.705	0.552	0.268	0.754
Std. dev.	0.009	0.016	0.022	0.018	0.013	0.044	0.050	0.036
All days	0.817	0.691	0.442	0.706	0.811	0.623	0.266	0.692

these vocabularies.

The results on the size ratio metric indicate that the sample datasets cover only small proportions of the vocabularies for different types of representation of the complete dataset. However, the size ratio metric treats every term equally; it is skewed by the many terms that appear just a few times in the dataset. Based on our observation, the infrequent terms are more likely to be typographical errors and/or user-created words, which may be less important for studying Twitter contents.

In order to distinguish the frequent terms from the infrequent ones, we adopt another vocabulary coverage metric namely collection term frequency (ctf) ratio [14]. This metric also calculates the proportion of the terms in the complete vocabulary that are covered by a sample vocabulary. But, it weights each term with its frequency of occurrence to give more credits to the frequent terms. The frequent and moderately-frequent terms convey the most information about the contents of a dataset [15]. Therefore, the ctf ratio is a more appropriate metric for measuring the quality of a sampled vocabulary. The closer the ctf ratio is to 1, the more a sample contains the terms that are frequent in the complete dataset [3]. The results of the ctf ratio are displayed in Table 3.6b. Generally, we find that the ctf ratio for every vocabulary is much higher than the corresponding size ratio, and the Gardenhose sample has higher ctf ratios than the Spritzer sample because of the higher sampling ratio.

Closer inspection reveals that different types of text representation behave differently.

- For the daily text term vocabularies, the ctf ratios are significantly high (e.g., they are around 0.75 for the Spritzer sample, and they exceed 0.9 for the Gardenhose sample). Therefore, even the small sample datasets contain the text terms that account for most of the word occurrences in the complete dataset.

- The ctf ratios for the Spritzer sample are about 0.37 for the daily hashtag vocabularies, whereas the Gardenhose ctf ratios are about 0.62. Sampling over a one-month timespan improves the Spritzer coverage to 0.60 and the Gardenhose coverage to 0.80. These results suggest that one might want to be cautious about drawing conclusions from daily variations in hashtag occurrences in the Spritzer stream, and even conclusions based on the Gardenhose stream ($10\times$ larger) will miss significant amounts of hashtag activity. Observations based on a one-month timespan are more reliable, but will necessarily miss some hashtag activity.

- The ctf ratios for urls are very low for both sample datasets, due primarily to the fact that many of the urls only occur once in the dataset. One may interpret

[3]Note that stopwords are not included in this comparison. If stopwords were included, they would dominate the weighting, and all methods would have ctf ratios close to 1.

this result as indicating that many tweeted urls are unimportant and thus safely ignored; or, it may mean that tweeting frequency is a less reliable method of determining the importance of a tweeted url. In any of the cases, the Spritzer and Gardenhose streams provide only very limited information about the urls in the underlying complete stream.

- Instead of using individual urls, some researchers may be interested in only the domains that produce urls, for example, to identify information sources that are popular with Twitter users. We find that the Spritzer sample has very high ctf ratios (i.e., around 0.83) for url domains, and the ctf ratios are even higher for the Gardenhose sample (i.e., about 0.92). Therefore, even though the sample datasets do not provide enough information for studying the popularity of individual urls, they preserve the important url domains very well.

In addition to analyzing the daily vocabularies, we also analyze the cumulative vocabularies of the one-month period. We find that extending the sampling period does not improve raw vocabulary coverage as the size ratios are not significantly improved with the cumulative vocabularies. This observation is consistent with Heaps' Law, which predicts the continued growth of a vocabulary as more texts are observed [39]. However, the long sampling period helps to improve the coverage of the frequent terms, as indicated by the increase in the ctf ratios for the cumulative vocabularies.

The size ratio and ctf ratio metrics only evaluate the proportions of (frequent) terms that are captured by the sample datasets. They do not evaluate whether the frequency information of the captured terms is well preserved by the sample datasets. In other words, they do not evaluate whether the term frequency information obtained from the sample datasets is correlated with the actual term frequency in the complete dataset. For mining tasks such as event detection, tweet content summarization, and sentiment analysis, term frequency information is crucial. Therefore, in the following analysis, we use another two metrics to study the quality of the term frequency information in the sample datasets: the Pearson product-moment correlation coefficient (PCC) and Spearman's rank correlation coefficient (SCC).

The Pearson product-moment correlation coefficient (PCC) measures the linear dependency of the *term frequency* values in a sample vocabulary and those in the complete vocabulary. Its value is in the range of $[-1, 1]$. The value is close to 1 if the term frequency values in the sample dataset and the complete dataset are strongly correlated, the value is 0 when the term frequency values are uncorrelated, and the value is -1 when the term frequency values are inversely correlated. Spearman's rank correlation coefficient (SCC) measures the linear dependency of the frequency-based *term rankings* of a sample vocabulary and the complete vocabulary. To calculate the SCC score, the terms in a vocabulary are ranked in descending order of their frequency in the dataset. Rank ties are handled by assigning a rank that is equal to the average of their positions in the ranked list. For example, if the top two terms both have the highest frequency in the ranked list, i.e., there is a tie between position 1 and position 2, the ranks assigned to these two terms are both $1.5 = \frac{1+2}{2}$. The SCC score is calculated based on the term rankings with a function similar to that of the PCC metric (see Table 3.5), and it has the same value range.

The results of these two metrics are shown in Table 3.6c and Table 3.6d. We find that in most of the cases, the PCC values are above 0.8 and 0.9 for the Spritzer and Gardenhose vocabularies respectively. Thus, the term frequency values of the sample datasets are linearly correlated with those of the complete dataset. In other words, the sample datasets accurately estimate the relative frequency of the terms in every vocabulary. The results of the SCC metric are not as good as those of the PCC metric, but still are reasonably high, except for the url vocabularies. Therefore, the sample datasets predict the term rankings of text terms, hashtags, and url domains to a certain extent. The degradation of the SCC performance is mainly caused by the ties in the term ranking. The url vocabularies have the most rank ties since many of the urls only appear once in the datasets, thus have the worst SCC performance. The improvement of the PCC and SCC scores by extending the sampling period is not very obvious.

Table 3.7: Features derived for sentiment classification.

Feature category	Features
Overall scores (6)	Sum of positive and negative scores for Adjectives. Sum of positive and negative scores for Adverbs. Sum of positive and negative scores for Verbs.
Score ratios to number of terms (6)	Ratios of positive and negative scores to total number of terms for each part of speech.
Positive to negative score ratios (3)	Positive to negative scores ratio for each part of speech.
Negation (1)	Percentage of negated terms in a tweet

Sentiment Analysis

To demonstrate the usefulness of the sample data for analyzing Twitter content, we perform a sentiment classification task on the sample datasets and the complete dataset and then compare the results. Sentiment classification is an opinion mining activity concerned with determining what is the overall sentiment orientation of the opinions contained within a given document (e.g., tweet). The sentiment orientation can be classified as positive or negative. We implement the binary classifier described by Ohana and Tierney to analyze the sentiment orientation of tweets [74]. We extract sentiment features of tweets using SentiWordNet and then train an SVM classifier to assign sentiment labels to the tweets in each dataset. SentiWordNet is a lexical database for opinion mining. Given a term and its part-of-speech tag, SentiWordNet returns three sentiment scores ranging from 0 to 1: positivity, negativity, and objectivity, each indicating the term's sentiment bias. The sum of the three scores equals 1. In the experiment, we use the GATE Twitter part-of-speech tagger (https://gate.ac.uk/wiki/twitter-postagger.html) to perform part-of-speech analysis on tweets then calculate SentiWordNet scores for terms found. We derive a total of 16 sentiment features based on the scores as described in Table 3.7.

We traine a linear SVM classifier with a set of 1224 manually classified tweets from a separate dataset. The training data consist of 570 positive tweets and 654 negative tweets. We apply the trained classifier to predict the sentiment orientation of the tweets in our datasets.

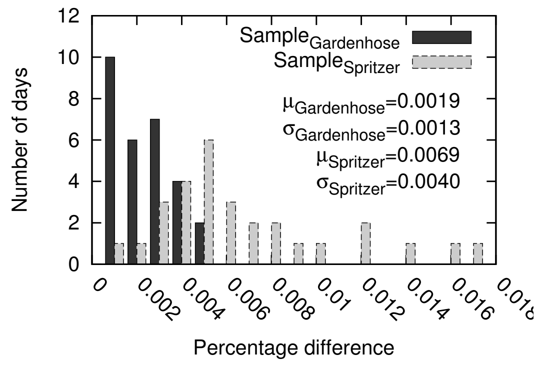
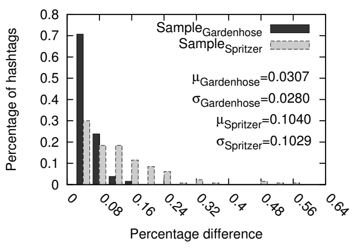


Figure 3.3: Percentage difference of positive (or negative) tweets daily. X-axis is binned difference. Y-axis is the count of days in each bin. μ is average difference of 30 days, σ is standard deviation.

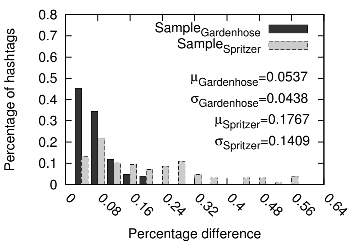
Firstly, we analyze Twitter's daily overall sentiment polarity by counting the percentages of tweets with positive sentiment and negative sentiment respectively in each day. In order to compare the sample datasets with the complete dataset, we calculate the absolute difference of the positive tweet percentages of a sample dataset and the complete dataset in each day [4]. Figure 3.3 presents the percentage difference distributions of the Spritzer and Gardenhose samples over the studied 31 days. We observe that for all the 31 days, the percentage differences of both datasets are fairly small (i.e., less than 1.8%), which indicates that both sample datasets can reflect Twitter's daily sentiment orientation very accurately. It is not surprising that the larger Gardenhose sample shows higher accuracy of predicting Twitter's daily sentiment polarity. Its daily percentage differences are all smaller than 0.6%.

Besides the overall sentiment orientation, we also study Twitter's sentiment orientation towards hashtags. We group the hashtags that are captured by both sample datasets based on their popularity in the complete dataset. We categorize the hashtags which were used by more than 1000 tweets as very popular, the hashtags which were used by less than 1000 but more than 500 tweets as moderately popular, and the hashtags which were used by less than 500 tweets as less popular. We ignore the hashtags that were used by less than 100 tweets because of their lack of popularity. We infer Twitter's sentiment orientation to a hashtag also by the percentages

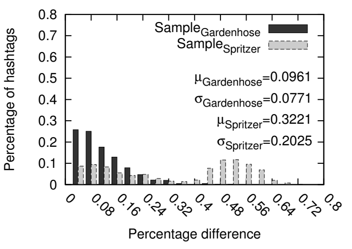
[4]The absolute difference of the negative tweet percentages is the same as that of the positive tweet percentages.



(a) Very popular hashtags



(b) Moderately popular hashtags



(c) Less popular hashtags

Figure 3.4: Percentage difference of positive (or negative) tweets in different hashtag groups with different popularity levels. X-axis is binned difference. Y-axis is the percentage of hashtags in each popularity group. μ is the average difference of all the hashtags in each popularity group, σ is standard deviation

of the positive and negative tweets containing this hashtag. We use the percentage difference to evaluate the error of a sample dataset for predicting Twitter's sentiment polarity to each hashtag. Figure 3.4 shows the percentage difference distributions of the three hashtag groups. We find that the Gardenhose sample relatively accurately reflects Twitter's sentiment orientation to the hashtags in the very popular and moderately popular groups. In these two groups, the Gardenhose sample captures sentiment orientation for most of the hashtags (i.e., more than 90% and 80% of the hashtags respectively) with the percentage difference less than 8%. The performance of the Gardenhose sample degrades greatly for the less popular hashtags. It can only guarantee small percentage differences (e.g., less than 8%) for around 50% of the hashtags. Not surprisingly, the performance of the smaller Spritzer sample is not as good as that of the Gardenhose sample. It can only achieve less than 10% percentage differences for around 70% of the very popular hashtags. For most of those not so popular hashtags, the percentage difference is not small.

To sum up, in this subsection, we find that both the Spritzer and Gardenhose samples can be used to estimate Twitter's overall sentiment orientation. However, for individual hashtags, the Gardenhose sample can be used to estimate Twitter's sentiment orientation for very popular and moderately popular hashtags, while the Spritzer sample may be only suitable for estimating Twitter's sentiment for very popular hashtags.

Table 3.8: Average number of tweets daily of captured and missed users.

Daily statistics	$Sample_{Gardenhose}$		$Sample_{Spritzer}$	
	captured users	missed users	captured users	missed users
Daily avg.	23.99	4.20	48.45	10.13
Std. dev.	2.56	0.08	2.49	0.55

3.4.3 User Coverage

According to the analysis in Section 3.4.1, the daily user sampling ratios of the Spritzer dataset and the Gardenhose dataset are about 10% and 45% respectively. Thus, more than half of the users in the complete dataset who tweet each day are not captured by the sample datasets. In the following analysis, we compare the properties of the users captured by the sample datasets with those of the users that are missed. Here, the missed users refer to the users who generate tweets and are included in the complete dataset, but their tweeting behavior is not captured by the sample datasets.

Firstly, we calculate the average numbers of the tweets generated daily by the captured and missed users respectively and report the results in Table 3.8. The results show that the captured users tend to publish more tweets than the missed users. To be more specific, the users captured by the Spritzer sample publish more than 48 tweets daily on average, while the missed users generate only around 10 tweets daily. The average daily tweeting frequency of the captured users and the missed users in the Gardenhose sample is around 24 and 4 respectively. These results imply that samples generated by the Twitter stream API are biased towards active users who tweet frequently every day, and they may lose the voice of less active users. The bias is more significant with sample datasets having smaller sampling ratios (e.g., $Sample_{Spritzer}$).

Although the user samples are biased towards active users, we expect that this bias can be remedied by extending the sampling period. In other words, with the extension of the sampling period, the chance of discovering inactive users will increase. To verify how fast the user samples grow as the sampling period increases, we first identified the 37,124 Singapore users who published tweets on the first day

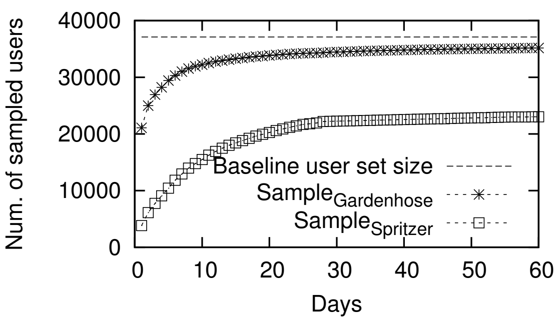


Figure 3.5: Total number of users observed over time.

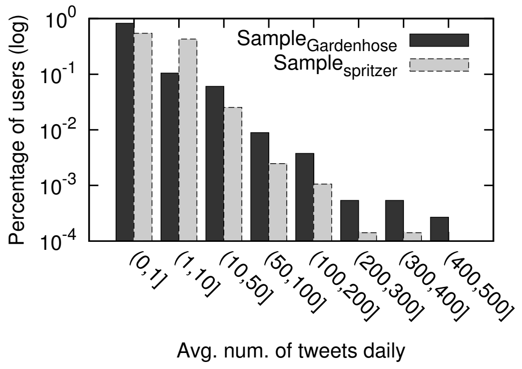


Figure 3.6: Average daily tweeting frequency distribution of missed users.

of the complete dataset as the baseline user set. Then, we monitored the Twitter stream API to see when these users were observed. Once a user's tweets were spotted, we added that user to the sample user set. We performed this monitoring using both the Spritzer stream and the Gardenhose stream for 60 days and maintained two sample user sets respectively. Figure 3.5 plots the size changes of these sample user sets over the 60 days.

The sample user set from the Gardenhose stream grew quickly for the initial 10 days, eventually reaching about 87% of the baseline user set. After that, it converged slowly to the baseline user set. After 60 days of monitoring, 35,174 Singapore users were discovered, which covered about 95% of the baseline user set. However, the set of users found in the Spritzer stream converged much more slowly because of the low sampling ratio. After 60 days, it only had captured 23,036 users, which covered 62% of the baseline user set. We conclude that extending the sampling period helps to improve user coverage, and a period of 10 days is enough for discovering most of the baseline users (i.e., more than 85% of them) using the Gardenhose stream.

After 60 days of monitoring, there were still some users that had not been seen in the Spritzer and Gardenhose samples. We examined the daily tweeting frequency distributions of these missed users. The results are displayed in Figure 3.6. More than 90% of the missed users in both datasets tweeted infrequently, i.e., generated no more than 10 tweets a day. Therefore, it is not surprising that these users did not appear in any sample.

The daily tweeting frequency distribution of the users not found in the Gardenhose sample is significantly skewed towards the extremely low frequency users (i.e., users who tweet less than once a day on average) compared with the Spritzer sample. This result confirms that the higher sampling rate increases the chance of discovering low-activity users, and only extremely inactive users are missed.

However, there is a group of users who are very active (i.e., post more than 100 tweets daily), but do not appear in any sample. Although this group is small -- less than 1% of the missed users -- it is perhaps surprising that these users do not appear in either of our samples. We manually checked the profiles and the tweets of these users. Most of these users are organizational users or marketers who periodically tweet urls linking to external websites or product promotions. We suspect that these users' tweets are intentionally excluded from the sample streams, perhaps because Twitter has identified them as robots, spammers, or other undesirable information producers. This group of missing users may not be a problem for most research, because it is a tiny group, and because the information that they provide may not represent "real" user content. However, they might be important for research that studies robot and spammer behavior.

3.4.4 User Interactions

Another type of valuable information embedded in Twitter data is the interactions between users. There are two main types of interactions between Twitter users: mention interactions and retweet interactions. A Twitter user *mentions* another user by inserting "@username" into the body of his tweet. Mentions are usually used to

Table 3.9: Proportions of the reciprocal and directed user mention interactions extracted from the complete and sample datasets daily and for all the studied days (i.e., one month).

Daily statistics	$Complete$		$Sample_{Gardenhose}$		$Sample_{Spritzer}$	
	reciprocal	directed	reciprocal	directed	reciprocal	directed
Daily avg.	11.44%	88.56%	7.18%	92.82%	4.10%	95.90%
Std. dev.	0.34%	0.34%	0.51%	0.51%	0.47%	0.47%
All days	10.67%	89.33%	9.25%	90.75%	5.73%	94.27%

signify quotes from other users' posts or to send direct messages to the users that are mentioned. A Twitter user *retweets* another user's tweet by clicking the "Retweet" button under that tweet. A retweet is a re-posting of someone else's tweet.

Mention and retweet interactions can be directed or reciprocal. Usually, a directed interaction indicates an "informational relationship" between users because the information only flows one-way from a user to another, while a reciprocal interaction indicates a "friendship relationship" because there is communication between users. In the rest of this Chapter, we use the terms "interaction" and "relationship" interchangeably.

Mention and retweet relationships can be obtained from tweet meta-data. They are commonly used for many tasks such as understanding user roles in Twitter, identifying influencers, and modeling information diffusion. In this section, we extract the mention and retweet relationships among the Singapore Twitter users from the complete and sample datasets respectively and study the representativeness of the sample datasets on these relationships. We first analyze the mention relationships. The same analysis is performed on the retweet relationships as well.

We first examine whether the sample datasets represent the proportions of the reciprocal and directed mention relationships in the complete dataset. Table 3.9 provides the results. Among all the Singapore Twitter users, about 11.4% of the mention relationships captured daily in the complete dataset are reciprocal, and 88.5% of them are directed. However, in the Spritzer sample, 4% of the mention relationships captured are reciprocal, and 96% of them are directed; while in the Gardenhose sample, the proportions of the reciprocal and directed relationships are around 7%

Table 3.10: Recall of user mention interactions daily and for all the studied days (i.e., one month).

(a) Recall on all users

Daily statistic	$Sample_{Gardenhose}$			$Sample_{Spritzer}$		
	$Recall_{Dir.}$	$Recall_{Rec.}$	$Recall_{All}$	$Recall_{Dir.}$	$Recall_{Rec.}$	$Recall_{All}$
Daily avg.	0.181	0.109	0.173	0.021	0.007	0.020
Std. dev.	0.013	0.021	0.014	0.001	0.001	0.001
All days	0.244	0.208	0.240	0.040	0.020	0.038

(b) Recall on captured users

Daily statistic	$Sample_{Gardenhose}$			$Sample_{Spritzer}$		
	$Recall_{Dir.}$	$Recall_{Rec.}$	$Recall_{All}$	$Recall_{Dir.}$	$Recall_{Rec.}$	$Recall_{All}$
Daily avg.	0.213	0.143	0.206	0.064	0.053	0.063
Std. dev.	0.014	0.026	0.016	0.004	0.008	0.004
All days	0.283	0.263	0.281	0.095	0.086	0.095

and 93% respectively. These observations tell us that the sample datasets tend to underestimate the amount of reciprocal relationships, and the underestimation of the Spritzer sample is more serious than that of the Gardenhose sample. Again, we find that extending the sampling period improves the estimation of the proportions of reciprocal and directed relationships. The Gardenhose sample for a one-month period has similar proportions of these relationships to those of the complete dataset.

Next, we study how many of the relationships in the complete dataset are captured by the sample datasets. We calculate the recall of reciprocal, directed, and all relationships and list the results in Table 3.10a. As observed from the table, the Spritzer dataset and the Gardenhose dataset recover around 2% and 17% of all the daily interactions among all the Singapore Twitter users respectively. Given the tweet sampling ratios of the two datasets (i.e., less than 1% and 10%), these recall values are reasonable. However, many applications that utilize user interaction information need a complete view of user relationships, for example, analyzing user network properties and studying network-based information diffusion. For these applications, the sample datasets do not provide sufficient information.

Even though the recall of mention relationships is increased by extending the sampling period to one month, it is still far from complete, i.e., in the best case,

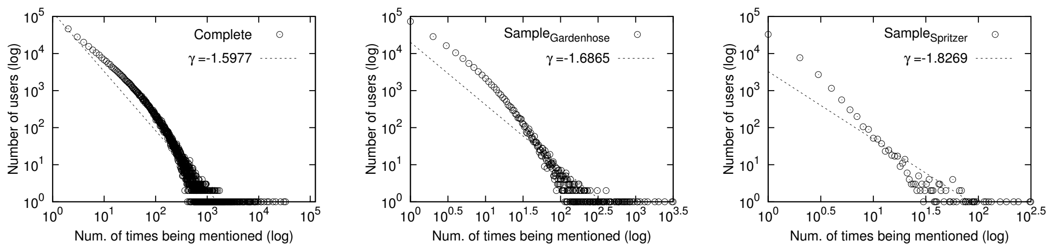
Table 3.11: Estimation of user popularity based on frequency of being mentioned using the data daily and for all the studied days (i.e., one month).

Daily statistic	$Sample_{Gardenhose}$			$Sample_{Spritzer}$		
	S_{ctf}	S_{PCC}	S_{SCC}	S_{ctf}	S_{PCC}	S_{SCC}
Daily avg.	0.544	0.938	0.547	0.168	0.808	0.321
Std. dev.	0.048	0.025	0.021	0.014	0.040	0.018
All days	0.906	0.994	0.825	0.576	0.973	0.553

the Gardenhose sample captures only 24% of all the interactions of the Singapore Twitter users based on the one-month period of sampling. To get a nearly complete view of user relationships, much longer sampling time may be needed. However, the relationships among Twitter users are relatively dynamic. Relationship information extracted from historical data may lose effectiveness. We also notice that the recall of reciprocal relationships is generally smaller than that of directed relationships, which indicates that reciprocal relationships are harder to capture from sample data.

The analysis described in Section 3.4.3 found that a sample dataset only covers a proportion of the active users every day. To make our analysis fairer, we also calculate the recall of the interactions between these captured users; the results are displayed in Table 3.10b. As observed, if we only focus on the captured users, the recall values of all the daily interactions of the Spritzer dataset and the Gardenhose dataset increase to around 6% and 20% respectively. By extending the sampling period to one month, the two datasets capture around 9.5% and 28% of all the mention interactions between the captured users respectively. Since much of the interaction information between users is missing from the sample data, researchers cannot construct a user mention network from the sample data that has similar properties to the user network constructed from the complete dataset.

In the final set of analysis, we study the intensity of users being mentioned. This piece of information is important for studying user roles and locating influencers in Twitter. Usually, popular users that are mentioned many times by many users are more important than less frequently mentioned users in the Twitter space. To perform the analysis, we extracted the frequency of users being mentioned in the complete and the sample datasets respectively and produced rankings of users based on



(a) $Complete$
$SE_\gamma = 0.022$, $R^2 = 0.82$

(b) $Sample_{Gardenhose}$
$SE_\gamma = 0.050$, $R^2 = 0.81$

(c) $Sample_{Spritzer}$
$SE_\gamma = 0.104$, $R^2 = 0.81$

Figure 3.7: Distribution of the frequency of users being mentioned based on the tweets of the one month period. γ is estimated power-law exponent. SE_γ is the standard error of γ. R^2 is the square error of the power-law fitting.

the frequency. Then the ctf ratio, PCC score, and SCC score were calculated based on the extracted information to evaluate the effectiveness of the sample datasets for preserving user popularity information. The results are presented in Table 3.11.

One day of the 1% Spritzer sample contains tweets that mention the users who are responsible for about 16% of the mentions in a day of the complete dataset (Table 3.11). One day of the 10% Gardenhose sample contains tweets that mention the users who are responsible for about 55% of the mentions in the complete dataset. The PCC scores based on the daily samples are very high, i.e., 0.8 and 0.93 for the Spritzer and Gardenhose samples respectively, which indicates that the mention frequency of users in the sample datasets is strongly correlated with the mention frequency in the complete dataset. If a user is mentioned in the sample data, the frequency information is relatively reliable. However, many of the users mentioned frequently in the complete dataset are not observed in a one-day sample.

Extending the sampling period to one month greatly improves the results. The ctf ratios are greatly improved, especially for the Gardenhose sample (i.e., over 0.9). We also find that the PCC score and the SCC score based on the extended sampling period increase to 0.99 and 0.82 respectively for the Gardenhose sample. Therefore, by extending the sampling period, the Gardenhose sample successfully captures most of the popular users and accurately predicts the users' relative popularity in terms of the frequency of being mentioned. Even though the ctf ratio of the Spritzer sample is also improved to 0.57 by extending the sampling period, it is still at risk of missing many important users. Therefore, it is preferable to use the Gardenhose sample with an extended sampling period for studying user popularity.

We also analyze the user popularity distribution based on the frequency of being mentioned using the data of the one-month period. The results are displayed in Figure 3.7. We find that the user popularity distribution in the complete dataset approximates a power-law distribution with an exponent of -1.5977. The user popularity distributions of the sample datasets preserve the power-law property but with smaller exponents, thus they overestimate the proportion of less popular users. Again, we find that the user popularity distribution of the Gardenhose sample is more similar to the original distribution compared with the Spritzer sample.

We performed exactly the same analysis on retweet interactions. Despite variations in numbers and details, the results are similar to the results based on mention interactions. All the observations made with mention interactions in this section are applicable to retweet interactions.

3.5 Summary

In this chapter, we provide a descriptive study of Twitter data samples obtained from the Twitter stream API with two different access priorities (i.e., Spritzer and Gardenhose). These two data streams are data sources for a variety of research and commercial applications. By comparing the sample data with the corresponding complete data in different aspects, we explore the nature of the sample data, their bias, and how well they represent the complete data stream. Our results provide insights about the sample data obtained from the Twitter stream API and provide incentives for people to use or not to use them for their research.

We find that the Twitter streams with the Spritzer and Gardenhose access priorities provide samples of the entire public tweets with actual sampling ratios of around 0.96% and 9.6% respectively. The sample datasets truthfully reflect the daily and hourly activity patterns of the Twitter users in the complete dataset. Moreover, the sample datasets capture the approximate power-law property of the user tweeting frequency distribution in the complete dataset, but with smaller exponents. In other words, the sample datasets preserve the same scaling property of the user tweeting frequency distribution as the complete dataset, but tend to overestimate the proportion of low-frequency users. The overestimation is more serious when the sampling ratio is small. These observations indicate that the sample datasets, even with a very small sampling ratio such as the Spritzer samples (i.e., 0.96%), are good for studying Twitter user activity patterns in general. However, researchers should be careful about the overestimation of low-frequency users when trying to analyze users based on their activity levels and, if possible, use the larger sample (i.e., Gardenhose) to reduce the estimation error.

Even with a very small sampling ratio (i.e., 0.96%), the sample datasets are able to capture certain important tweet contents, e.g., text terms and url domains, and preserve the relative importance (i.e., the frequency of appearance) of the content terms. Our work demonstrates that the sample datasets are useful for many tasks that analyze tweet contents, such as event detection, sentiment analysis, and tweet summarization. For some other types of contents, e.g., hashtags, the small Spritzer sample is not adequate to preserve accurate information, and the larger Gardenhose sample is needed. However, for some content entities like urls, of which the appearances in tweets are temporal (e.g., each url only appears once or a few times), the importance of the terms is not reinforced by recurrence. In this case, the sample datasets may only capture small portions of the data, and may miss lots of crucial information.

The sample datasets are biased towards active users, as one might expect. We find that extending the sampling period or increasing the sampling rate both help to improve user coverage. By carefully examining the users that are difficult to sample, we find that the majority of them are extremely inactive with very low average daily tweeting frequency (e.g., post less than 1 tweet a day). A small proportion of the users that are not sampled are highly active, but probably spammers that Twitter

Table 3.12: Summary of conclusions.

Tasks	Usefulness of sample data (Yes/No)	Comments
General tweet statistics	Yes	Spritzer samples are enough.
Event detection	Yes	If hashtags are used, Gardenhose samples are preferable.
Sentiment analysis	Yes	
Tweet summarization	Yes	
User network analysis	No	
Information diffusion	No	
User popularity	Yes	Gardenhose samples with a long sampling period are preferable.

deliberately excludes. For the tasks of studying the general Twitter user base, these two types of users are the least interesting because the extremely low-activity users hardly contribute anything to Twitter, and the spammers most likely only generate noise information. Therefore, the Twitter stream API can be used for collecting representative Twitter users.

Finally, we find that owing to the low sampling ratios on tweets, the sample datasets cover only small proportions of the user interactions, i.e., mentions and retweets, embedded in tweets. For example, in the best case, the Gardenhose sample captures around 28% of the mention relationships between the captured users in the one-month sample of data. Therefore, the sample datasets cannot provide a complete view of the user interaction network, thus are unsuitable for the tasks that study user network properties and information diffusion. However, for the tasks that study users' popularity based on their frequency of being mentioned or retweeted, Gardenhose samples for a one-month period provide relatively accurate information.

In general, the Twitter data samples obtained via the Twitter stream API preserve enough information for the research or applications conducted based on general tweet or content statistics, such as user activity pattern characterization, event detection, sentiment analysis, and tweet summarization. They may also be useful for analyzing the Twitter user base and user popularity. However, they cannot provide a complete view of the user interaction network for tasks such as user network analysis and information diffusion modeling. Table 3.12 summarizes our findings.

Although our results provide new information about the quality of Twitter data streams, they are limited by the scope of the datasets, which were collected based on a set of Singapore Twitter users. Even though our work focuses on general patterns and metrics that are not population specific, analysis of a different user population might lead to different conclusions. We believe that our observations about the Spritzer and Gardenhose samples will apply to other populations, however this remains an open question.

We notice that the "Spritzer" and "Gardenhose" samples have many characteristics which are similar to what people could expect from random samples e.g., user activity pattern, retweet ratio, and tweeting frequency distribution. However, we could not conclude that these samples are truly random samples because we have observed that the public tweets from certain active users are excluded from the samples presumably due to their suspected spam behavior. We think it is an interesting problem for the future works to compare the Twitter "Spritzer" and "Gardenhose" samples with some truly random samples.

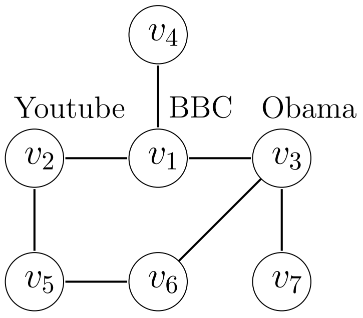
Chapter 4

K-Anonymity based Social Network Conversion against Connection Fingerprint Attacks

In this chapter, we study the privacy protection issue on preparing social network data for analysis. We discuss a new type of attacks on social network data, which has not been discovered by prior work, namely *connection fingerprint (CFP) attacks*. In CFP attacks, malicious adversaries utilize the connection information of a victim (anonymous) user to some known public users to re-identify the user. We first formally define CFP attacks and then analyze the privacy risk of these attacks on real-world social networks. In addition, we adopt the k-anonymity scheme to convert a social network against CFP attacks. We propose two k-anonymization algorithms with their own respective advantages and demonstrate their ability to generate k-anonymized networks with good utility.

4.1 Motivation

Propelled by the emergence and rapid rise of social media within the past decade, people are now driving interactions online, thus contributing to the vast amount of user-generated content on online social media platforms. On the one hand, both industry and academia are very happy with the blooms of social network data as they provide lots of new opportunities for business and research. For example, many firms are investing time and resources in mining social network data so it helps public relations, marketing, and sales engage more relevantly with consumers. On the other hand, the increasing sophistication of information technology with its capacity to collect, analyze, and disseminate social network data is posing significant threats to social network users' privacy. A common practice is to release a social network after removing the real identities of vertices for tasks such as mining and analysis. However, prior research has demonstrated even though all the personal identifiers (e.g., names, social security numbers) were removed, an attacker could re-identify a vertex based on the unique topological structure around it in a social network. The topological structure includes vertex degree [59], neighborhood [107], subgraph, distances to hub vertices, and so on [38]. This type of attacks is called *structural re-identification attacks*.



Vertex ID	Connection fingerprint
v_4	{BBC}$_1$ {YouTube, Obama}$_2$
v_5	{YouTube}$_1$ {BBC, Obama}$_2$
v_6	{Obama}$_1$ {YouTube, BBC}$_2$
v_7	{Obama}$_1$ {BBC}$_2$

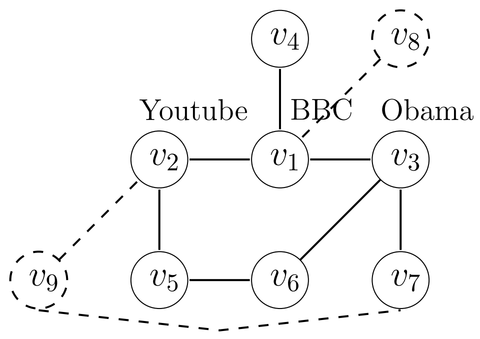
Figure 4.1: Example of a released social network with public users and the connection fingerprint information of private users within 2 hops.

Prior work on structural re-identification attacks assumes that the identities of all social network users are sensitive and should be removed from the released dataset. However, in reality, not all the identities of social network users are sensitive. Sina Weibo, a popular Chinese microblogging social network, hosts around 176,700 government accounts (e.g., Hong Kong, Mainland China, Taiwan, and Macau), 110,000 media accounts (e.g., PR Newswire, NBA, People's Daily, and Xinhuanet), and millions of celebrity accounts (e.g., Boris Johnson, Kevin Rudd, David Cameron, and Kai-Fu Lee). All these users' identities are public, and they in total account for over

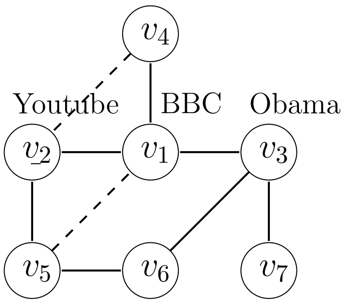
1% of the overall half billion registered user accounts [93, 96]. Facebook has over 50 million public pages for brands, communities, companies, etc. with 10 or more likes from the one billion registered profiles by 2013 [24]. These users seek publicity, and they want to influence the social networks. In other words, they do not consider their identities as privacy, and they prefer their information to be publicly recognizable.

In this work, we define the group of social network users whose identities are not sensitive as *public users* to be distinguished from *private users*. Releasing the identities of public users with social network data can benefit both research and the users themselves. For example, these data can be used for studying the social influence of government organizations, simulating information propagation through media, helping corporates make smart targeted advertising plans, and so on. However, it potentially leads to privacy disclosure of other private users. Take a simple social network, modeled as an unweighted and undirected graph, in Figure 4.1 as an example. It has seven vertices representing three public users (i.e., v_1, v_2, v_3) and four private users (i.e., v_4, v_5, v_6, v_7). Assume that the identities of the private users are removed when this social network is released for certain data analysis tasks, and an attacker knows that in this social network a user Tom connects with a public user BBC within 1 hop[1]. As there is only one anonymous vertex v_4 who directly connects to this public user BBC, the attacker identifies v_4 as Tom without any doubt. In this attack, the attacker infers the identity of an anonymous private user via the public users that he connects to. In other words, the group of public users that a private user connects to may leak the identity of the private user. For ease of presentation, we name the identities of the public users that a user u_i connects to as u_i's *connection fingerprint (CFP)*. For instance, $\{$*BBC*$\}$ is the 1st-hop CFP of vertex v_4, denoted as $CFP_1(v_4)$ =$\{$*BBC*$\}$, and $\{$*Youtube, Obama*$\}$ is the 2nd-hop CFP of vertex v_4, denoted as $CFP_2(v_4)$ =$\{$*Youtube, Obama*$\}$. The formal definition of n^{th}-hop CFP will be presented later in Section 4.2. The table in Figure 4.1 lists the

[1]The number of hops between two vertices in an unweighted graph is the number of edges on the shortest path between them.



(a) Anonymized social network by adding dummy vertices.



(b) Anonymized social network by modifying edges.

Figure 4.2: Examples of 2-anonymity based on connection fingerprint information.

1st-hop and 2nd-hop CFPs of the private users in the example social network.

In this work, we study re-identification attacks based on CFP information namely *connection fingerprint (CFP) attacks*, which have not been well studied by prior work. First, we formally define the CFP re-identification problem. Next, we analyze the risk of CFP attacks with real-world network data. Our study demonstrates that even a small proportion of public users (e.g., 1%) can harm the privacy of a considerable amount of private users. Then, we propose to adopt the widely accepted k-anonymity strategy to resist CFP attacks. Informally, k-anonymity requires that for every private user u_i in a social network, there are at least $k-1$ other private users who share the same CFP as u_i, which guarantees that an attacker cannot re-identify any private users based *only* on CFP information with probability larger than $\frac{1}{k}$.

We propose two different methods to anonymize social networks against CFP attacks, and we recognize their respective advantages and limitations. The first method is based on adding dummy vertices. For example, in Figure 4.2, two dummy (private) vertices v_8 and v_9 are added to the network as well as some dummy edges to connect them to the network. The dummy vertices and edges are signified by dashed circles and lines. The social network satisfies 2-anonymity ($k = 2$) in terms of CFPs (within 2 hops). This is because in the modified social network, for every private vertex, there is at least one other private vertex having the same 1st-hop and 2nd-hop CFPs. As will be explained later in Section 4.4, this approach can be easily extended to prevent any CFP attacks within n hops, where $1 \leq n \leq m$, and m is the maximum number of hops between two vertices in the network. Although it is general, this algorithm may limit the utility of released social networks because of the large number of dummy vertices added. The second method is based on modifying (i.e., adding and/or deleting) edges. Take Figure 4.2b as an example. We insert an edge between v_2 and v_4 and another edge between v_1 and v_5. The social network then satisfies 2-anonymity based on 1^{st}-hop CFPs since $CFP_1(v_4) = CFP_1(v_5) = \{BBC, Youtube\}$, and $CFP_1(v_6) = CFP_1(v_7) = \{Obama\}$. This approach works well on preventing CFP attacks of 1 hop and preserves network utility better than the first method. However, it is hard to extend this approach for n-hop CFP attacks ($n > 1$). This will be further explained in Section 4.5.

Finally, we conduct comprehensive experiments to evaluate the performance of our algorithms. We find that with realistic assumptions of the number of public users and the privacy parameter k, our algorithms are able to generate anonymized networks with good utility efficiently.

4.2 Preliminaries and Problem Formulation

We model a social network as an undirected and unweighted graph $G(V, E)$, where V is a set of vertices representing user entities in the social network, and E is a set of edges representing social connections between users (e.g., friendships, contacts, and collaborations). The notation $e(v_i, v_j) \in E$ represents an edge between two vertices v_i and v_j. We use the notation $|S|$ to represent the cardinality of a set S. For ease of presentation, we use "graph" and "social network" interchangeably in the following discussion.

Social network data are released for research and analysis with possible modifications for protecting privacy, e.g., removing user identities. We assume there are some public users whose identities are not sensitive in a social network G and hence are retained in the released social network G^* as specified in Definition 1. The real identity of a public vertex v_i is denoted by $ID(v_i)$. Except the public users, the rest of the users in G are private users, whose identities are sensitive and hence are excluded from the released social network G^*.

Definition 1 (Released Social Network) *The released version $G^*(V^*, E^*)$ of a social network $G(V, E)$ is obtained by removing all the identity information of* private *vertices of G, with possible modifications (i.e., adding dummy vertices and edges, modifying edges). $G^*(V^*, E^*)$ is used in place of G for research and analysis, with $V^*_{pub} \subset V^*$ (and $\subset V$) referring to the public vertices.*

We study the privacy breach in G^* caused by releasing public user identities. First, we specify the hop distance $h(v_i, v_j)$ between two vertices v_i and v_j as the number of edges on the shortest path between them [2]. Given a hop distance value n, we give the formal definitions of the n^{th}-hop connection fingerprint $CFP_n(v_j)$ and n-range connection fingerprint $CFP(v_j, n)$ for a private vertex v_j in Definition 2 and Definition 3, respectively. Accordingly, we introduce *connection fingerprint (CFP) re-identification attacks*, as stated in Definition 4. This kind of attacks is initiated by attackers who are aware of the connection fingerprint of a target user.

Definition 2 (n^{th}-Hop Connection Fingerprint) *The n^{th}-hop connection fingerprint $CFP_n(v_j)$ of a private vertex v_j in a social network $G(V, E)$ consists of the group of public vertices whose hop distances to v_j are exactly n, i.e., $CFP_n(v_j) = \{ID(v_i)|v_i \in V^*_{pub} \wedge h(v_i, v_j) = n\}$.*

Definition 3 (n-Range Connection Fingerprint) *The n-range connection fingerprint of a private vertex v_j, denoted by $CFP(v_j, n)$, is formed by v_j's x^{th}-hop connection fingerprints, where $1 \leq x \leq n$, i.e., $CFP(v_j, n) = \cup_{x\in[1,n]}\{CFP_x(v_j)\}$. We consider the n-range CFPs of two vertices v_i and v_j to be the same (i.e., $CFP(v_i, n) = CFP(v_j, n)$) iff $\forall x \in [1, n]$, $CFP_x(v_i) = CFP_x(v_j)$.*

Take the social network in Figure 4.1 as an example. The 1st-hop CFP and 2nd-hop CFP of vertex v_4 are $\{$*BBC*$\}$ and $\{$*Youtube, Obama*$\}$ respectively, and v_4's 2-range connection fingerprint $CFP(v_4, 2)$= $\{\{$*BBC*$\}, \{$*Youtube, Obama*$\}\}$.

[2]In an unweighted graph, $h(v_i, v_j)$ equals the shortest path distance between v_i and v_j.

Definition 4 (n-Range Connec. Fingerprint Re-identification) *Given a social network $G(V, E)$, its released version $G^*(V^*, E^*)$, and a target entity $v_j \in V - V^*_{pub}$ with n-range connection fingerprint $CFP(v_j, n)$, let the group of the private vertices in G^* sharing the same n-range connection fingerprint as v_j be v_j's candidates, denoted by $V_{CFP(v_j,n)}$, i.e., $V_{CFP(v_j,n)} = \{v \in V^* - V^*_{pub} \,|CFP(v, n) = CFP(v_j, n)\}$. If $|V_{CFP(v_j,n)}| \ll |V^* - V^*_{pub}|$, v_j has high probability to be re-identified.*

We adopt the k-anonymity notion as formally stated in Definition 5, to prevent n-range CFP attacks. Given a k-anonymized network, assume that an attacker knows there is a target private user u in the social network having the connection fingerprint $CFP(u, n)$. She performs the re-identification attack and finds a group of x candidate vertices having the same n-range CFP as u. According to the k-anonymity notion, $x \geq k$. Without further information, the attacker cannot differentiate these x vertices, and the probability for her to identify u among these x vertices is $\frac{1}{x} \leq \frac{1}{k}$. Therefore, the k-anonymity notion prevents re-identification attacks by providing an upper bound (i.e., $\frac{1}{k}$) for re-identification probability. The larger the k, the smaller the upper bound, and hence the lower the risk of identity disclosure.

Definition 5 (K-Anonymity) *A released social network $G^*(V^*, E^*)$ satisfies k-anonymity with respect to n-range connection fingerprint information, iff for each private vertex $v_j \in V^* - V^*_{pub}$, there are at least $k - 1$ other private vertices $v' \in V^* - V^*_{pub} - \{v_j\}$ with $CFP(v', n) = CFP(v, n)$.*

Our objective is to design an algorithm to efficiently convert a social network so that it achieves k-anonymity against n-range CFP re-identification attacks. The algorithm should make as few changes to the social network as possible to preserve the utility of the network so that the anonymized network can be used for research.

Table 4.1: Properties of networks

Datasets	polblogs	facebook	grqc	route-view
$\lvert V\rvert$	1222	4039	4158	6474
$\lvert E\rvert$	16714	88234	13422	12572
Density	0.0224	0.0108	0.0015	0.0006
Diameter	8	8	17	9
Avg. clustering coef.	0.226	0.519	0.629	0.009

4.3 Privacy Risk Analysis

In this section, we use experiments to evaluate the vulnerability of social networks under n-range CFP re-identification attacks. We perform the evaluation on four real-world networks described as follows.

The **polblogs** network was crawled from the US political blogosphere in 2005. The vertices are blogs of a set of US politicians, and an edge between two blogs represents there is a hyperlink from one blog to the other [1]. The **facebook** network was collected from the survey participants using a Facebook app. The vertices are Facebook users, and an edge between two users represents the established friendship between them [65]. The **grqc** network is a collaboration network collected from e-print arXiv. It covers scientific collaborations between the authors of the papers submitted to the General Relativity and Quantum Cosmology category. The vertices are authors, and the edges represent coauthorship [56]. **Route-view** is a network of autonomous systems of the Internet connected with each other. The vertices are autonomous systems, and the edges denote communication [56]. All the networks are represented by undirected and unweighted graphs with no isolated vertices. Table 4.1 presents some basic statistics of the networks.

The four real-world networks do not contain public user identities. In other words, all the vertices in the networks are anonymous. In order to evaluate the risk of CFP attacks, we select a set of vertices in each network and assume that their identities are public. Then, based on these public vertices, we generate the CFPs of the remaining private vertices and evaluate how many private vertices can potentially be re-identified via the CFP information. We adopt three different strategies to select public vertices.

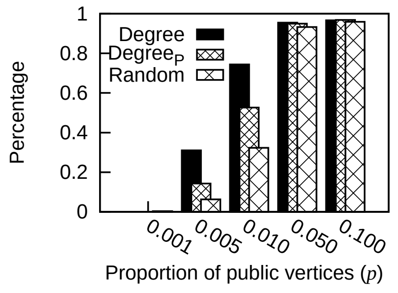
The **degree based strategy**, referred to as $Degree$, chooses the vertices with the highest numbers of adjacent edges (i.e., degree) as public vertices. This is based on the assumption that public users in a social network such as celebrities and news media users normally have much larger degree compared with ordinary users because of their popularity.

The **degree probability strategy**, referred to as $Degree_p$, also picks public vertices based on degree information. The probability that it picks a vertex is proportional to the vertex's degree. In this way, it does not only pick vertices with the highest degree values, but also some vertices with lower degree.

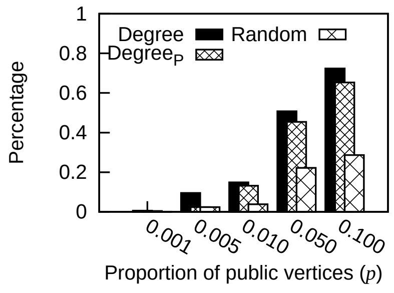
The **random strategy**, referred to as $Random$, simply chooses public vertices uniformly randomly from a network.

Based on the public vertex set of a network, we calculate the n-range CFPs of the private vertices and then evaluate the risk of the private vertices under re-identification attacks. We group the private vertices based on their n-range CFPs. The private vertices with the same n-range CFP are grouped together as an *equivalent group*, denoted as EG. An attacker cannot differentiate the vertices in an equivalent group based only on CFPs. Therefore, given a vertex in an equivalent group EG_i, the probability that an attacker can re-identify it is at most $1/|EG_i|$, referred to as its *maximum re-identification probability*. Given a privacy parameter k, k-anonymity requires that the maximum re-identification probability of every private vertex should be no larger than $1/k$. The vertices with maximum re-identification probability larger than $1/k$ are considered to be potentially at risk of re-identification attacks. We calculate the percentage of private vertices in each network which are potentially at risk of CFP attacks based on varying proportions of public vertices selected, denoted as p, hop numbers n, and privacy thresholds k.

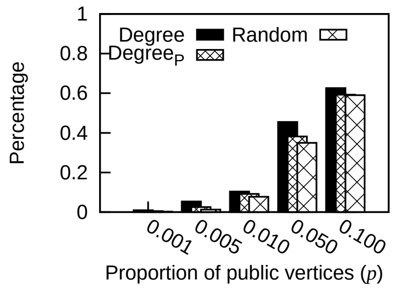
Figure 4.3 shows the percentage of vulnerable private vertices in each network with varying proportions of public vertices p. We observe from the figure that generally the percentage of vulnerable vertices increases significantly with the proportion



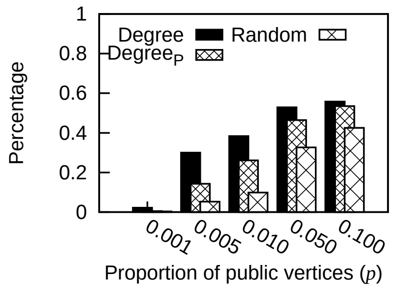
(a) polblogs



(b) facebook



(c) grqc



(d) route-views

Figure 4.3: Percentage of vulnerable vertices in each social network varying with proportion of public vertices ($k = 5, h = 2$).

of public vertices. It means that the more the public users in a network, the more vulnerable the network is to CFP attacks. Moreover, we observe from the figure that the $Degree$ strategy causes the most number of vulnerable private vertices, followed by the $Degree_p$ strategy. Randomly chosen public vertices have a relatively low impact on re-identifying private vertices. Therefore, the public vertices with the highest degree values threaten the identity privacy of private users the most. Across the four networks evaluated, we find there are considerable portions of private vertices that are vulnerable to CFP attacks. For example, with a moderate proportion of public vertices (e.g., 0.01) chosen based on the $Degree$ strategy, there are around 70% of the private vertices that may be at risk of CFP attacks in the **polblogs** network, and this percentage is between 10% and 40% in the other three networks. If the proportion of public vertices increases to 0.05, the percentage of vulnerable vertices rises to more than 90% in the **polblogs** network and to around 50% in the other three networks [3].

[3]The differences of the vulnerable vertex percentages in the **polblogs** network and the other three networks are caused by different network density. The other three networks are sparser than the

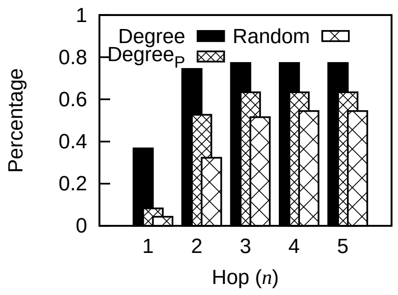


Figure 4.4: Percentage of vulnerable vertices increases with hop number in the **polblogs** network ($p = 0.01, k = 5$).

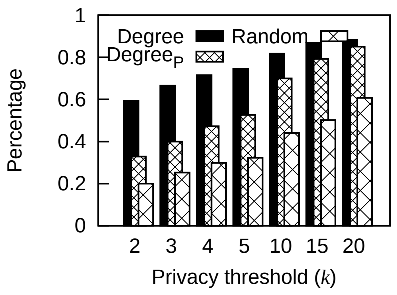


Figure 4.5: Percentage of vulnerable vertices increases with privacy threshold in the **polblogs** network ($p = 0.01, n = 2$).

In Figure 4.4 and Figure 4.5, we fix the proportion of public vertices at 0.01, and show how the percentage of potentially risky vertices changes with the hop number n and privacy threshold k increasing, respectively. We only present the results on the **polblogs** network, and the results on the other three networks have similar trends. It is not surprising that the larger the hop number n or the threshold k, the more vertices are vulnerable to re-identification attacks. When the hop number n is larger than 1, the percentage of vulnerable vertices increases greatly.

To sum up, in this section, we verify the practicability of CFP-based re-identification attacks. The results on real-world networks demonstrate that the re-identification risk is real and could be very serious sometimes.

4.4 Dummy Vertex Addition based Anonymization for n-Range CFP Attacks

In this section, we design a k-anonymization algorithm to protect a social network against n-range CFP attacks based on adding dummy vertices. The main idea, although simple, is effective. It first groups the private vertices in a social network G into equivalent groups such that vertices in the same group have the same n-range CFP. Then, for every equivalent group EG_i with less than k vertices, it inserts

polblogs network. They may contain a large set of private vertices that do not connect to any public vertices thus are not vulnerable to CFP attacks.

Algorithm 1: K-anonymity algorithm for n-range CFP attacks.

```
Input: A social network $G(V, E)$, privacy parameter $k$, hop parameter $n$
Output: A $k$-anonymized social network $G^*(V^*, E^*)$
1  $G^*(V^*, E^*) = G(V, E)$;
2  $\forall v \in V^* - V^*_{pub}$, calculate $CFP(v, n)$;
3  Group vertices in $V^* - V^*_{pub}$ into equivalent groups $EG$s;
4  foreach $EG_i$ do
5      if $|EG_i| < k$ then
6          Insert $k - |EG_i| + \delta$ ($\delta > 0$) dummy vertices into $EG_i$;
7          foreach dummy vertex $v_d$ inserted do
8              Pick a non-dummy vertex $v_i$ randomly in $EG_i$;
9              For every $e(v_i, v_j) \in E^*$, insert $e(v_d, v_j)$ to $E^*$;
10             foreach $e(v_d, v_j)$ inserted, $v_j$ is a private vertex do
11                 Find the equivalent group $EG_j$ containing $v_j$;
12                 Randomly pick a vertex $v_t \in EG_j$;
13                 Remove $e(v_d, v_j)$, and insert $e(v_d, v_t)$;
14 return $G^*$;
```

$k - |EG_i| + \delta$ dummy vertices and connects each dummy vertex carefully with the vertices in G so that it has the same n-range CFP as other vertices in EG_i. Here, δ is a very small positive integer [4]. In this way, all the equivalent groups contain at least k vertices and hence the social network satisfies k-anonymity.

Algorithm 1 provides the pseudo-code of the k-anonymization algorithm for n-range CFP attacks. In the algorithm, an important step is to insert a dummy vertex v_d into the social network (line 7-13). To achieve k-anonymity, we must make sure that v_d has the same n-range CFP as other vertices in the same equivalent group EG_i, and the n-range CFPs of other private vertices in the social network are not affected. In order to achieve this, for a dummy vertex v_d to be inserted into an equivalent group EG_i, we first pick a real vertex v_i from EG_i and then copy all the edges of v_i to v_d (line 8-9). In other words, v_d becomes an exact duplicate of v_i since it connects to exactly the same set of neighbor vertices as v_i. Obviously, $CFP(v_d, n) = CFP(v_i, n)$, and the n-range CFPs of all the other vertices remain unchanged. This clearly satisfies our requirements for inserting dummy vertices into a social network. However, the existence of the dummy vertex may be easily recognized because it connects to exactly the same set of neighbors as another vertex.

[4]We choose δ randomly in the range $[0, k)$.

To make the dummy vertex less recognizable, for each inserted edge $e(v_d, v_j)$ where v_j is a private vertex, we randomly perform an edge switching operation. To be more specific, we replace $e(v_d, v_j)$ with an edge $e(v_d, v_t)$ with v_j and v_t in the same equivalent group. As stated in Lemma 1, this edge switching operation does not change the n-range CFPs of any private vertices including v_d.

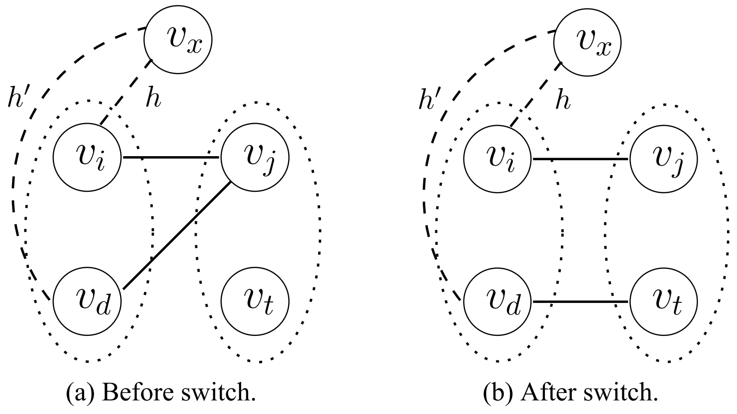


Figure 4.6: v_i and v_d are in the same equivalent group having the same n-range CFP. Switching edge $e(v_d, v_j)$ to edge $e(v_d, v_t)$ where v_j and v_t are in the same equivalent group does not change the n-range CFPs of any vertices.

Lemma 1 *Assume a dummy vertex v_d is a duplicate of a real vertex v_i of an equivalent group EQ_a, with both connecting to a vertex v_j in an equivalent group EQ_b. If we replace an edge $e(v_d, v_j)$ with an edge $e(v_d, v_t)$, where $v_t \in EQ_b$, none of the private vertices' n-range CFPs will be changed.*

Proof 1 *As shown in Figure 4.6, we assume vertex v_i is in an equivalent group EQ_a, and vertices v_j and v_t are in an equivalent group EQ_b [5]. The equivalent groups are signified by dotted circles. After we insert a dummy vertex v_d into EQ_a as a duplicate of v_i, both v_i and v_d are connected to v_j but not v_t (see Figure 4.6a). At this moment, v_i and v_d have the same n-range CFP, and v_j and v_t have the same n-range CFP. Then we perform an edge switch by replacing the edge $e(v_d, v_j)$ with the edge $e(v_d, v_t)$ (see Figure 4.6b). We first prove that the n-range CFPs of v_j and v_t are not changed by this edge switching operation. We perform the proof by contradiction.*

[5] EQ_a and EQ_b can be the same equivalent group. It does not affect the proof and conclusion.

Without loss of generality, we assume the x^{th}-hop CFPs ($x \in [1, n]$) of v_j and/or v_t are changed. To be more specific, we assume there is a public vertex v_x which is in the x^{th}-hop CFPs of v_j and v_t before the edge switch, i.e., $h(v_x, v_j) = h(v_x, v_t) = x$. We assume the hop distances from v_i and v_d to v_x are h and h' respectively, as shown in the figure. According to the definition of hop distance (i.e., shortest path distance), $x \leq 1 + h$ and $x \leq 1 + h'$. After the edge switch, we assume the hop distances from v_j and/or v_t to v_x are changed. The edge switch has two steps i.e., deleting $\mathrm{e}(v_d, v_j)$ and adding $\mathrm{e}(v_d, v_t)$. In the following, we prove both steps will not change the hop distances from v_j and v_t to v_x. Deleting $\mathrm{e}(v_d, v_j)$ removes a path from v_j to v_x via v_d with a length of $1 + h'$. It changes the hop distance from v_j to v_x only if $x = 1 + h' < 1 + h$ when the original shortest path from v_j to v_x crosses v_d. Therefore, we have $h' < h$ which is not valid. This is because v_i and v_d are in the same equivalent group EQ_a and $h' = x - 1 < n$, therefore, $v_x \in CFP_{h'}(v_d)$ and hence $v_x \in CFP_{h'}(v_i)$. In other words, $h' = h$. Therefore, we know that the hop distance from v_j to v_x is not changed. Then, adding $\mathrm{e}(v_d, v_t)$ creates a new path from v_t to v_x via v_d with a length of $1 + h'$. It changes the hop distance from v_t to v_x only if $x > 1 + h'$ when the original shortest path from v_t to v_x is replaced by this path. Then this contradicts the previous statement that $x \leq 1 + h'$. Therefore, the hop distance from v_t to v_x is not changed as well. Since the hop distances from v_j and v_t to the public vertices within n range do not change, the n-range CFPs of v_j and v_t do not change.

As the n-range CFPs of v_j and v_t remain the same after the edge switch, the n-range CFP of v_d dose not change. This is because any public vertex that v_d can reach within n-hop via v_j before the switch can be reached by v_d via v_t by exactly the same number of hops after the switch. Therefore, the n-range CFP of v_d remains the same. Since the n-range CFPs (i.e., the hop distances to public vertices) of the vertices which are directly affected by the edge switching operation (i.e., v_d, v_j, and v_t) are not changed, the n-range CFPs of the other vertices in the network will not change as well. Our proof completes.

4.5 Edge Modification based Anonymization for 1-Range CFP Attacks

In Section 4.4, we proposed a k-anonymization algorithm based on adding dummy vertices to effectively protect a social network against n-range CFP attacks ($n \geq 1$). However, adding many dummy vertices to a social network may greatly affect the utility of the network. Therefore, in this section, we propose another k-anonymization algorithm purely based on modifying edges, which makes less damage to network utility. However, this algorithm is only applicable to resisting 1-range CFP attacks, and it is hard to be extended to prevent n-range CFP attacks when $n > 1$.

In order to convert a social network to achieve k-anonymity by edge modifications only, we need to group the private vertices into groups of size at least k. For private vertices in the same group, we modify their edges so that they connect to the same set of public vertices (i.e., their 1^{st}-hop CFPs are identical). To facilitate the comparison of their 1^{st}-hop CFPs, we use a standard binary vector to represent $CFP_1(v)$ for every private vertex v. Each bit in the binary vector corresponds to a public vertex v_{pub} in G. If v_{pub} presents in $CFP_1(v)$, the bit is on (i.e., set to one); otherwise the bit is off (i.e., set to zero). Take the social network depicted in Figure 4.1 as an example. Since there are 3 public vertices, a 3-bit vector is employed with the bits corresponding to v_1, v_2, and v_3 respectively. Then, vector 100 represents $CFP_1(v_4)$ as v_4 only connects to BBC (i.e., v_1), and vector 010 represents $CFP_1(v_5)$ as v_5 only connects to $Youtube$ (i.e., v_2).

In order to preserve the utility of the network, we prefer to modify only a small number of edges. Therefore, the groups should be formed carefully. Then, given a group of private vertices, their 1^{st}-hop CFPs in the form of binary vectors should be changed to the *median center* defined by Hamming distance [6]. For each vertex v in the group, the Hamming distance between the binary vertex corresponding to

[6]The median center of a set of binary vectors is itself a binary vector which has the same interpretation as the individuals.

Algorithm 2: Greedy clustering algorithm for 1-range CFP attacks.

```
   Input: A social network $G(V, E)$, privacy parameter $k$
   Output: A clustering of the private vertices $\mathcal{C}_G$
1  $\mathcal{C}_G = \emptyset$;
2  $V_{nc} = \{v_i | v_i \in V - V_{pub}\}$;
3  $\forall v \in V - V_{pub}$, calculate $CFP_1(v)$;
4  foreach $v_i \in V_{nc}$ do
5      $C_{v_i} = v_i \cup NeareastNeighbor(v_i, V_{nc}, k-1)$;
6  while $\mathcal{C}_{temp} = \{C_{v_i} | C_{v_i} \subseteq V_{nc}\} \neq \emptyset$ do
7      Find $C_{v_i} = \underset{C_{v_i} \in \mathcal{C}_{temp}}{\arg\min}\ cost(C_{v_i})$;
8      $\mathcal{C}_G = add(C_{v_i}, \mathcal{C}_G)$;
9      $V_{nc} = V_{nc} - C_{v_i}$;
10 while $|V_{nc}| \geq k$ do
11     Pick a vertex $v_j \in V_{nc}$ randomly;
12     $C_{v_j} = v_j \cup NeareastNeighbor(v_j, V_{nc}, k-1)$;
13     $\mathcal{C}_G^{new} = assign(C_{v_j}, \mathcal{C}_G)$;
14     if $cost(C_{v_j}) < (cost(\mathcal{C}_G^{new}) - cost(\mathcal{C}_G))$ then
15         $\mathcal{C}_G = add(C_{v_j}, \mathcal{C}_G)$;
16     else
17         $\mathcal{C}_G = \mathcal{C}_G^{new}$;
18     $V_{nc} = V_{nc} - C_{v_j}$;
19 if $V_{nc}$ is not empty then
20     $\mathcal{C}_G = assign(V_{nc}, \mathcal{C}_G)$;
21 return $\mathcal{C}_G$;
```

$CFP_1(v)$ and the median center reflects the number of edges of v that need to be modified. We then introduce the *anonymization cost* of a group as the total number of edges that need to be changed, considering all the vertices in the group. Based on this new measure, the 1-range CFP k-anonymizaiton problem can be reformed as a cost minimization clustering problem, formally presented in Definition 6.

Definition 6 (1-range CFP k-anonymizaiton problem) *Given a social network G and 1-range CFPs of all the private vertices, the 1-range CFP k-anonymizaiton problem is to cluster the private vertices to non-overlapping groups of size at least k and minimize the sum of the anonymization cost of every group.*

We propose a greedy clustering algorithm to solve the 1-range CFP k-anonymization problem. The main strategy is to cluster each private vertex with at least $k-1$ other private vertices having similar 1-range CFPs to reduce group anonymization cost.

Algorithm 3: Algorithm for assigning vertices to existing clusters $assign(C, \mathcal{C}_G)$.

```
Input: A group of vertices $C$ that need to be assigned, the existing cluster set $\mathcal{C}_G$
Output: A new cluster set $\mathcal{C}_G$
1  foreach vertex $v \in C$ do
2      Find the cluster $C_i \in \mathcal{C}_G$ nearest to $v$;
3      $C_i = add(v, C_i)$;
4      if $|C_i| \geq 2k$ then
           // split the cluster to two clusters
5          Find two vertices $v_l, v_t \in C_i$ which have the largest distance among all pairs
           of vertices in $C_i$;
6          $C_i^0 = \{v_l\}, C_i^1 = \{v_t\}$;
7          $C_i = C_i - \{v_l, v_t\}$;
8          while $C_i$ is not empty do
9              Find a vertex $v_x$ which is the nearest to $v_l$;
10             $C_i^0 = add(v_x, C_i^0)$;
11             $C_i = C_i - \{v_x\}$;
12             if $C_i$ is not empty then
13                 Find a vertex $v_y$ which is the nearest to $v_t$;
14                 $C_i^1 = add(v_y, C_i^1)$;
15                 $C_i = C_i - \{v_y\}$;
16         Remove $C_i$ from $\mathcal{C}_G$;
17         $\mathcal{C}_G = add(\{C_i^0, C_i^1\}, \mathcal{C}_G)$
18 return $\mathcal{C}_G$;
```

Note, with the binary vector representation, we employ Hamming distance to quantify the similarity of two 1-range CFPs. Then, we can adopt existing k nearest neighbor search algorithms for the clustering process. The clustering process is sketched in Algorithm 2.

The algorithm takes a social network G and a privacy parameter k as inputs, and it partitions all the private vertices of G into disjoint clusters such that the size of every cluster is no less than k, and vertices in the same cluster have identical 1-range CFPs. In the algorithm, we first define a set $\mathcal{C}_G$ to store resulting clusters, which is initialized to an empty set (line 1). Then, we form a vertex set V_{nc} containing all the private vertices that have not yet been assigned to a cluster (line 2). For each unassigned vertex $v_i \in V_{nc}$, we search for its $k-1$ nearest neighbors based on Hamming distance corresponding to 1^{st}-hop CFP vectors. v_i and its $k-1$ nearest neighbors form a group of size k (i.e., C_{v_i}) (line 4-5). Among all the $|V_{nc}|$ nearest neighbor groups, we choose the group with the lowest anonymization cost and at the same time, not overlapping with any existing clusters in $\mathcal{C}_G$ to form a cluster, and remove the vertices in the chosen nearest neighbor group from V_{nc}. The process repeats until none of the remaining nearest neighbor groups satisfies the conditions (lines 6-9).

Because we purposely only convert a nearest neighbor group to a cluster if it does not overlap with any existing clusters, it is very likely there are still some vertices in V_{nc} not yet assigned to any clusters. Two options are available to further process these unassigned vertices. We can either construct new clusters for those unassigned vertices or assign them to existing clusters. We decide which option to execute based on the anonymization cost that they cause. In other words, we always execute the one with lower anonymization cost (lines 10-18). To be more specific, for an unassigned vertex, we reform the nearest neighbor group by searching for its $k-1$ nearest neighbors in V_{nc} (line 12). We can form a new cluster with this nearest neighbor group. Alternatively, we can assign the vertices in this group to the existing clusters in $\mathcal{C}_G$ via the function $assign(C, \mathcal{C}_G)$, whose details will be given later (line 13). To decide which option to execute, we compare the anonymization cost incurred by adding a new cluster with the total anonymization cost incurred by assigning these vertices to existing clusters (line 14), and we execute the one with lower cost (lines 15-17). We repeat the above process until the number of the remaining vertices in V_{nc} is smaller than k. As cluster size must be no smaller than k, the remaining vertices cannot form a new cluster, thus we assign them to existing clusters (line 20).

As mentioned before, the function $assign(C, \mathcal{C}_G)$ is to assign a group of vertices C to the existing clusters in $\mathcal{C}_G$, and Algorithm 3 provides the details. For each vertex v in C, we assign it to its nearest cluster (lines 2-3). Here, the distance between a vertex v and a cluster C_i is measured by the Hamming distance between the binary vector corresponding to $CFP_1(v)$ and the binary vector representing the median center of C_i. As $assign(C, \mathcal{C}_G)$ keeps adding vertices to existing clusters, the size of clusters increases. In order to prevent a cluster from being over-sized, we propose a *cluster splitting* operation. To be more specific, when the size of a cluster

$C_i \in \mathcal{C}_G$ reaches $2k$ or larger, C_i is split into two new clusters to cut down overall anonymization cost. As stated in Lemma 2, the total anonymization cost of the two new clusters will always be lower than that of the original cluster no matter how we split the cluster. However, to gain a large reduction in anonymization cost, we adopt a greedy splitting process to make sure that vertices in the same new cluster have similar 1^{st}-hop CFPs, and the distance between the two new clusters is large (lines 5-17).

In the following, we detail the cluster splitting process. Firstly, we locate a pair of vertices v_l and v_t among all pairs of vertices in C_i with the maximum Hamming distance and then form two new clusters C_i^0 and C_i^1 around v_l and v_t respectively (lines 5-6). In order to meet the cluster size requirement, we split the rest of the vertices in C_i evenly between C_i^0 and C_i^1 so that each of the new clusters will have size no smaller than k. We locate the nearest vertex to v_l or v_t in an alternate fashion until all the remaining vertices in C_i are assigned (lines 8-17).

Lemma 2 *Given a group of binary vectors C, let C^0 and C^1 be two subgroups such that $C^0 \cup C^1 = C$ and $C^0 \cap C^1 = \emptyset$. Suppose vectors c, c^0, and c^1 signify the median centers of C, C^0, and C^1 respectively, $\sum_{v \in C^0} dist(v, c_0) + \sum_{v \in C^1} dist(v, c_1) \leq \sum_{v \in C} dist(v, c)$, where $dist()$ calculates Hamming distance.*

Proof 2 *The proof of this lemma is very simple. According to its definition, the median center of a group of vectors is the vector to which the sum of the distance from every member in the group is minimized. Therefore, $\sum_{v \in C^0} dist(v, c_0) \leq \sum_{v \in C^0} dist(v, c)$ and $\sum_{v \in C^1} dist(v, c_1) \leq \sum_{v \in C^1} dist(v, c)$. Because $C^0 \cup C^1 = C$, we have $\sum_{v \in C^0} dist(v, c_0) + \sum_{v \in C^1} dist(v, c_1) \leq \sum_{v \in C} dist(v, c)$.*

4.6 Experimental Evaluation

In this section, we conduct comprehensive experiments to evaluate the performance of the proposed k-anonymizaiton algorithms and report the results. We use four real-world networks in the experiments including **polblogs**, **facebook**, **grqc**, and **route-view**, which were introduced in Section 4.3.

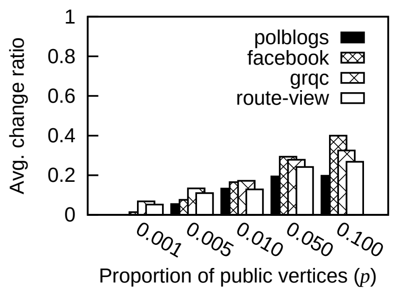
We evaluate the performance of the algorithms based on utility and time efficiency. We use *centrality*, a class of important network metrics, to evaluate the utility of the k-anonymized networks generated by the algorithms. The centrality of a vertex measures its relative importance in a network. Many network applications are developed based on centrality metrics, such as user influence analysis, information diffusion, and network robustness analysis. In this work, we evaluate four basic centrality metrics [95] listed as follows.

- The **degree centrality (DGE)** of a vertex v is defined as the number of edges adjacent to v.
- The **closeness centrality (CLS)** of a vertex v is defined as the inverse of the sum of its shortest path distances to all the other vertices, i.e., $\frac{|V|-1}{\sum_{\forall v_i \neq v \in V} h(v,v_i)}$.
- **Betweenness centrality (BTW)** quantifies the number of times a vertex acts as a bridge along the shortest path between two other vertices.
- **PageRank centrality (PRK)** measures the influence of a vertex in a network. It assigns relative scores to all the vertices in the network. The score of a vertex is defined recursively and depends on the number and scores of all the vertices it connects to. A vertex that connects to many high-score vertices receives a high score itself.

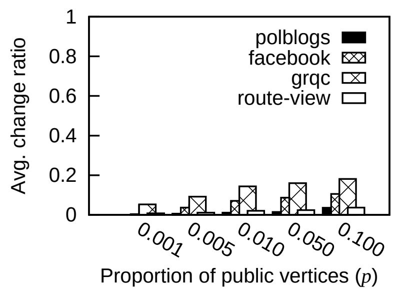
Given a network G and an anonymized network G^* generated based on G, we evaluate the utility of G^* based on the similarity of the centrality scores of vertices in G^* and G. Given a centrality function referred to as $cscore()$ and a set of vertices V_e that we would like to evaluate, we calculate the centrality scores of every vertex $v \in V_e$ in G and G^*, signified by $cscore(v, G)$ and $cscore(v, G^*)$ respectively. Then, we employ two metrics to evaluate the similarity of the centrality scores, i.e., the utility of G^*. The first is the *average change ratio* of centrality (i.e., $Avg_{v \in V_e} \frac{|cscore(v,G^*)-cscore(v,G)|}{cscore(v,G)}$). A small average change ratio indicates that the centralities of the vertices are similar in G and G^*. The second is *Spearman's rank correlation coefficient (SRCC)* [71]. To calculate this value, we rank the vertices in descending order of their centrality scores in G and G^* respectively. SRCC is then calculated by $\frac{\sum_{v \in V_e} (r_v^G - \overline{r^G})(r_v^{G^*} - \overline{r^{G^*}})}{\sqrt{\sum_{v \in V_e} (r_v^G - \overline{r^G})^2} \sqrt{\sum_{v \in G} (r_v^{G^*} - \overline{r^{G^*}})^2}}$, where r_v^G and $r_v^{G^*}$ are the ranks of a vertex v in G and G^* respectively, and $\overline{r^G}$ and $\overline{r^{G^*}}$ are the mean values of the ranks of all the evaluated vertices in G and G^* respectively. It evaluates the linear dependency of the centrality rankings of the same set of vertices in G and G^*. Its value is in the range of $[-1, 1]$. The value is close to 1 if the rankings of the vertices in G and G^* are strongly correlated, the value is close to 0 when the rankings are uncorrelated, and the value is close to -1 when the rankings are inversely correlated.

The experiments are performed on the datasets where public users are selected based on vertex degree (i.e., the degree based strategy in Section 4.3). According to the risk analysis, public users with the highest degree values cause the highest privacy risk. Accordingly, to achieve anonymity, we expect more modifications to networks, which result in larger utility loss. We evaluate the utility performance of our algorithms in this worst-case scenario.

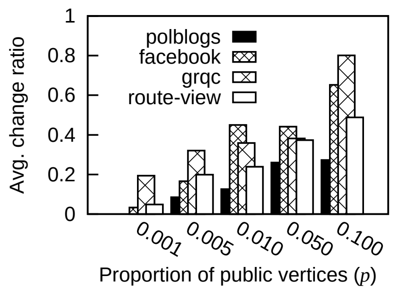
In the following, we first evaluate the n-range CFP anonymization algorithm presented in Section 4.4 and then the 1-range CFP anonymization algorithm presented in Section 4.5. Note these two algorithms are designed based on different principles, so we evaluate them separately. In addition, the 1-range CFP anonymization algorithm is less flexible than the n-range CFP anonymization algorithm as it is designed to only support 1-range CFP anonymization and not extendable for n-range CFP anonymization. However, it is more effective for preserving network utility compared with the general n-range CFP anonymization algorithm. It actually is able to preserve other important types of network utility in addition to centrality. We further demonstrate its capability of preserving other types of network utility with a new set of experiments, to be presented in Section 4.6.3.



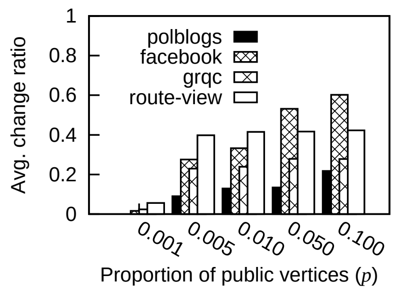
(a) DEG



(b) CLS



(c) BTW



(d) PRK

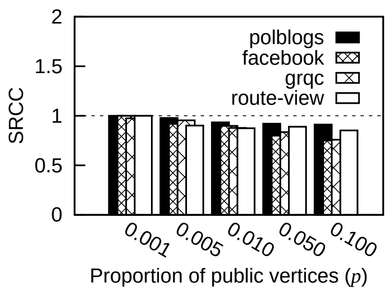
Figure 4.7: Average change ratio of every centrality property changes with proportion of public vertices ($k = 5, n = 2$).

4.6.1 Evaluating n-Range CFP Anonymization Algorithm

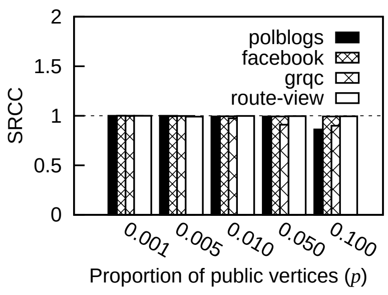
In this section, we evaluate the general n-range CFP anonymization algorithm. Because this algorithm anonymizes a network by adding dummy vertices which cannot be differentiated from the real private vertices in the anonymized network, we cannot effectively align the private vertices of the original network with those of the anonymized network and evaluate their centrality changes. In addition, it is expected that the algorithm will change many general network properties (e.g., diameter and density). However, we use experiments to show that the algorithm can preserve the centrality of public users [7].

Figure 4.7 and Figure 4.8 present the average change ratio and SRCC changes of the four centrality metrics with the proportion of public users p increasing, respectively. In the experiments, we set the privacy threshold k at 5 and the hop number n at 2. Generally, from these two figures, we observe that the utility of the anonymized networks degrades (i.e., the average change ratio increases, and the SRCC

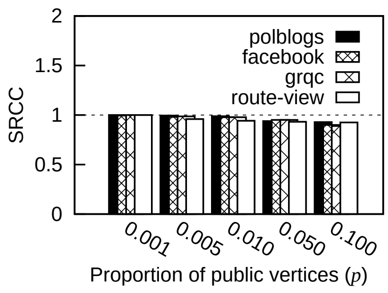
[7] To make centrality scores comparable between two networks with different numbers of vertices, we use normalized centrality scores.



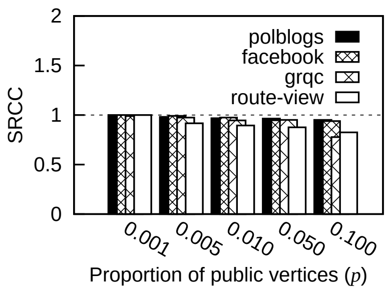
(a) DEG



(b) CLS



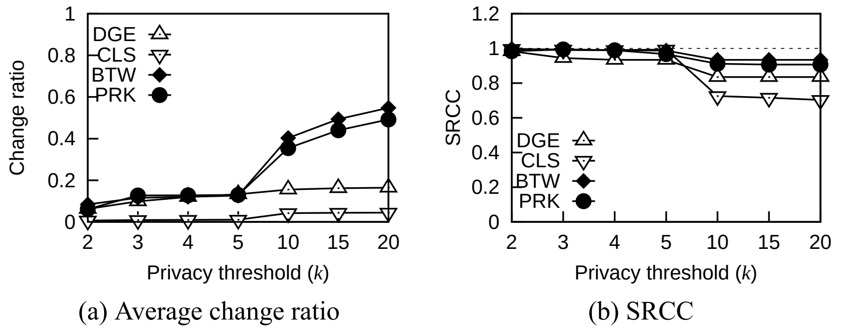
(c) BTW



(d) PRK

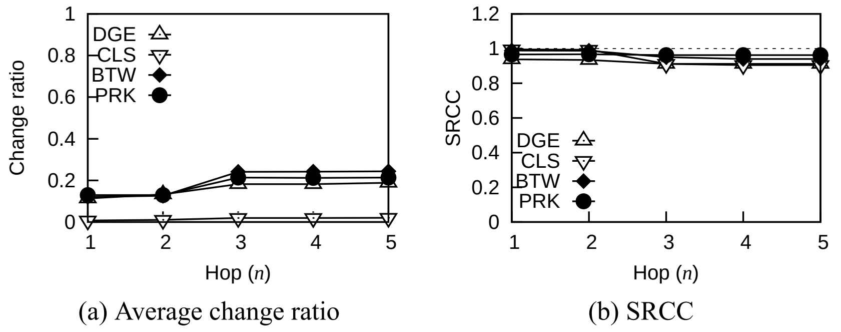
Figure 4.8: Spearsman's rank correlation coefficient of every centrality property changes with proportion of public vertices ($k = 5, n = 2$).

decreases) as p increases. It is consistent with our intuition that the more users' identities are released as public, the more privacy risk they cause, thus, the more utility is sacrificed to protect privacy. We further observe that for the **polblogs** network, the average change ratios are small under all the settings (i.e., less than 25%). Therefore, the algorithm is able to preserve the centrality of the public vertices well in this network. On the other hand, we find that for some networks (e.g., **facebook** and **grqc**), the average change ratios of some centrality metrics (e.g., **DEG**, **BTW**, and **PRK**) are small only when p is small (e.g., $p \leq 0.01$). The reason that the algorithm performs differently on preserving centrality on different networks is because these networks have different structures. However, the more specific relationships are yet to be discovered. Although our algorithm may cause large changes on the raw centrality scores sometimes, the SRCC values always remain close to 1 (see Figure 4.8). It demonstrates that our algorithm can preserve the ranks of public vertices based on centrality very well, which is useful for many applications, e.g., finding the most influential public users in social networks to initiate information propagation.



(a) Average change ratio (b) SRCC

Figure 4.9: Utility degrades as privacy threshold k increases on the polblogs network ($p = 0.01, n = 2$).



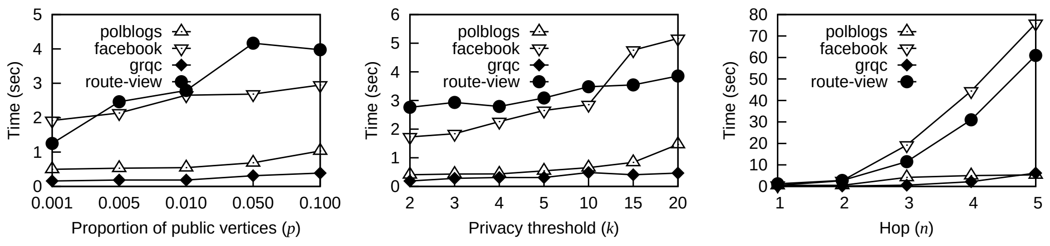
(a) Average change ratio (b) SRCC

Figure 4.10: Utility degrades as hop number n increases on the polblogs network ($p = 0.01, k = 5$).

Figure 4.9 and Figure 4.10 show the utility of anonymized networks changes with the privacy threshold k and hop number n increasing, respectively. We only show the results on the **polblogs** network since the results on the other networks have similar trends. Unsurprisingly, we find that utility degrades as k and n increase. With a relatively large k or n value (e.g., $k = 10$ or $n = 3$), our algorithm may cause large changes on some centrality scores (e.g., **BTW** and **PRK**). However, it preserves the centrality ranks of public vertices well as we find that the SRCC values are close to 1.

In addition to utility, we also evaluate the time performance of the n-range CFP anonymization algorithm [8]. Figure 5.21 reports the running time of this algorithm with increasing p, k, and n. We find that the running time increases slightly with p and k, and the algorithm is very efficient as the running time does not exceed 5

[8]We run the experiments on a server with 3.07GHz CPU and 128G RAM running a 64-bit windows OS.



(a) Running time V.S. p ($k = 5, n = 2$) (b) Running time V.S. k ($p = 0.01, n = 2$) (c) Running time V.S. n ($p = 0.01, k = 5$)

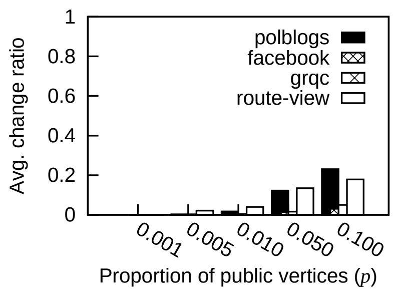
Figure 4.11: Time performance of the n-range CFP anonymization algorithm.

seconds on all the four tested networks even with large p and k ($n = 2$). We also find that the running time is much more sensitive to the change of the hop number n. When n is larger than 3, the running time increases rapidly. This is caused by the expensive cost of calculating the n-range CFP of every private vertex, which is based on breath-first network traversal. However, we think it is very hard for an attacker to learn the CFPs of target users beyond 3 hops in real-life situations. In addition, we find that the running time on the **facebook** and **route-view** networks is significantly longer than that on the other two networks. It is because **facebook** and **route-view** have the largest number of vertices and the highest edge density respectively.

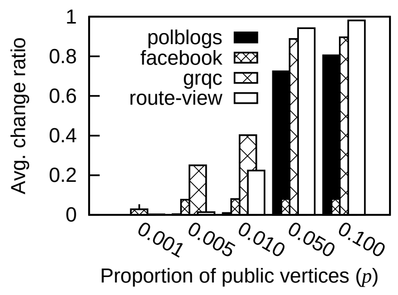
To sum up, we use extensive experiments to evaluate the performance of the n-range CFP anonymizaion algorithm. We find that this algorithm runs very efficiently on different networks. It causes small changes to the raw centrality scores of public vertices when p, k, and n values are small, and it can preserve the ranks of public vertices based on centrality very well.

4.6.2 Evaluating 1-Range CFP Anonymization Algorithm

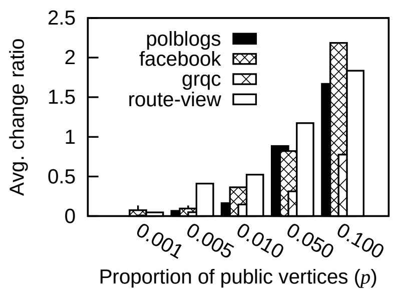
In the second set of experiments, we evaluate the special 1-range CFP anonymiation algorithm. Because this algorithm is implemented based on edge modifications and does not add dummy vertices to networks, we can easily align the private vertices in an original network with those in the corresponding anonymized network and evaluate the utility (e.g., centrality) of the anonymized network based on *all* vertices but



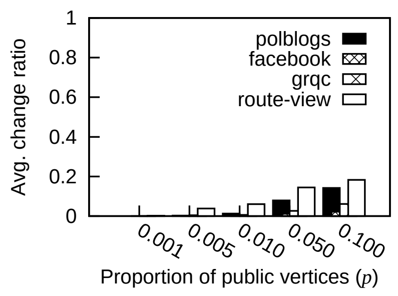
(a) DEG



(b) CLS



(c) BTW

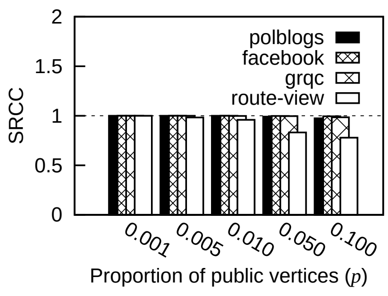


(d) PRK

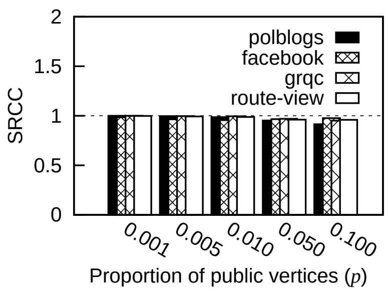
Figure 4.12: Average change ratio of every centrality property changes with proportion of public vertices ($k = 5$).

not only public vertices. We evaluate the utility of an anonymized network based on the four centrality metrics mentioned earlier. For each centrality metric, we calculate the average change ratio and SRCC based on all vertices in a network but not only public vertices.

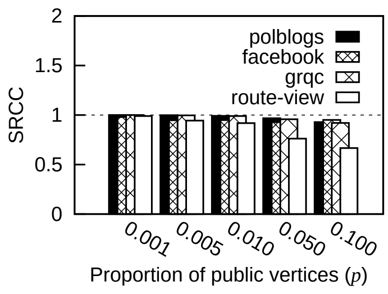
Figure 4.12 and Figure 4.13 present the average change ratio and SRCC changes of the four centrality metrics with the proportion of public users p increasing, respectively. In the experiments, we set the privacy threshold k at 5, and the hop number n is always 1. From Figure 4.12, we observe that our algorithm is able to preserve the DEG and PRK centrality scores well. The average change ratios of these two centrality metrics are always smaller than 22% on all the four tested networks. However, for the CLS and BTW centrality, we find the average change ratios become very large when p exceeds 0.005. This is mainly caused by a small number of outlier vertices which are previously at the boundary of a network with very small centrality. After the anonymization, they are moved to more "central" positions in the network with large centrality. These changes incur a few extremely large centrality change



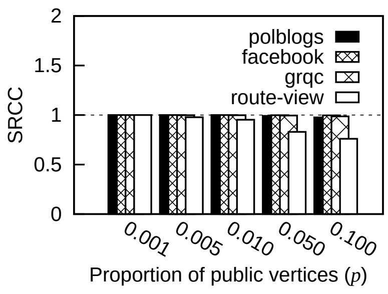
(a) DEG



(b) CLS



(c) BTW



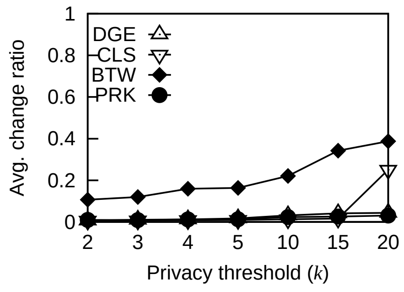
(d) PRK

Figure 4.13: Spearsman's rank correlation coefficient of every centrality property changes with proportion of public vertices ($k = 5$).

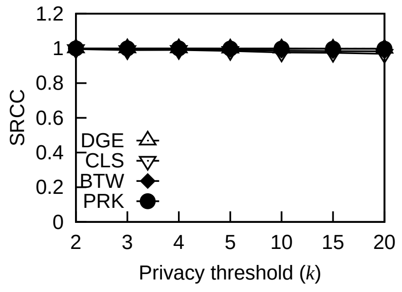
ratios, which are responsible for the very large average change ratios. Despite the large average change ratios, we find the SRCC values are always close to 1 for all the four evaluated centrality metrics as shown in Figure 4.13. It demonstrates that our algorithm can well preserve the centrality ranks of vertices.

Figure 4.14 reports the utility performance of the 1-range CFP anonymization algorithm changes with the privacy threshold k. We only show the results on the **polblogs** network but ignore the results on the other networks as they are very similar. We find in the figure that the average change ratios increase as k increases. They grow noticeably large when k exceeds a large value (e.g., 10), but the SRCC values remain almost 1 for all the tested centrality metrics.

We also evaluate the time cost of the 1-range CFP anonymiation algorithm and show the results in Figure 4.15. As observed from Figure 4.15a, the time cost of the algorithm slightly increases as p increases. It is because the algorithm greatly relies on the NN search of vertices based on 1^{st}-hop CFPs. The number of public vertices affects the length of the 1^{st}-hop CFP vector, and a long 1^{st}-hop CFP vec-

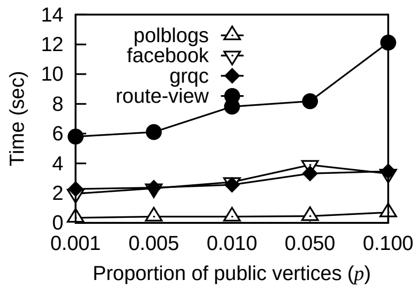


(a) Average change ratio

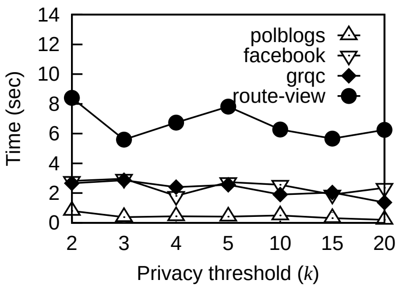


(b) SRCC

Figure 4.14: Utility degrades as privacy threshold k increases on the polblogs network ($p = 0.01$).



(a) Running time V.S. p $(k = 5)$



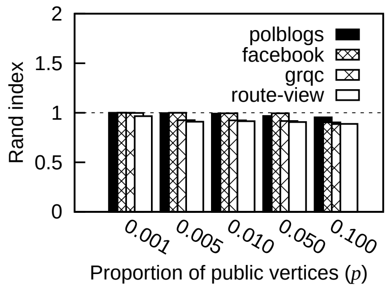
(b) Running time V.S. k $(p = 0.01)$

Figure 4.15: Time performance of the 1-range CFP anonymization algorithm.

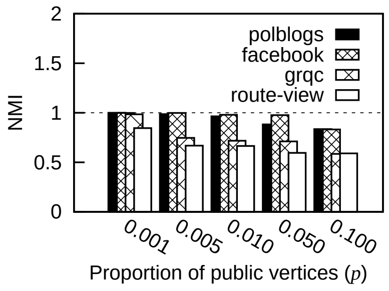
tor results in long time for calculating the distance between two vectors in the NN search. However, there are no obvious relationships between time cost and the privacy threshold k based on the results shown in Figure 4.15b. Moreover, we find that running time is also affected by the number of vertices in a network, which directly affects the running time of the NN search. In the experiments, the **route-view** network takes the longest running time, which is still less than 12 seconds under all the tested parameter settings.

4.6.3 Utility performance on other network properties

The previous experiments evaluate the utility performance of the proposed algorithms based on centrality. In this section, we demonstrate that the 1-range CFP anonymization algorithm is not only able to preserve centrality utility but also some other important types of network utility such as community structure and shortest paths.



(a) Rand index

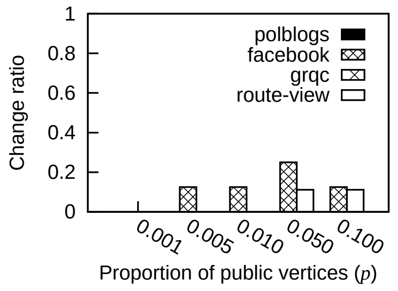


(b) NMI

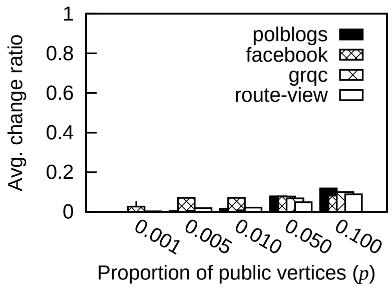
Figure 4.16: Community structure utility performance ($k = 5$).

We evaluate community structure utility by running a community detection algorithm on an anonymized network and the corresponding original network respectively and compare the generated communities. We use Clauset and Newman's classic greedy community detection algorithm for large social networks [22] to generate communities and use two standard metrics, namely *Rand index (Rand)* and *normalized mutual information (NMI)* [30], to compare the communities generated on an anonymized network with those of the corresponding original network. The Rand index is the ratio of the number of vertex pairs correctly clustered in both community partitions (i.e., either in the same or in different communities) to the total number of pairs. It has a value between 0 and 1, with 0 indicating that the two community partitions do not agree on any pair of vertices and 1 indicating that the partitions are exactly the same. Normalized mutual information is an information theory-based metric. It evaluates how much extra information one needs to infer one partition given the other. It equals 1 if the partitions are identical, whereas it has an expected value of 0 if the partitions are independent.

Figure 4.16 reports the results on the four tested networks with increasing p. As we can see in Figure 4.16a, the Rand index values are always very high (i.e., close to 1). However, the high Rand index values can be caused by the agreement of community partitions on the pairs of vertices which are not partitioned in the same community when there are many communities detected. Therefore, we further calculate the NMI metric and report the results in Figure 4.16b. We observe that, on the **polblogs** and **facebook** networks, the NMI values are high for every p setting,



(a) Diameter



(b) Pairwise shortest path distance

Figure 4.17: Shortest path utility performance ($k = 5$).

while on the other two networks, the NMI values are low when p is larger than 0.01. Therefore, the 1-range CFP algorithm is able to preserve network community structure when p is small (e.g., 1%).

We also evaluate shortest path utility. We calculate the all pair shortest paths of an anonymized network and the corresponding original network, and compare the shortest path lengths of the two networks. We evaluate the change ratio of network diameter, which is the length of the longest shortest path in a network, and the average change ratio of the shortest path lengths between all pairs of vertices. Figure 4.17 shows the results. We find that on the four tested networks, both the change ratio of diameter and the average change ratio of all-pair shortest path lengths are low. It means that the 1-range CFP anonymization algorithm is able to preserve network shortest path length information.

To sum up, in this section, we evaluate the anonymization algorithm proposed specially for 1-range CFP attacks. We find that with small p and k settings, this edge modification-based algorithm is able to preserve many different types of network utility, such as centrality, community, and shortest path lengths, of all vertices in a network but not only the centrality of public vertices.

4.7 Summary

In this Chapter, we study a new type of re-identification attacks, namely connection fingerprint (CFP) attacks, on social networks, in which an attacker re-identifies a private user in a social network based on his connection information with some known public users in the network. With a formal risk analysis on real-world networks, we find that CFP attacks can cause serious damage to identity privacy. Therefore, we propose two efficient network conversion algorithms against CFP attacks based on the notion of k-anonymity. The first algorithm is based on adding dummy vertices. It can resist powerful attackers with the connection information of a user with the public users within n hops ($n \geq 1$). Our experimental results show that it can preserve the centrality utility of public users. The other algorithm is based on edge modifications. It is only able to resist attackers with the connection information of a user with the public users within 1 hop but preserves a rich spectrum of network utility such as centrality, community, and shortest path lengths, of all users in a network but not only public users.

Chapter 5

High-Utility K-Anonymization for Social Network Conversion

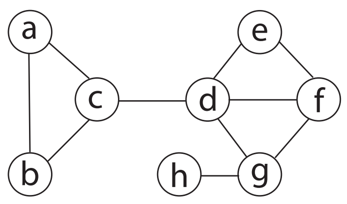
In this chapter, we discuss another aspect of a good privacy preserving network conversion scheme: utility. Utility is important because it directly affects the quality of converted social networks, which is crucial to the success of subsequent analysis tasks. Prior work converts a social network to achieve k-anonymity to protect privacy and preserves the utility of the converted network by minimizing a utility loss function evaluated by the number of edges modified on the network during the conversion. In this work, we find this simple utility benchmark is inadequate to preserve the utility of networks with complex structure. To improve utility performance, we propose a new utility benchmark which is defined by the changes that a network conversion algorithm makes to network community structure. In addition, we design a general network conversion algorithm framework which can be used with various k-anonymity schemes (e.g., k-degree anonymity, k-neighborhood anonymity) to protect privacy, and it minimizes utility loss based on the new utility benchmark. We conduct extensive experiments on real-world networks. The results demonstrate that our network conversion algorithm with the new utility benchmark can significantly improve the utility of converted networks compared with the existing algorithms using the simple utility benchmark.

5.1 Motivation

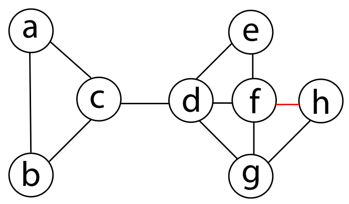
As discussed in Chapter 2.2, structural re-identification is one of the primary privacy concerns on social network data. To resist structural re-identification attacks, various *k-anonymity*-based network conversion algorithms have been proposed. These algorithms convert a social network by inserting some fake edges and/or deleting some existing edges to make sure for each vertex in the social network, there are at least $k - 1$ other vertices that are structurally indistinguishable from it based on certain attacker's background knowledge. Thereafter, the probability that an attacker successfully re-identifies a vertex based *only* on her background knowledge about the structure properties of the vertex will not exceed $1/k$.

Logically, a larger k provides stronger privacy protection, but it *does* come with a price. In an extreme case, a social network G is converted to a fully connected network G^*, in which every vertex is connected to all the other vertices, and all the vertices become structurally isomorphic. G^* achieves the highest possible level of anonymity against any structural background knowledge. However, G^* actually loses all the structural properties of G and hence is useless for data mining and analysis. The cost of network conversion is quantified by *utility loss*. Ideally, we prefer a converted social network which attains a desirable level of anonymity with the smallest utility loss so that the data mining and analysis results derived from the converted social network are very similar to those obtained from the original network.

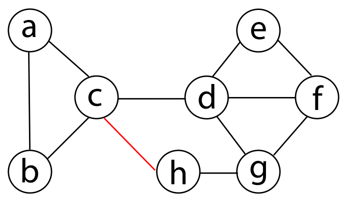
Prior work converts a social network by modifying (i.e., inserting and/or deleting) edges to achieve k-anonymity and reduces utility loss by minimizing the number of edges modified. This utility benchmark is based on a very simple intuition that the smaller the number of edges we modify on a network, the smaller the changes we make to the network structural properties. In this work, we argue that sometimes this simple intuition is not correct. A network is a complex structure, and modifying a small number of edges can make large changes to its structural properties. For example, removing a small number of bridge edges can change a connected network to



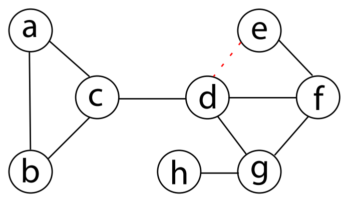
(a) A social network G



(b) Converted social network G_1^*



(c) Converted social network G_2^*



(d) Converted social network G_3^*

Figure 5.1: Examples of 2-degree anonymity achieved by modifying edges.

Table 5.1: Properties of original and anonymized networks

Networks	Num. of edges modified	APL	CC	BTW	CLN
G	0	2.07	3.75	0.50	0.50
G_1^*	1	2.00	3.50	0.54	0.52
G_2^*	1	1.86	3.00	0.41	0.55
G_3^*	1	2.21	4.25	0.43	0.47

disconnected. Therefore, the utility benchmark based only on this simple intuition may fail to safeguard important network properties.

We further demonstrate this point using another simple example. Given a social network G in Figure 5.1a, three converted networks G_1^*, G_2^*, and G_3^* are formed which all satisfy 2-degree anonymity as illustrated in Figure 5.1b, Figure 5.1c, and Figure 5.1d respectively. G_1^* and G_2^* are formed by inserting an edge between vertices f, h and vertices c, h respectively; G_3^* is formed by deleting the edge between vertices d and e. Based on the prior simple utility benchmark, G_1^*, G_2^*, and G_3^* have equally good utility because the numbers of edges modified on them are all 1. However, if considering some other important network structural properties, such as average path length (APL), average betweenness (BTW), clustering coefficient (CC), and average closeness (CLN), we find that these anonymized networks actually are very different as illustrated in Table 5.1. Among the three anonymized networks, G_1^* is the most similar to the original network G in terms of APL, BTW, CC, and CLN, while G_2^* and G_3^* are much more different from G. Consequently, the simple utility measure cannot capture the utility differences among G_1^*, G_2^*, and G_3^*, and naturally the k-anonymization algorithms developed based on it are very likely to generate networks (e.g., G_2^*, G_3^*) which do not have high utility.

Motivated by the fact that most of the existing k-anonymization algorithms use the simple utility model and may cause significant utility loss, we want to design a new utility measure. Instead of inferring utility merely based on the number of edges modified, it can directly evaluate the changes on network structure caused by modifying edges. Then, by minimizing the utility loss evaluated by this utility benchmark, k-anonymization algorithms can generate networks with high utility. However, this task is challenging due to the following issues.

The first issue is that the usage of converted networks is unknown. A social network is a complex data structure and has many topological properties, such as vertex degree, eigenvector, shortest paths, and community structure. Applications may have distinct requirements for the network properties that should be preserved in the converted networks. Moreover, there are many data mining tasks focusing on discovering unknown network properties. Therefore, we want to find an effective utility measure which can help to preserve many important network structural properties.

The second issue is that the utility measure needs to be quantitative and easy to calculate. In order to search for an optimal converted social network G^* that is the"closest" to the original network G in utility, we need to quantitatively evaluate the difference between G and G^* in terms of utility efficiently. There are some well-known metrics that can represent important structural features of a network and can be used to compare G and G^*, but they are difficult to calculate. For example, the average path length reflects the small-world property of social networks, but its calculation is based on expensive all-pair shortest path operations. Another example is the eigenvector, which has been proved to be strongly correlated with many network topological features. However, it requires expensive matrix decomposition.

The third issue is that the existing anonymization algorithms designed to minimize the number of edge modifications are not directly applicable to optimizing utility loss based on other utility measures, thus new algorithms need to be developed accordingly.

Optimizing utility during the social network anonymization process is a challenging problem, and it cannot be perfectly solved in one study. In this work, we take the initiative to explore this problem. We propose to build a utility measure based on *network community structure*. In other words, we evaluate the utility loss caused by modifying an edge based on the change it makes to network community structure. The community structure of a network reflects locally inhomogeneous edge distribution among the vertices, with high density of edges between vertices within one community and low density of edges between vertices from different communities. It is a central organizing principle of complex social networks, and has a strong correlation with many other important social network topological features such as betweenness, eigenvector, and clustering coefficient [84]. Thus by maintaining the community structure, we can preserve many important topological features of a network. In addition, the influence of edge insertion or deletion operations during the anonymization process can be easily reflected by the changes in the edge distribution within or between communities, and thus can be quantitatively evaluated. We try to minimize the impact of edge operations on network community structure during the anonymization process, so that we can generate an anonymized G^* which is structurally similar to G with respect to many important network properties.

To sum up, the main contributions of this work are listed as follows:

1. We identify the deficiency of the simple utility measure adopted by many existing k-anonymization algorithms for preserving network utility.

2. We propose a new utility measure based on network community structure to better capture the impact of an anonymization process on network topology. We consider both a flat community model and a hierarchical community model.

3. We design a general algorithm framework to anonymize a social network and minimize the utility loss evaluated by our new utility measure. This framework is general and can be used with many different k-anonymity schemes (e.g., k-degree anonymity and k-neighborhood anonymity).

4. We conduct extensive experiments to evaluate the proposed approach. The results show that our method achieves significant improvement in utility compared with existing k-anonymization algorithms.

5.2 Preliminaries

Following the notions defined in Section 4.2, we also model a social network as an undirected and unweighted graph $G(V, E)$ with a vertex $v_i \in V$ representing an entity in the social network, and an edge $e(v_i, v_j) \in E$ representing the social relationship between two entities v_i and v_j. We assume all the vertices are private in the social network so that their identities are removed in the converted social network G^* as specified in Definition 7.

Definition 7 (Converted Social Network) *The converted version $G^*(V^*, E^*)$ of a social network $G(V, E)$ is obtained by removing the identity information of all the vertices in G and with possible edge modifications (i.e., edge insertion and/or deletion) . $G^*(V^*, E^*)$ is used in place of G for data mining and analysis.*

We focus on *structural re-identification* attacks on social networks, in which an attacker aims to identify some target entities as vertices in G^* using her background knowledge about the structural properties of the targets. We use F to denote a background knowledge function that an attacker uses to determine the structural property of an entity and $F(v)$ to represent the structural property of an entity v with respect to F. For instance, F can be the number of adjacent edges of a vertex (i.e., degree), neighborhood, subgraph, etc. We formalize this attack in Definition 8.

Definition 8 (Structural Re-identification Attack) *Given a social network $G(V, E)$, its converted version $G^*(V^*, E^*)$, the background knowledge function F, and a target entity $v_t \in V$, the attacker searches for all the vertices in G^* that can provide the same result for F as v_t, referred to as $V_{F(v_t)}$, i.e., $V_{F(v_t)} = \{v \in V^* | F(v) = F(v_t)\}$. If $|V_{F(v_t)}| \ll |V^*|$, v_t has high probability to be re-identified.*

K-anonymity, as formally defined in Definition 9, is a widely adopted notion to prevent structural re-identification attacks on social networks [59, 99, 107, 108, 110]. By converting a social network to satisfy k-anonymity, an attacker will not be able to re-identify a vertex with probability larger than $1/k$ based on her background knowledge F. Note that we need to specify the type of background knowledge F that an attacker has in order to formally define k-anonymity (e.g., k-degree anonymity [59], k-neighborhood anonymity [107]). However, when the meaning of F is clear in context, we use k-anonymity for brevity of presentation.

Definition 9 (K-Anonymity) *Given a converted social network $G^*(V^*, E^*)$ and the background knowledge function F, G^* satisfies k-anonymity with respect to F, iff for each $v \in V^*$, there are at least $k-1$ other vertices $v' \in V^*$ with $F(v') = F(v)$.*

Most existing algorithms use edge modification-based approaches to anonymize a social network. That is to anonymize a network via inserting and/or deleting edges. In this work, we also focus on the edge modification-based approaches. It is expected that an anonymized social network is structurally different from the original network and hence loses some utility compared with the original network. Ideally, a social network conversion algorithm should take both privacy and utility into consideration. In other words, it should generate social networks that satisfy k-anonymity at minimum utility loss.

5.3 Utility Measure

Our goal is to design a high-utility social network anonymization scheme. Therefore, the key issue to address first is how to properly measure the utility loss of a converted social network. As discussed in Section 5.1, the prior utility benchmark simply measures the number of edges modified, and may fail to capture many important network topology changes. Hence, we want to design a more proper utility measure that can better capture complex social network structural properties.

We propose to build the utility measure based on network community structure. The community structure of a network reflects locally inhomogeneous edge distribution among the vertices. Given fixed community organization, the influence of an edge insertion or deletion operation can be reflected by the changes in the edge distribution within or between communities. In addition, community structure has a strong correlation with many other important topological features of social networks such as betweenness, eigenvector, and clustering coefficient [84]. Therefore, the impact of an anonymization process on community structure can also reflects its influence on other social network topological features. Our experimental study to be presented in Section 5.6 will validate this point.

There are various ways to model social network communities [84]. We consider both a flat community model and a hierarchical community model. Our utility measure is built around the idea of measuring the edge distribution change within and among communities. Therefore, the anonymization algorithm designed to minimize utility loss based on this utility measure preserves network community structure via preserving the edge distribution among a fixed community partition. In the following subsections, we will introduce our utility model in detail.

5.3.1 Flat Community-based Utility Model

First, we introduce how to build our utility measure based on a flat community model. The flat community model is defined as a non-overlapping partitioning of the vertices in a network, so that the vertices within one community are connected with edges of high density and the vertices across different communities are connected with edges of low density. Given a network $G(V, E)$, we assume it is divided into m disjoint communities denoted by $\mathcal{C}_G = \{C_1, C_2, \dots, C_m\}$, such that $\forall v \in V$, there

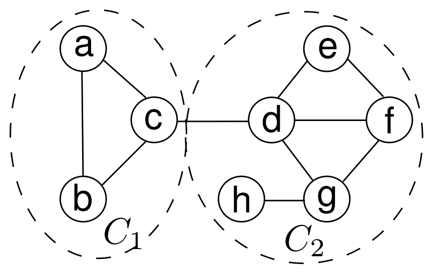


Figure 5.2: Example of flat community model.

Table 5.2: Edge distribution sequences and utility loss based on flat community model

(Anonymized) Networks	$ES\langle \frac{n_{11}}{\lvert E\rvert}, \frac{n_{12}}{\lvert E\rvert}, \frac{n_{22}}{\lvert E\rvert} \rangle$	$UL(G, G^*)$
G	$\langle \frac{3}{10}, \frac{1}{10}, \frac{6}{10} \rangle$	N.A.
G_1^*	$\langle \frac{3}{11}, \frac{1}{11}, \frac{7}{11} \rangle$	0.0727
G_2^*	$\langle \frac{3}{11}, \frac{2}{11}, \frac{6}{11} \rangle$	0.1636
G_3^*	$\langle \frac{3}{9}, \frac{1}{9}, \frac{5}{9} \rangle$	0.0888

is one community C_i containing v. For example, in Figure 5.2, the social network G is divided into 2 communities as signified by the dashed circles. Community C_1 contains vertices $\{a, b, c\}$, and C_2 contains $\{d, e, f, g, h\}$.

Here, we explain how to quantitatively measure the utility loss of a given modified network $G^*(V^*, E^*)$ based on the flat community model compared with the original network $G(V, E)$. The algorithm used to generate communities will be introduced later. Suppose the community structure of $G(V, E)$ is available. We can quantitatively capture the edge distribution within and among the communities via an edge distribution sequence, denoted by ES. To be more specific, we first count, i) for each community C_i, the number of edges that connect vertices within C_i, denoted by n_{ii}, and ii) for each pair of communities C_i and C_j $(i \neq j)$, the number of edges that connect vertices from C_i to vertices from C_j, denoted by n_{ij}. Thereafter, the percentage of edges distributed within community C_i is $\frac{n_{ii}}{|E|}$, and the percentage of edges distributed across C_i and C_j is $\frac{n_{ij}}{|E|}$. The edge distribution sequence ES is then defined as $\langle \frac{n_{11}}{|E|}, \frac{n_{12}}{|E|}, \ldots, \frac{n_{ij}}{|E|}, \ldots, \frac{n_{mm}}{|E|} \rangle$, $(1 \leq i \leq j \leq m)$. For instance, based on the community model of G depicted in Figure 5.2, there are 3 and 6 edges within community C_1 and community C_2 respectively, and one edge across them. Since the total number of edges in G is 10, $ES_G = \langle \frac{3}{10}, \frac{1}{10}, \frac{6}{10} \rangle$. Table 5.2 lists the ESs of the original network G and its three anonymized counterparts.

Given a network G and its converted version G^*, we assume their corresponding edge distribution sequences are ES_G and ES_{G^*} respectively. Compared with G, the utility loss caused by G^*, denoted by $UL(G, G^*)$, is measured by the L_1 distance between ES_G and ES_{G^*}, as defined in Equation (5.1). Using this measure, we calculate the utility loss of all the three anonymized networks in Table 5.2. We observe that anonymized network G_1^* causes the smallest utility loss. This is consistent with our observation in Section 5.1.

$$UL(G, G^*) = ||ES_G - ES_{G^*}||_1 = \sum_{1 \leq i \leq m} |ES_G[i] - ES_{G^*}[i]| \qquad (5.1)$$

There are various community detection algorithms proposed in the literature. In this work, we adopt a modularity-based method, one of the most popular community detection methods, to generate the flat community model. Modularity is a quality function defined based on a null model to express the "strength" of communities. By assumption, high values of modularity indicate good community partitions. In the interests of running time, we implement the greedy modularity maximization algorithm proposed in [73]. It is an agglomerative grouping method, where groups of vertices (partitions) are successively joined to form larger communities. It starts with every vertex as one partition. Then in each step, an edge is added to join two partitions such that the maximum increase or minimum decrease of modularity w.r.t. the previous configuration can be achieved. The algorithm terminates when one unified partition is formed. Among all the partition configurations generated in every step of this process, the one with the largest modularity value is returned as the result. The complexity of the algorithm is $O((|V| + |E|) \times |V|)$, and it performs well on large social networks [73]. However, we want to emphasize that our utility model is *independent* of the algorithm used for generating communities and any algorithm that generates good community partitions is applicable.

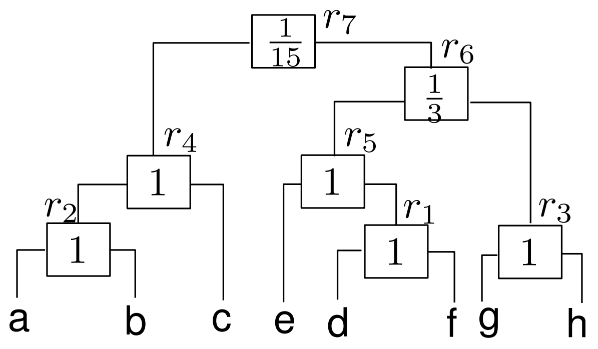


Figure 5.3: Example of hierarchical community model.

5.3.2 Hierarchical Community-based Utility Model

Some recent work suggests that communities of social networks often exhibit hierarchical organization (i.e., large communities in a social network may contain small communities). A hierarchical community model capture the structure information of a social network from coarse to fine scales, and presumably capture more detailed structure information of the network than the flat community model. As a result, the utility measure built upon the hierarchical community model is more sensitive to small network structure changes caused by an anonymization process, thus providing better control of utility loss. In the following, we discuss how to measure utility loss based on the hierarchical community model.

In this work, we use the *hierarchical random graph* (HRG) model proposed in [20, 21] to capture the hierarchical organization of social network communities. An HRG of a network $G(V, E)$ is a binary tree denoted by $\mathcal{H}_G$. Its leaf nodes correspond to the vertices in V, and each non-leaf node r roots a sub-tree denoted by T_r. The vertices in a sub-tree T_r are regarded as a community C_r. Thus, $\mathcal{H}_G$ organizes the communities hierarchically.

Each non-leaf node r of $\mathcal{H}_G$ is associated with a *connection probability* p_r. Here, p_r is the probability that a vertex in the left subtree T_r^L is linked by an edge with a vertex in the right subtree T_r^R. The larger the p_r, the stronger the connection between r's two subtrees. Mathematically, the connection probability p_r is defined in

Table 5.3: Edge distribution sequences and utility loss based on HRG model

(Anonymized) Networks	$ES\langle r_1, r_2, r_3, r_4, r_5, r_6, r_7\rangle$	$UL(G, G^*)$
G	$\langle\frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{2}{10}, \frac{2}{10}, \frac{2}{10}, \frac{1}{10}\rangle$	N.A.
G_1^*	$\langle\frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{2}{11}, \frac{3}{11}, \frac{2}{11}, \frac{1}{11}\rangle$	0.1454
G_2^*	$\langle\frac{1}{11}, \frac{1}{11}, \frac{1}{11}, \frac{2}{11}, \frac{2}{11}, \frac{2}{11}, \frac{2}{11}\rangle$	0.1636
G_3^*	$\langle\frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{2}{9}, \frac{1}{9}, \frac{2}{9}, \frac{1}{9}\rangle$	0.1777

Equation (5.2).

$$p_r = \frac{|E_r|}{|T_r^L| \cdot |T_r^R|} \tag{5.2}$$

where $|E_r|$ is the number of edges $e(v_i, v_j) \in E$ with $v_i \in T_r^L$ and $v_j \in T_r^R$, and $|T_r^L|$ and $|T_r^R|$ represent the numbers of vertices in r's left and right subtrees respectively. An HRG of the network shown in Figure 5.1a is depicted in Figure 5.3. The connection probability p_{r_4} of node r_4 is 1 as all the nodes in the left subtree (i.e., node a and node b) connect to all the nodes in the right subtree (i.e., node c). On the other hand, the connection probability p_{r_6} of node r_6 is $\frac{1}{3}$ as $|E_{r_6}| = |\{e(g,d), e(g,f)\}| = 2$, $|T_{r_6}^L| = |\{e, d, f\}| = 3$, and $|T_{r_6}^R| = |\{g, h\}| = 2$.

Given the HRG model of a network, we observe that the edges of the network are distributed among all the non-leaf nodes r_i, i.e., $\cup_{r_i} E_{r_i} = E$. We then generate the edge distribution sequence ES based on the edge distribution among these non-leaf nodes. Thus, $ES = \langle\frac{|E_{r_1}|}{|E|}, \frac{|E_{r_2}|}{|E|}, \ldots, \frac{|E_{r_m}|}{|E|}\rangle$, where m is the total number of non-leaf nodes on $\mathcal{H}_G$. Again, we define utility loss based on the L_1 distance between the ESs of the original and anonymized networks. Table 5.3 lists the ESs of the original network G and its three anonymized versions based on the HRG model together with their corresponding utility loss. Consistent with our expectation, we find that G_1^* causes the smallest utility loss. This simple example serves only to demonstrate that the utility measures based on the flat community model and the HRG model both can help to identify good anonymization options. We will conduct comprehensive experiments on real-world social networks to evaluate the performance of the community models in Section 5.6.

Given a social network G, the optimal HRG that fits it can be determined using the Markov Chain Monte Carlo method (MCMC) proposed in [20]. It first defines a likelihood function $\mathcal{L}$ to evaluate the fitness of a given HRG $\mathcal{H}_G$ to G, with $\mathcal{L}(\mathcal{H}_G) = \prod_{r \in \mathcal{H}_G} [p_r^{p_r}(1-p_r)^{1-p_r}]^{|T_r^L| \cdot |T_r^R|}$. The higher the likelihood score, the better $\mathcal{H}_G$ captures the topological structure of G. Then, the MCMC method samples the space of all possible HRGs with probability proportional to $\mathcal{L}$ and returns the one having the maximum $\mathcal{L}$ value. The MCMC method creates a Markov chain by defining several transitions from an HRG $\mathcal{H}_G$ to a new $\mathcal{H}'_G$. It calculates the logarithm of $\mathcal{L}$ of each HRG during sampling, and accepts a transition $\mathcal{H}_G \rightarrow \mathcal{H}'_G$ if $\Delta \log \mathcal{L} = \log \mathcal{L}(\mathcal{H}'_G) - \log \mathcal{L}(\mathcal{H}_G) \geq 0$. Otherwise, the transition is accepted with probability $exp(\log \mathcal{L}(\mathcal{H}'_G) - \log \mathcal{L}(\mathcal{H}_G))$. In the worst case, the time complexity of the MCMC method is exponential. However, it is stated in [20] that in practice MCMC converges to a plateau roughly after $O(|V|^2)$ steps.

5.4 High-Utility Anonymization

After introducing the community structure based utility measures, we are ready to present our high-utility k-anonymization algorithm that converts a given social network via edge operations and minimizes utility loss. In the following, we first present the basic idea of the algorithm and then detail its three main components, i.e., estimating local structure information, generating candidate edge operations, and refining target local structure.

5.4.1 Basic Idea and Algorithm Framework

Following prior work, we only consider achieving k-anonymity via performing a sequence of edge operations on a social network. As a result, the utility loss of the network is affected by both the number of edge operations performed and the impact of each edge operation. Consequently, our algorithm is designed based on a greedy strategy which tries to approach an k-anonymized network with minimum utility loss by finding a short edge operation sequence with each edge operation causing small utility loss.

Algorithm 4: High-utility k-anonymization algorithm

```
    Input: $G(V, E)$, $F$, and $k$
    Output: $K$-anonymized network $G^*$
 1  $M_G$ = constructCommunityModel($G$);
 2  $G^*(V^*, E^*) = G(V, E)$;
 3  $LS^{\triangle}$ = estimate($G, F, k$);
 4  while $G^*$ is not $k$-anonymized do
 5      $S_{op}$ = findCandidateOp($G^*, LS^{\triangle}, M_G$);
 6      while $S_{op} \neq \emptyset$ do
 7          operation $op = S_{op}.min\_op()$;
 8          $execute(op, G^*)$;
 9          $S_{op}$ = findCandidateOp($G^*, LS^{\triangle}, M_G$);
10      if $G^*$ is not $k$-anonymized then
11          $LS^{\triangle}$=refine($G$,$LS^{\triangle}$,$G^*$);
12  return $G^*$;
```

The basic idea of our algorithm is as follows. Given a network G, an attack model F, and a privacy requirement k, we carefully perform one edge operation at a time on G to approach k-anonymity against attacks under F. On one hand, to keep the number of edge operations as small as possible, we choose the edge operation that directs the current G towards its "nearest" k-anonymized network, which refers to the network that satisfies k-anonymity with the minimum number of edge operations denoted by $G^{\triangle}$ to facilitate our explanation. On the other hand, among all the possible edge operations that can direct G towards $G^{\triangle}$, at each step we perform the one which causes the smallest utility loss based on our utility loss measure introduced in Section 5.3.

In this process, the knowledge of $G^{\triangle}$ is essential, which, however, is unknown and hard to locate. Given the fact that forming $G^{\triangle}$ directly is not always possible, we try to estimate $G^{\triangle}$ by deriving the local structure information of its vertices (e.g., the vertex degree, and/or the neighbors' degrees of each vertex) which, based on the given G, F, and k, is possible. Then, according to the local structure information of $G^{\triangle}$, a set of candidate edge operations leading G towards $G^{\triangle}$ can be derived, and we always perform the one that causes the smallest utility loss.

Algorithm 4 sketches a high-level outline of our high-utility k-anonymization algorithm. It takes a network G, an attack model F, and a privacy parameter k as inputs, and outputs a converted network G^* that is k-anonymized and has small utility loss. Initially, the algorithm constructs and maintains a flat/hierarchical community model of G as M_G (line 1). Then, it initializes G^* to G, and derives the local structure information $LS^\triangle$ of $G^\triangle$ based on G, F, and k (lines 2-3). Thereafter, it generates a set of candidate edge operations (i.e., edge insertion operations and/or edge deletion operations) based on the current G^* and the estimated target $LS^\triangle$. The candidate operations are maintained by a set S_{op} together with the utility loss score of every edge operation that is calculated based on the community model M_G (line 5). At each step, our algorithm performs the edge operation which causes the smallest utility loss on G^*, and re-generates the candidate operation set based on the updated G^* (lines 7-9). This process continues until S_{op} becomes empty (lines 6-9). After performing all the identified candidate edge operations, there are two possible outcomes depending on whether the current G^* is k-anonymized. If G^* is k-anonymized, the algorithm terminates and returns G^* as the result (line 12). Otherwise, G^* does not satisfy the privacy requirement, i.e., the k-anonymized network with the local structure information $LS^\triangle$ has not been achieved. In this case, we need to refine the target $LS^\triangle$ via small adjustments, and continue the previous process (line 11). We would like to point out that in refining the target $LS^\triangle$, we only consider additive adjustments, which nudge $LS^\triangle$ towards the local structure of a complete network. Thus, in the worst case, G^* will be the complete network, which always satisfies the privacy requirement (if $|V| \geq k$) [1]. Therefore, our algorithm is convergent.

In the following, we detail the three key components of Algorithm 4, i.e., estimating local structure information, generating candidate edge operations, and refining local structure information.

[1]We want to highlight that the additive adjustments only apply to the refining local structure information step of Algorithm 1 (i.e., lines 10-11). In the other parts of the algorithm (e.g., lines 5-9 where edge operations are performed to change the current network towards the target network), we consider both edge addition and deletion operations.

5.4.2 Estimating Local Structure Information

Deriving the local structure information of $G^{\triangle}$ (i.e., the k-anonymized network with the minimum number of edge operations) is essential as it sets the target for the anonymization algorithm. In this section, we explain how this task is performed. As stated earlier, we only focus on k-degree anonymity for simplicity. Since the privacy requirement sets constraints on the vertex degree of the target anonymized network, we estimate the *degree sequence* $DS^{\triangle}$ as the local structure information of $G^{\triangle}$. The degree sequence DS of a network $G(V, E)$ is a vector of size $|V|$ with each element $DS[i] \in DS$ representing the degree of a vertex in G. We further assume that the degree sequence is sorted in non-ascending order of its elements (i.e., $DS[1] \geq DS[2] \geq \ldots \geq DS[|V|]$). There are some observations that can guide the estimation. First, $DS^{\triangle}$ has the same size as DS, because we only consider network modification via edge operations but not vertex operations. Second, $DS^{\triangle}$ must be k-anonymized since $DS^{\triangle}$ is the degree sequence of a k-degree anonymized network $G^{\triangle}$. In other words, for each element $DS^{\triangle}[i] \in DS^{\triangle}$, there are at least $k-1$ other elements with the same value as $DS^{\triangle}[i]$. Third, because $DS^{\triangle}$ is the degree sequence of the "nearest" k-anonymized network $G^{\triangle}$, the L_1 distance between $DS^{\triangle}$ and DS should be minimized. Based on the above observations, we employ the dynamic programming method proposed in [59] to find the optimal $DS^{\triangle}$ with $O(|V|^2)$ time complexity. A greedy algorithm is also available in [59] with time complexity $O(k|V|)$. We ignore the detail because it is not the focus of our work.

5.4.3 Generating Candidate Edge Operation Set

Once the target local structure information $DS^{\triangle}$ is ready, we need to find candidate edge operations that convert the current G^{*} to a k-anonymized network with its degree sequence matching $DS^{\triangle}$. Before introducing the detailed algorithm, we define three basic types of edge operations, i.e., *edge insertion*, *edge deletion*, and *edge shift* denoted by $ins(v_i, v_j)$, $del(v_i, v_j)$, and $shift((v_i, v_j), (v_i, v_k))$. As suggested by their names, $ins(v_i, v_j)$ inserts a new edge that links vertex v_i to vertex

v_j, and $del(v_i, v_j)$ removes the edge between v_i and v_j. $shift((v_i, v_j), (v_i, v_k))$ replaces edge $e(v_i, v_j)$ with edge $e(v_i, v_k)$. Note that $e(v_i, v_j)$ and $e(v_i, v_k)$ in the edge shift operation are carefully selected so that it does not change the edge distribution of the constructed community model, thus the operation causes zero changes to the utility loss score. For example, as shown in Figure 5.2, G is partitioned into two communities. Edge $e(c, d)$ is the crossing edge between these two communities. An edge shift operation $shift((c, d), (c, h))$ shifts the end point d of this edge to vertex h. Since d and h are in the same community, there is still one edge spanning these two communities, thus it does not change the edge distribution on the flat community model. Edge shift operations based on the flat community model is formally expressed in Definition 10. Similarly, edge shift operations can be defined on the hierarchical community model as described in Definition 11. Due to the unique feature of edge shift operations, they should receive high priority when modifying the network to achieve k-anonymity.

Definition 10 (Edge Shift on Flat Community Model) *Given a network $G(V, E)$, the corresponding flat community model $\mathcal{C}_G$, an edge $e(v_i, v_j) \in E$, and a vertex $v_k \in V$ such that $e(v_i, v_k) \notin E$, if v_j and v_k are in the same community, an edge shift operation $shift((v_i, v_j), (v_i, v_k))$ replaces $e(v_i, v_j)$ with $e(v_i, v_k)$.*

Definition 11 (Edge Shift on HRG) *Given a network $G(V, E)$, the corresponding HRG $\mathcal{H}_G$, an edge $e(v_i, v_j) \in E$, and a vertex $v_k \in V$ such that $e(v_i, v_k) \notin E$, let r be the lowest common ancestor of v_j and v_k on $\mathcal{H}_G$. If v_i is not in the subtree of r, an edge shift operation $shift((v_i, v_j)$, $(v_i, v_k))$ replaces $e(v_i, v_j)$ with $e(v_i, v_k)$.*

The goal of the edge operations is to modify the network such that its new degree sequence DS^* matches the target degree sequence $DS^{\triangle}$. Therefore, the degree difference sequence $\delta = (DS^{\triangle} - DS^*)$ can give some guidance. Each element $\delta[i] \in \delta$ with $\delta[i] > 0$ (i.e., $DS^*[i] < DS^{\triangle}[i]$) indicates that a vertex in $G^*(V^*, E^*)$ with degree $DS^*[i]$ needs to increase its degree, i.e., it should connect to more edges. We maintain $DS^*[i]$s with $\delta[i] > 0$ via a set DS^+ and all the vertices $v \in V^*$ that

$$\begin{array}{lcl} DS & : & \{4\ 3\ 3\ 3\ 2\ 2\ 2\ 1\}\ VS^{+} = \{\{c, f, g\}_{EX}, h\} \\ DS^{*} & : & \{4\ 4\ 3\ 3\ 2\ 2\ 2\ 2\}\ VS^{-} = \emptyset \\ \delta & : & \{0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\} \end{array}$$
Candidate operations: $ins(f, h),\ ins(c, h)$

Figure 5.4: Generation of candidate operations.

have degree $DS^*[i]$ via a set VS^+. VS^+ contains all the vertices that may require edge insertion operations. Similarly, each element $\delta[j] \in \delta$ with $\delta[j] < 0$ (i.e., $DS^*[j] < DS^{\triangle}[j]$) indicates that a vertex in G^* with degree $DS^*[j]$ needs to decrease its degree, i.e., it should connect to fewer edges. We maintain $D^*[j]$s with $\delta[j] < 0$ via a set DS^- and all the vertices $v \in V^*$ that have degree $DS^*[j]$ via a set VS^-. VS^- contains all the vertices that may require edge deletion operations. Note that a degree value of $DS^*[i]$ or $DS^*[j]$ may correspond to multiple vertices in G^*, and we treat them equally. In addition, if the degree value $DS^*[i]$ ($DS^*[j]$) only appears once in DS^+ (DS^-), we cannot perform edge insertion (deletion) to connect (disconnect) two vertices v_l, v_k both with the degree of $DS^*[i]$ ($DS^*[j]$). Hence we mark these vertices as being *mutually exclusive*, denoted by $EX(v_l, v_k) = True$. The *mutual exclusive set* is formally described in Definition 12.

Definition 12 (Mutual Exclusive Set) *Given a difference sequence $\delta = (DS^{\triangle} - DS^*)$ of the target degree sequence $DS^{\triangle}$ and the degree sequence DS^* of a current network $G^*(V^*, E^*)$, we define the degree set $DS^+ = \{DS^*[i] \in DS^* | \delta[i] > 0\}$ and the vertex set $VS^+ = \{v_j \in V^* | \exists DS^*[i] \in DS^+, d(v_j) = DS^*[i]\}$ with $d(v_j)$ referring to the degree of vertex v_j. Let $G_d = \{v_j | v_j \in VS^+ \wedge d(v_j) = d\}$ and $D_d = \{DS^*[i] | DS^*[i] \in DS^+ \wedge DS^*[i] = d\}$. If $|D_d| = 1$, $G_d = \{v_1, v_2, \ldots, v_t\}$ is a mutual exclusive set denoted by $\{v_1, v_2, \ldots, v_t\}_{EX}$ and each pair of vertices $v_l, v_k \in G_d$ are mutually exclusive, denoted by $EX(v_l, v_k) = True$. The mutual exclusive set on DS^- can be defined analogously.*

Back to the network G depicted in Figure 5.1a. Its degree sequence DS and the target 2-degree anonymized degree sequence $DS^{\triangle}$ are shown in Figure 5.4. We derive $\delta = (DS^{\triangle} - DS) = (0, 1, 0, 0, 0, 0, 0, 1)$, and find $\delta[2] = \delta[8] = 1 > 0$.

Hence, $DS[2]$ $(= 3)$ and $DS[8]$ $(= 1)$ are inserted into DS^+. Then, all the vertices in G with degree 3 or 1 are inserted into VS^+, i.e.,$VS^+ = \{c, f, g, h\}$. Note that $\{c, f, g\}$ is a mutual exclusive set. As there are no elements of δ with values smaller than 0, $DS^- = VS^- = \emptyset$.

The reason that we form the VS^+ and VS^- sets is to facilitate generating candidate edge operations. As VS^+ contains the vertices that need larger degrees, $ins(v_i, v_j)$ is a candidate operation if $v_i, v_j (i \neq j) \in VS^+ \wedge e(v_i, v_j) \notin E^* \wedge EX(v_i, v_j) \neq True$. We enumerate all candidate edge insertion operations based on VS^+ and put them in a set Op^{ins}. Similarly, $del(v_i, v_j)$ is a candidate edge deletion operation if $e(v_i, v_j) \in E^* \wedge v_i, v_j (i \neq j) \in VS^- \wedge EX(v_i, v_j) \neq True$. Again, we explore all candidate edge deletion operations and save them in a set Op^{del}. We also consider candidate edge shift operations. For a pair of vertices (v_j, v_k) $(j \neq k)$ with $v_j \in VS^-$ and $v_k \in VS^+$, if there is a vertex $v_i (i \neq j, k)$ such that v_i, v_j, and v_k satisfy the condition in Definition 10 (or Definition 11 if the HRG model but not the flat community model is used), $shift((v_i, v_j), (v_i, v_k))$ is a candidate. All possible edge shift operations form another set Op^{shift}. We continue the above example shown in Figure 5.4. As $VS^- = \emptyset$, we only need to consider possible edge insertion operations, i.e., $Op^{del} = Op^{shift} = \emptyset$. Based on $VS^+ = \{c, f, g, h\}$ and the mutual exclusive set, we have $Op^{ins} = \{ins(c, h), ins(f, h)\}$.

Given all the candidate edge operations maintained in the operation sets Op^{ins}, Op^{del}, and Op^{shift} respectively, we insert them into a candidate operation set S_{op} which is used by the high-utility k-anonymization algorithm (i.e., Algorithm 4). Before inserting the edge operations into S_{op}, we need to calculate the utility loss caused by each of them, so that edge operations causing smaller utility loss will be performed earlier. Continue our example depicted in Figure 5.4 with the utility loss scores calculated based on the flat community model in Figure 5.3. The corresponding candidate edge operation set S_{op} is set to $\{\langle ins(f, h), 0.0727\rangle, \langle ins(c, h), 0.1636\rangle\}$. Algorithm 5 presents the pseudo code for finding candidate operations.

Algorithm 5: **findCandidateOp** algorithm

```
    Input: $G^*(V^*, E^*)$, $DS^\triangle$, a community-based network model $M_G$
    Output: Candidate operation set $S_{op}$
1  $DS^* =$ degree sequence of $G^*$;
2  $\delta = (DS^\triangle - DS^*)$;
3  $DS^+ = \{DS^*[i] \mid \delta[i] > 0, 1 \leq i \leq |DS^*|\}$;
4  $DS^- = \{DS^*[j] \mid \delta[j] < 0, 1 \leq j \leq |DS^*|\}$;
5  $VS^+ = VS^- = \emptyset$;
6  foreach $d \in DS^+$ do
7      $VS^+ = VS^+ \cup \{v_i | v_i \in V^*, d(v_i) = d\}$;
8  foreach $d \in DS^-$ do
9      $VS^- = VS^- \cup \{v_j | v_j \in V^*, d(v_j) = d\}$;
10 $Op^{ins} = getOp(VS^+, VS^+)$;
11 $Op^{del} = getOp(VS^-, VS^-)$;
12 $Op^{shift} = getOp(VS^+, VS^-, M_G)$;
13 calculate the cost of each operation in $Op^{ins}$, $Op^{del}$, and $Op^{shift}$;
14 $S_{op}.insert(Op^{ins}, Op^{del}, Op^{shift})$;
15 return $S_{op}$;
```

5.4.4 Refining Target Local Structure Information

As mentioned above, our high-utility k-anonymization algorithm generates $DS^\triangle$ that estimates the local structure information of the "nearest" k-anonymized network as the target, and performs edge operations to change the current network towards $DS^\triangle$. However, it is possible that the k-anonymized network with the degree sequence $DS^\triangle$ is not achievable by the current executed operation sequence. If this happens, we fine-tune $DS^\triangle$ and start another attempt. We prefer that the new target degree sequence is close to the old one, and we only consider additive adjustments on $DS^\triangle$ to ensure the convergence of our algorithm. Hopefully, the new target $DS^\triangle$ will be achievable through our anonymization process.

In order to decide how to adjust $DS^\triangle$, we first consider VS^+ which contains the vertices that have not been k-anonymized and need to increase their degrees. For any vertex v_i in VS^+, we cannot find a vertex v_j to form a valid $ins(v_i, v_j)$ operation to increase its degree according to the current $DS^\triangle$. Therefore, we want to find an element on $DS^\triangle$ to increase its value so that later we could find a candidate operation that increases v_i's degree. For each $v_i \in VS^+$, we find some vertices $v_j \in V$ $(i \neq j)$ such that $e(v_i, v_j) \notin E^*$ and $EX(v_i, v_j) \neq True$. These v_js form the candidates of which we could increase the target degree on $DS^{\triangle}$. We could randomly choose one among the candidate v_js to increase its target degree by a small value (e.g., 1) on $DS^{\triangle}$, but it may break the k-anonymity of $DS^{\triangle}$. Therefore, instead of directly increasing the target degree of v_j on $DS^{\triangle}$, we increase the degree of v_j on DS, and then we re-generate $DS^{\triangle}$ based on the updated DS. As the change made to DS is very small, the new $DS^{\triangle}$ should be very similar to the old one.

We only consider VS^{+} above because we want to make additive changes to $DS^{\triangle}$. However, when VS^{+} is empty, we have to utilize VS^{-} that contains the vertices which have not been k-anonymized and need to decrease their degrees. Similar to the above, for each vertex $v_i \in VS^{-}$, we could find a vertex v_j to reduce its target degree to make a valid candidate operation $del(v_i, v_j)$. However, this is against our goal of only making additive changes. Alternatively, for each vertex $v_i \in VS^{-}$, we increase the degree of another vertex v_j on DS whose degree value is closest to that of v_i in G, then re-generate $DS^{\triangle}$ based on the updated DS. The rationale is that because v_j and its corresponding $v_i \in VS^{-}$ have similar degree, they are very likely to be anonymized to the same degree in the anonymized network. After the degree of v_j has been increased, v_i may not need to decrease its degree anymore.

5.4.5 Computational Complexity

In this subsection, we briefly discuss the time complexity of our algorithm by analyzing the time complexity of each step in the k-anonymization process.

Based on Section 5.4.2, estimating local structure information can be processed in $O(k|V|)$ time. Then, the following step is to generate candidate operations. In this step, firstly, a one-time scan on the difference sequence δ is performed to generate the vertex sets VS^{+} and VS^{-}, which takes time $O(|V|)$. Given $t = \max\{|VS^{+}|, |VS^{-}|\}$, it takes $O(t^2)$ time to generate all candidate operations, and the total number of candidate operations generated is at most $O(t^2)$. For each candidate operation, we need to calculate the utility loss that it may cause based on the given community model, which is the L_1 distance between the edge distribution sequences as described in Section 5.3. Usually, the time complexity of calculating L_1 distance is linear to the length of the edge distribution sequence represented by l. However, we observe that each edge operation only affects the number of edges corresponding to at most one element of the edge distribution sequence, as well as the total number of edges. For example, if we add one edge $e(c, h)$ in the example network in Figure 5.1a, it will only affect n_{12} (i.e., the number of edges distributed between communities C_1 and C_2) of the flat community model in Figure 5.2, and increase the total number of edges by 1. Suppose ES and ES^* are the edge distribution sequence of the original network and that of the modified network via adding $e(c, h)$, the L_1 distance between them is calculated as follows: $||ES - ES^*||_1 = \sum_{ij}(\frac{n_{ij}}{|E|} - \frac{n_{ij}}{|E|+1}) - (\frac{n_{12}}{|E|} - \frac{n_{12}}{|E|+1}) + (|\frac{n_{12}}{|E|} - \frac{n_{12}^*}{|E|+1}|) = (\frac{1}{|E|} - \frac{1}{|E|+1}) \sum_{ij} n_{ij} - (\frac{n_{12}}{|E|} - \frac{n_{12}}{|E|+1}) + (|\frac{n_{12}}{|E|} - \frac{n_{12}^*}{|E|+1}|)$. With $\sum_{ij} n_{ij}$ being calculated only once within $O(l)$ time, the utility loss of every candidate operation can be easily calculated in constant time. This observation also applies to the HRG model. Consequently, calculating the utility loss scores of all candidate operations can be proceed in $O(l + t^2)$ time, including spotting the operation with the lowest utility loss. To sum up, the overall computation time of the second step is $O(|V|) + O(l + t^2)$. In the worst case, when all vertices need to increase or decrease their degrees, $t = \max\{|VS^+|, |VS^-|\} = |V|$. For the flat community model, $l = m^2$ with m being the number of communities in the model. For the HRG model, $l = |V| - 1$. Therefore, the upper bound of the value of l is $|V|^2$. Consequently, the worst-case time complexity of generating candidate operations is $O(|V|^2)$. However, in real cases, the degree sequence of a social network usually follows a power-law distribution, resulting in low frequencies of vertices sharing large degree values and high frequencies of vertices having small degree values. In other words, in a social network, it is very likely that a large amount of low-degree vertices already satisfy k-degree anonymity and hence do not need to change their degree, i.e., $t << |V|$. Then, the number of communities m is also far smaller than $|V|$. Therefore, the running time of the candidate operation generation process most likely will be much smaller than that in the worst case.

The running time of the refinement process depends on the size of the remaining VS^+ or VS^- sets. Suppose the size is t, the time complexity of this process is $O(t|V|)$. Generally, $t << |V|$, thus $t|V| << |V|^2$.

Based on the above analysis, the worst-case computational complexity of one iteration of the k-anonymization process is $O(|V|^2)$, but the real running time of this process in most cases is much shorter. As it is hard to anticipate the number of iterations that the algorithm will run before convergence, we will conduct experiments to further test the time performance of our algorithm in Section 5.6.

5.5 Discussion

In this section, we further enrich this work by discussing the security of the k-anonymity scheme against minimality attacks and discuss the extendability of our k-anonymization algorithm to other k-anonymity schemes e.g., k-neighborhood anonymity.

5.5.1 Privacy Analysis

Minimality attacks first discussed in [98] primarily focus on k-anonymity schemes on tabular data. As stated in [98], in order to preserve the utility of published data, k-anonymity mechanisms all have the same underlying principle of minimizing the distortion to the original data. In other words, k-anonymity algorithms only distort the input data when necessary. Sometimes, this information can be used by adversaries to launch minimality attacks on published data to infer sensitive information of individuals and jeopardize privacy. In the following, we discuss the minimality notions defined in our k-degree anonymity algorithm, and show that these notions will not lead to minimality attacks.

The first minimality notion used in our algorithm is minimizing the distortion on the degree sequence of the input network (i.e. minimizing the number of edges modified). We use this notion to generate an objective degree sequence (i.e. the degree

User Identity	Degree
Alice	4
Bob	3
Tom	3
Michael	3
Grace	2
Andrew	2
Jamie	2
Anne	1

(a) Attacker's worst-case background knowledge

Vertex ID	Degree
c	4
d	4
g	3
f	3
a	2
b	2
e	2
h	2

(b) Vertex degree of the 2-degree anonymized network

User Identity	Mapped Vertex ID	Probability
Alice	c, d	$1 \times 1/2 = 1/2$
Bob Tom Michael	c, d	$1/3 \times 1/2 = 1/6$
	g, f	$2/3 \times 1/2 = 1/3$
Grace Andrew Jamie	a, b, e, h	$1 \times 1/4 = 1/4$
Anne	a, b, e, h	$1 \times 1/4 = 1/4$

(c) Attacker's inference probability

Figure 5.5: Minimality attack: attacker's worst-case background knowledge, vertex degree of published network, and inference probability.

sequence of the published network). We assume that in the worst case, an adversary has background knowledge of both the identity and degree of every individual vertex of a social network. Figure 5.5a shows the adversary's worst-case background knowledge of the social network depicted in Figure 5.1a, and Figure 5.5b lists the degree of every vertex in the published 2-degree anonymized network in Figure 5.1c. The adversary's goal is to map the real identities of the individuals to the vertices in the published network. With the minimality notion, the adversary reasons there must be only one vertex of degree 3 and one vertex of degree 1 having their degrees increased, and the degrees of other vertices remain. Therefore, the adversary infers that $Alice$ with degree 4 must be mapped to a vertex also with degree 4 in the published network. Since there are two vertices c and d with degree 4, without any extra information, the probabilities of $Alice$ being mapped to c and d are equal (i.e. $\frac{1}{2}$). However, Bob with degree 3 can be mapped to a vertex either with degree 4 or degree 3. Because there are totally three individuals with degree 3 in the original social network, the probability of Bob being the one that has degree increased is $\frac{1}{3}$, and the probability that his degree is unchanged is $\frac{2}{3}$. Therefore, the probability of Bob being mapped to vertex c or d is $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$, and the probability of him being mapped to vertex g or f is $\frac{2}{3} \times \frac{1}{2} = \frac{1}{3}$. Similarly, the mapping probability of every individual is calculated as shown in Figure 5.5c. Lemma 3 proves that the mapping probability that an adversary can calculate based on this minimality notion will not exceed $\frac{1}{k}$.

Lemma 3 *Given a published k-degree anonymized social network which is constructed by the minimality notion of minimizing the distortion on the degree sequence, we assume that in the worst case, an attacker knows the real identity and degree of every individual in the social network, and she is also aware of the minimality notion. The attacker cannot map an individual to any vertex with probability larger than $\frac{1}{k}$.*

Proof 3 *We denote the inferred mapping probability of an individual x to a vertex v with degree d in the published network as p_v^x. $p_v^x = p_1 \times p_2$ where p_1 is the probability that x is mapped to a vertex with degree d, and p_2 is the probability that among all the vertices with degree d, x is mapped to v. In the worst case, with the minimality notion, the attacker can infer p_1 with the maximum probability of 1. According to the k-anonymity condition, there are at least k vertices in the published network having degree d. Without any extra information, the attacker can only infer the mapping from x to these vertices with equal probability. Therefore, the maximum value of p_2 is $\frac{1}{k}$. Therefore, $p_v^x \leq 1 \times \frac{1}{k} = \frac{1}{k}$.*

The second minimality notion used in our algorithm is modifying a network to achieve k-anonymity by adding and deleting edges which make the smallest utility changes evaluated by the community-based utility model. For an attacker to use this notion to infer the details of the edge modification (i.e., anonymization) process then possibly to further infer user identities, she has to know the utility loss caused by every possible edge operation. However, in order to calculate utility loss, she has to know the community model based on which the utility measure is defined. Without the exact community model, the attacker cannot utilize this minimality notion to make inferences. Therefore, in order to prevent attackers from utilizing the second minimality notion, when publishing an anonymized network, the community model constructed should be kept secret. Moreover, revealing the community model will also provide attackers extra information about the original network besides vertex degree, which is against the worst-case assumption of our privacy scheme that attackers only know the degree information of vertices. This further explains that the community model should not be disclosed. However, we do not mind if attackers know the algorithm used to construct the community model. This is because attackers, based on the construction algorithm and vertex degree information, cannot reconstruct the community model.

We need to point out that the analysis above is based on the assumption that attackers only have background knowledge of vertex degree. If attackers have some extra information about the edges among certain individuals, they may be able to infer the identities of some vertices with higher probability. For instance, in the example above, besides the vertex degree information, the attacker knows that $Alice$ is connected with $Grace$ and $Andrew$ in the social network, she can further infer that vertex c is $Alice$. Because c connects with more than two vertices which can potentially be $Grace$ and $Andrew$ (i.e., a, b, and h) but the other candidate of $Alice$ (i.e., vertex d) does not. Therefore, privacy is compromised. We reserve this topic as an interesting problem to be studied in the future.

5.5.2 Extendability to Other K-Anonymity Schemes

As mentioned before, the proposed high-utility k-anonymization framework is general, and it is applicable to other k-anonymity schemes. In the following, we briefly explain how to extend the framework (as defined in Algorithm 4) to support k-neighborhood anonymity. Due to the focus of this work, we only present how to estimate local structure information and how to generate candidate edge operations in the context of k-neighborhood anonymity.

A k-neighborhood anonymized network $G^*(V^*, E^*)$ requires that for every vertex $v \in V^*$, there are at least $k-1$ other vertices having isomorphic neighborhoods as v. The neighborhood $NB(v)$ of v is a subgraph containing all the vertices within certain distance d to v and the edges among these vertices. d decides the size of the neighborhood. To simplify the discussion, we only consider neighborhoods within a distance of 1 (i.e., d=1), which is a common practice in other work [107]. Unless otherwise noted, the term "neighborhood" only refers to a neighborhood with $d = 1$.

The local structure information used by k-neighborhood anonymization is the neighborhood subgraph of every vertex. Like using the degree sequence to capture the local degree information of a network in k-degree anonymization, we use the neighborhood set NS to record the local neighborhood $NB(v_i)$ of every vertex v_i of a network, and we try to generate the neighborhood set $NS^{\triangle}$ of the "nearest" anonymized network. Again, for a given network G, its "nearest" anonymized network $G^{\triangle}$ refers to an anonymized network that satisfies k-neighborhood anonymity and is generated via the smallest number of edge operations. As generating $G^{\triangle}$ is not always possible, we use the neighborhood set $NS^{\triangle}$ of $G^{\triangle}$ as the estimation of the local structure information of $G^{\triangle}$.

Given G and the privacy parameter k, $NS^{\triangle}$ has the following properties. First, $NS^{\triangle}$ has the same cardinality as NS. Second, $NS^{\triangle}$ is k-anonymized, meaning that for each $NB^{\triangle}(v_i) \in NS^{\triangle}$, it has at least $k-1$ other isomorphic subgraphs in $NS^{\triangle}$. Third, the sum of the graph edit distance from every $NB^{\triangle}(v_i)$ to its corresponding $NB(v_i)$ is minimized (i.e., the number of edge operations needed to change G to $G^{\triangle}$ is minimized). Again, take the network shown in Figure 5.1a as an example. Assume $k = 2$, the neighborhood $NS(v_i)$ and its corresponding target neighborhood $NS^{\triangle}(v_i)$ of every vertex v_i are depicted in Fig. 5.6. We group the vertices having isomorphic neighborhoods in the target anonymized network into one *equivalent group* EG, i.e., $\forall v_i, v_j \in EG$, $NS^{\triangle}(v_i) = NS^{\triangle}(v_j)$. As shown in the figure, $\{a, b\}$

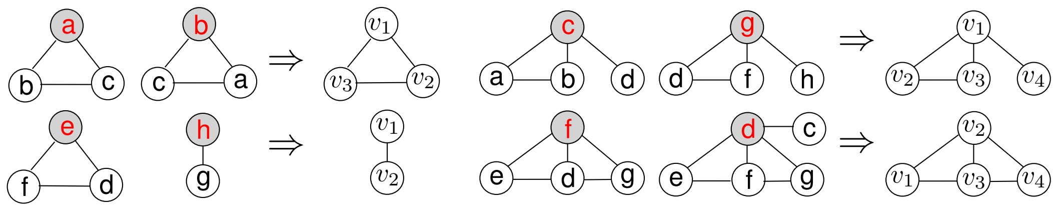
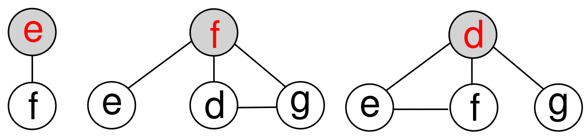


Figure 5.6: Example of the vertices' neighborhoods and estimated target neighborhoods.

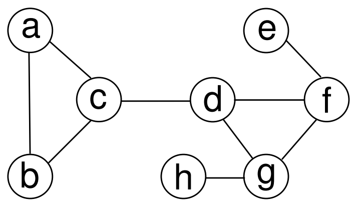
is an equivalent group, so as $\{c, g\}$, $\{e, h\}$, and $\{d, f\}$.

After obtaining $NS^{\triangle}$, we need to find candidate operations that can direct the current network to a k-neighborhood anonymized network having the target vertex neighborhoods $NS^{\triangle}$. In order to do this, we first identify the vertices whose neighborhoods are not the same as their target neighborhoods and store them in a set VS. And then, for each of these vertices $v_i \in VS$, all the viable edge operations that can nudge $NB(v_i)$ towards $NB^{\triangle}(v_i)$ are added into a candidate operation set. Continue the previous example. We find that, to change their current neighborhoods to the target neighborhoods, both vertex e and vertex d need to delete one edge (vertex) from their neighborhoods respectively. Consequently, the edge deletion operation $del(e, d)$ qualifies as a candidate operation. Given the set of all candidate operations, we calculate the utility loss caused by each of them and perform the operation with the smallest utility loss. Then, we update VS and regenerate the candidate operation set.

Obviously, a candidate edge operation serves at least one vertex v_i to convert the neighborhood of v_i to its target neighborhood $NS^{\triangle}(v_i)$. For example, operation $del(e, d)$ serves vertex e and vertex d. However, we want to highlight that an edge operation may affect the neighborhoods of some vertices which it does not meant to affect, hence the target neighborhoods of these vertices may no long be applicable after the edge operation is performed. For instance, $del(e, d)$ will also affect the neighborhood of f. Note that this issue does not exist in k-degree anonymity as a given edge operation only changes the degrees of the two associated vertices. In order to make the target neighborhoods consistent in an iteration of the algorithm, we identify all the vertices whose neighborhoods are not supposed to be affected



(a) $del(e, d)$ changes $NB(d)$, $NB(e)$, $NB(f)$.



(b) Example output.

Figure 5.7: Example of 2-neighborhood anonymization

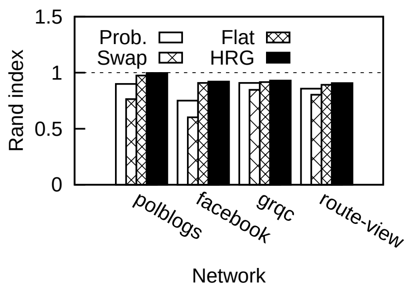
but are changed, as well as the vertices in the same equivalent groups as them, and exclude them from the updated set VS when we regenerate candidate operations. The process repeats until no candidate operations are generated. If the obtained graph G^* is k-anonymized, the algorithm stops. If not, we regenerate the target $NS^{\triangle}$ based on the current G^*. In this step, the previously excluded vertices are considered, and their new target neighborhoods are regenerated. To ensure the convergence of the algorithm, we are slightly biased towards choosing edge insertion operations among all candidate operations.

Continue the above example. Since there is only one candidate operation $del(e, d)$, we have no other choices but execute it. All the neighborhoods affected by this operation are illustrated in Figure 5.7a including the neighborhoods of vertices e, d, and f. As this edge deletion operation is to serve vertices e and d, the vertex that it does not meant to affect but is actually affected is f. Consequently, all the vertices that are within the same equivalent group as f (i.e., vertex d) are excluded when we regenerate candidate edge operations after performing $del(e, d)$. However, we find that after performing this operation, no candidate operations can be found. As the network is already 2-anonymized, the algorithm completes. The generated 2-neighborhood anonymized network is shown in Figurei 5.7b.

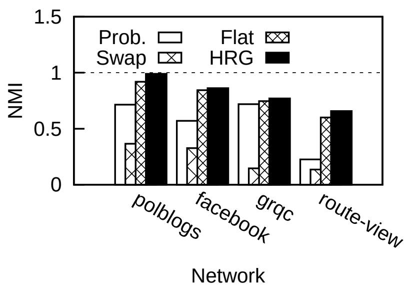
5.6 Experimental Evaluation

In this section, we conduct comprehensive experiments to demonstrate the advantages of the proposed high-utility k-anonymization algorithm. We focus on k-degree anonymity and implement two anonymization algorithms based on the flat community model and the hierarchical community model, respectively referred to as Flat and HRG. We also implement two existing k-degree anonymization algorithms proposed in [59] as competitors. Note that [59] is the only work in the literature focusing on k-degree anonymity, and the algorithms are designed to preserve network utility by minimizing the number of modified edges. The first *probing method* only considers edge addition operations. It first constructs a target k-anonymized degree sequence for the input network based on adding the smallest number of edges then inserts edges to the network to form a k-anonymized network with the desired target degree sequence. While the second *greedy swap method* considers both edge addition and deletion operations. It first derives a target k-anonymized degree sequence of the input network based on both adding and deleting edges then constructs a new network with the target degree sequence. In order to make the new network have a similar edge set to the input network (i.e., minimize the number of edges modified), it performs edge swap operations on the new network which keep the degree sequence intact and maximize the intersection of the edge sets of the new network and the input network. We refer these two algorithms to as Prob. and Swap respectively.

We use *utility* and *running time* as the main performance metrics. Our utility model is designed based on social network community structure, and it is expected to preserve social network community structure. Therefore, we evaluate the utility of a k-anonymized network mainly based on how well it preserves the community structure of the original network. Community is an important social network property, and it is useful to many applications. For example, identifying communities of customers with similar interests can help online retailers to set up recommendation systems to enhance business opportunities [81], and communities of a large network can be used to create data structures to efficiently store the network and



(a) Rand index



(b) NMI

Figure 5.8: Community utility loss caused by anonymization ($k = 10$).

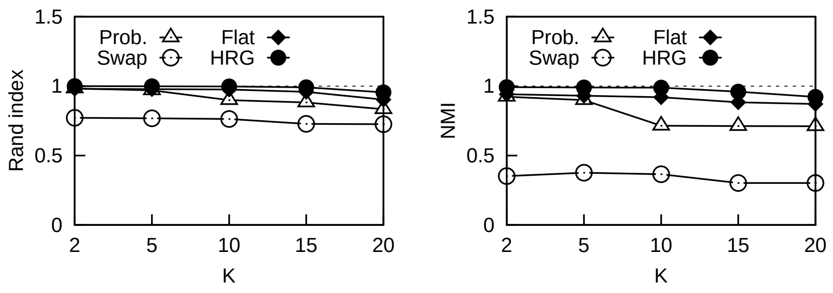
handle navigational queries [4].

We use the four real-world networks introduced in Section 4.3 in the experiments including **polblogs**, **facebook**, **grqc**, and **route-view**. Our experiments run on a Linux server with 2.67GHz CPU and 128G RAM.

5.6.1 Community Utility

In this section, we evaluate the performance of the algorithms on preserving community utility. Similar to Section 4.6.3, we run Clauset and Newman's classic greedy community detection algorithm [22] on networks and use the *Rand index* and *normalized mutual information (NMI)* [30] metrics to compare the communities generated on an anonymized network with those generated on the corresponding original network. The details of the two metrics were introduced in Section 4.6.3. That the values of these two metrics are close to 1 indicates that the community partitions are very similar.

Figure 5.8 presents the results on the four tested networks. In the experiments, we set the privacy parameter k at 10. We find that the Rand index values and NMI values of our Flat and HRG algorithms are all higher than those of the existing Prob. and Swap algorithms. The Rand index values of our algorithms are all higher than 0.9, and the NMI values of our algorithms are all larger than 0.6 on the four tested networks. These indicate that our algorithms can generate networks with similar community structure to the original networks. However, in many cases the exist-



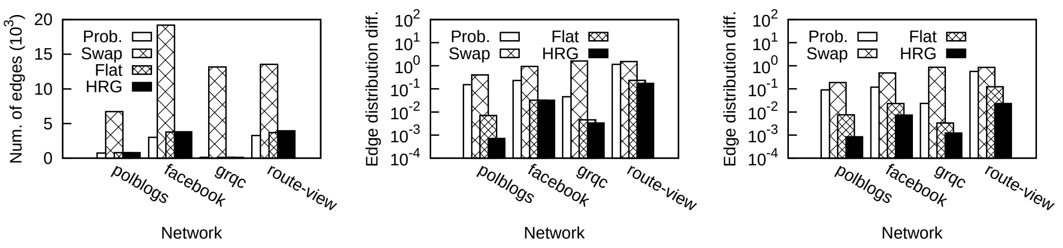
(a) Rand index (b) NMI

Figure 5.9: Community utility changes with k on polblogs network.

ing algorithms cannot preserve network community structure well, for example, the Prob. and Swap algorithms on the **route-view** network. Even though the Rand index values of these algorithms are high, they are more likely caused by the pairs of vertices which are not partitioned in the same communities as the corresponding NMI values are very low. Therefore, our algorithms have great advantages over the existing algorithms for preserving community utility.

We also find the HRG algorithm that runs based on the hierarchical community model (i.e., HRG) generally generates networks with better community utility than the Flat algorithm which is based on the flat community model. The reason is that the complex HRG model captures more detailed information of network community structure, and it is more sensitive to small numbers of edge modifications. Therefore, the anonymization algorithm which evaluates utility loss based on this model is more accurate, thus generating networks that better represent the features of the original networks.

We also test the utility performance of the four algorithms with increasing k and present the results in Figure 5.9. We only present the results on the **polblogs** network, and the results on the other networks are omitted as they have similar trends. We find that the utility performance of the four algorithms degrades as k increases. It is not surprising because in order to achieve higher levels of privacy with larger k, we need to make more distortions to networks. In the figure, we find our algorithms outperform the two existing algorithms under all the k settings.



(a) Utility benchmark based on the number of modified edges. (b) Utility benchmark based on flat community structure. (c) Utility benchmark based on hierarchical community structure.

Figure 5.10: Utility benchmarks ($k = 10$).

5.6.2 Utility Benchmark

From the above experiments we find the four algorithms generate networks with different community utility. The HRG algorithm generates networks of the best utility followed by the Flat algorithm and then the Prob. algorithm. The Swap algorithm generates networks of the worst utility. In this section, we investigate whether the different utility benchmarks can reflect the real utility of the networks generated by the four algorithms. There are three utility benchmarks used. The existing algorithms measure utility using the number of modified edges. We propose two novel utility benchmarks which evaluate the edge distribution changes within and between a fixed community partition of a network caused by anonymization. One of them is designed based on a flat community model, and the other is designed based on a hierarchical community model.

We calculate the utility of the networks generated by the four algorithms based on the three utility benchmarks respectively and report the results in Figure 5.10. Figure 5.10a shows the numbers of modified edges of different algorithms on the four tested networks. According to this utility benchmark, the networks generated by the Prob. algorithm have the best utility as they have the smallest numbers of modified edges, the networks generated by the Flat algorithm have higher utility than those generated by the HRG algorithm, and the networks generated by the Swap algorithm have the worst utility. This is not consistent with our experimental results on real community utility. Therefore, the simple utility benchmark used by the existing algorithms cannot reflect the real community utility of anonymized networks.

Figure 5.10b and Figure 5.10c show the edge distribution changes caused by different algorithms based on the flat and hierarchical community models respectively. According to these two utility benchmarks, the four algorithms sorted in descending order of their utility are HRG, Flat, Prob., and Swap. This is consistent with our experimental results on real community utility. Therefore, both of the two utility benchmarks we proposed can reflect the real community utility of the networks generated by different algorithms.

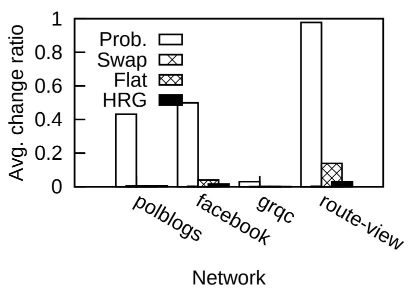
5.6.3 Other Utility

The experimental results above have demonstrated that our high-utility k-anonymization algorithms can preserve network community utility very well. As community structure is closely related to many important network structural properties, we expect our algorithms can preserve other types of network utility as well. Therefore, in this section, we test the utility performance of the algorithms based on two other types of network utility including centrality utility and shortest path utility.

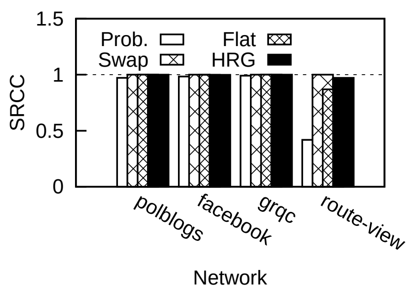
Centrality Utility

We first evaluate the performance of the algorithms on centrality utility. Similar to Section 4.6 in Chapter 4, we perform the evaluation based on four basic centrality metrics including degree centrality, betweenness centrality, closeness centrality, and PageRank centrality, and we also use the *average change ratio of centrality scores* to evaluate how well the anonymization algorithms preserve raw centrality scores of vertices and use *Spearman's rank correlation coefficient (SRCC)* to evaluate how well the anonymization algorithms preserve the ranks of vertices based on centrality.

Figure 5.11-5.14 provide the experimental results. In the experiments, we set the privacy parameter k at 10. We find from the figures that for three centrality metrics (i.e., closeness, betweenness, PageRank) our algorithms (i.e., Flat and HRG) outperform the existing algorithms (i.e., Prob. and Swap) as our algorithms result in much smaller average change ratios on the raw centrality scores and higher SRCC

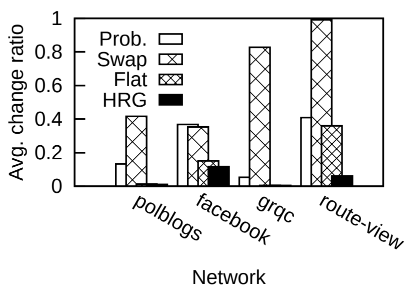


(a) Average change ratio

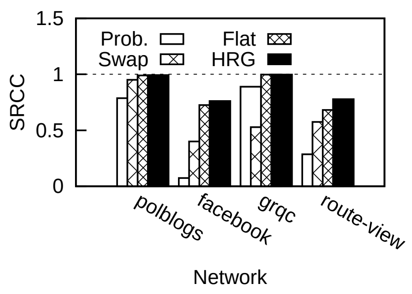


(b) SRCC

Figure 5.11: Degree centrality utility loss caused by anonymization ($k = 10$).



(a) Average change ratio

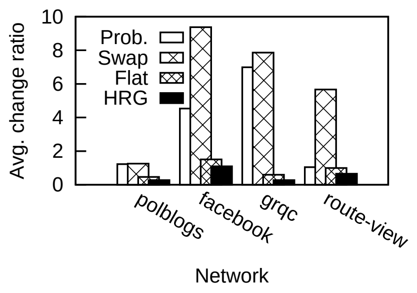


(b) SRCC

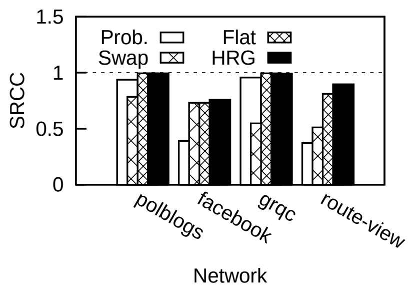
Figure 5.12: Closeness centrality utility loss caused by anonymization ($k = 10$).

values. However, for degree centrality, the Swap algorithm causes the lowest average change ratio and the highest SRCC values. It is because the Swap algorithm is developed based on edge swap operations which intend to keep vertex degree intact. Therefore, this algorithm makes the smallest distortion on vertex degree. From Figure 5.11, we observe that even our algorithms do not perform edge swap operations, they cause very small changes to vertex degree. The average change ratios on the four tested networks are all smaller than 0.13, and the SRCC values are all larger than 0.87.

We also observe from the figures that in most of the cases our algorithms cause very small distortions to the raw centrality scores and the centrality ranks. The average change ratios of our algorithms are smaller than 0.3 and the SRCC values are larger than 0.7. However, for the betweenness centrality, we find that our algorithms cause relatively large average change ratios, which are still much smaller than those caused by the existing algorithms. This is mainly because of some outlier vertices

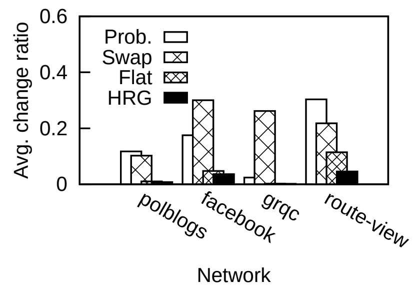


(a) Average change ratio

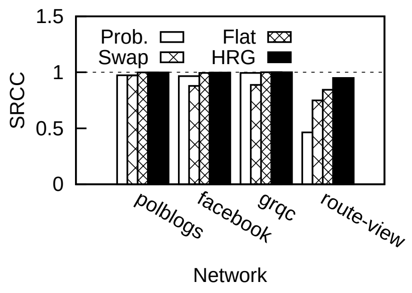


(b) SRCC

Figure 5.13: Betweenness centrality utility loss caused by anonymization ($k = 10$).



(a) Average change ratio



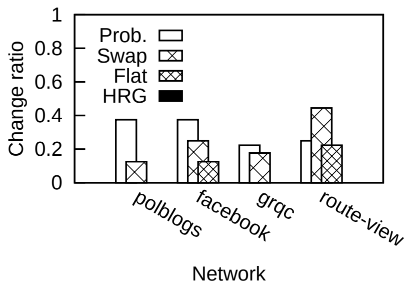
(b) SRCC

Figure 5.14: PageRank centrality utility loss caused by anonymization ($k = 10$).

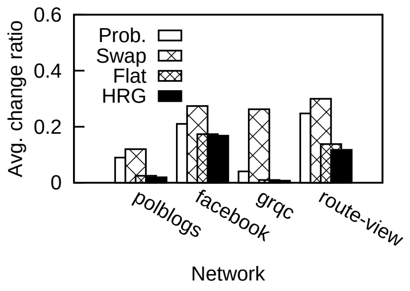
which originally have very small betweenness scores. Small increases in the betweenness scores of these vertices during the anonymization process can result in very large change ratios. Despite the large average change ratios, the SRCC values of our algorithms are always very high, which means our algorithms can preserve the ranks of vertices based on betweenness centrality well. Finally, we find the HRG algorithm generates networks with better centrality utility than the Flat algorithm.

Shortest Path Utility

In this section, we evaluate the shortest path utility performance of the algorithms. We use the change ratio of diameter which is the longest shortest path length in a network and the average change ratio of the shortest path lengths between all pairs of vertices as the metrics. Figure 5.15 shows the results. We find that our algorithms cause smaller changes to the diameters and the shortest path lengths of the four tested networks compared with the existing algorithms. Especially, the HRG

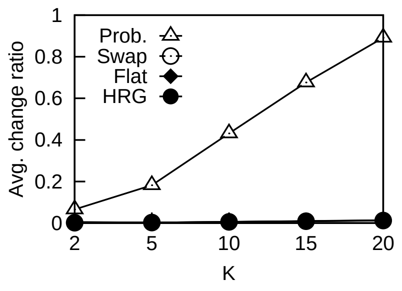


(a) Diameter change ratio

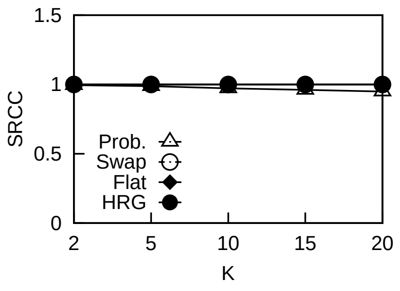


(b) Average change ratio of shortest path lengths

Figure 5.15: Shortest path utility loss caused by anonymization ($k = 10$).



(a) Average change ratio



(b) SRCC

Figure 5.16: Degree centrality utility changes with k on polblogs network.

algorithm causes no changes to the diameters of the four networks, and the average change ratios of shortest path lengths of our algorithms are always smaller than 0.2. These findings demonstrate that our algorithms can preserve shortest path length information very well. We also observe that the HRG algorithm performs slightly better than the Flat algorithm.

Other Utility Performance V.S. K

We also test the centrality utility and shortest path utility performance of the four algorithms with increasing k and present the results in Figure 5.16-5.20. We only present the results on the **polblogs** network, and the results of similar trends on the other networks are omitted. Similar to the findings in Section 5.6.1, we find that generally the utility performance of the four algorithms degrades as k increases. In the figures, we find our algorithms outperform the two existing algorithms in all the tested utility metrics except degree centrality, under all the k settings. The

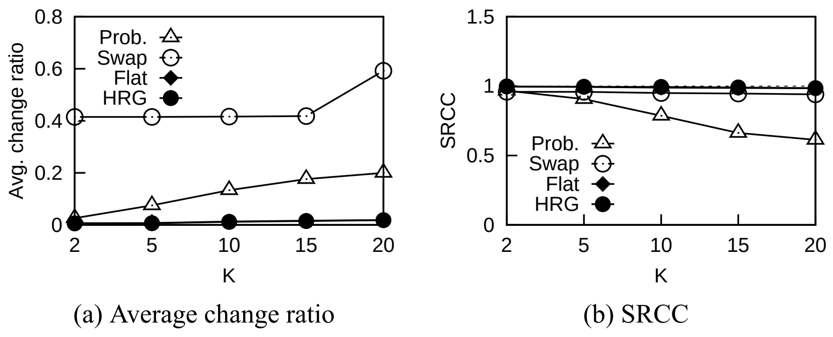


Figure 5.17: Closeness centrality utility changes with k on polblogs network.

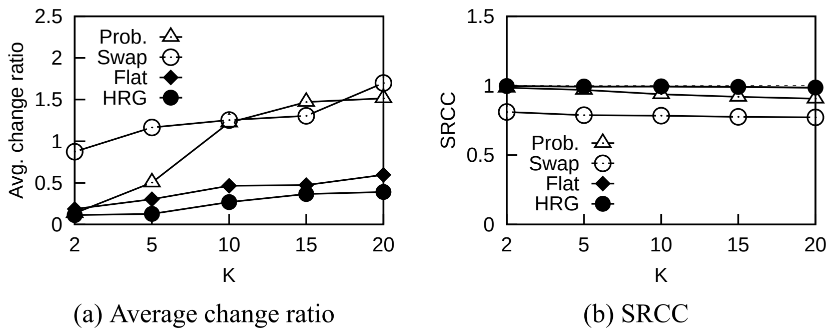
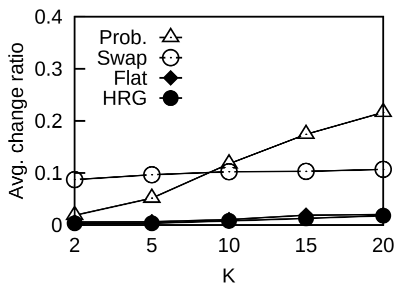


Figure 5.18: Betweenness centrality utility changes with k on polblogs network.

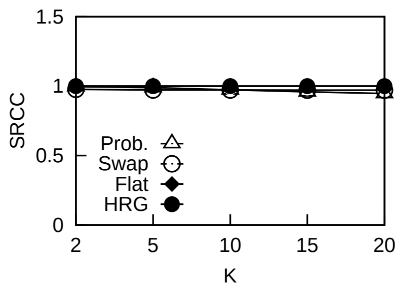
Swap algorithm performs the best on degree centrality, but the performance of our algorithms is comparable.

5.6.4 Running Time

In this section, we evaluate the time performance of the algorithms and report the results in Figure 5.21. Figure 5.21a shows the running time of the four algorithms on the four tested networks as k is set at 10. From the figure, we find the Prob. algorithm achieves the best time performance. The running time of the Flat algorithm is slightly longer but comparable to that of the Prob. algorithm. Sometimes the running time of the Swap algorithm and the HRG algorithm can be much longer than that of the other two algorithms (e.g., on the **polblogs** and **facebook** networks). The long running time of the Swap algorithm is because of the large number of possible edge swap operations, and the long running time of the HRG algorithm is mainly because of the the large number of candidate operations.

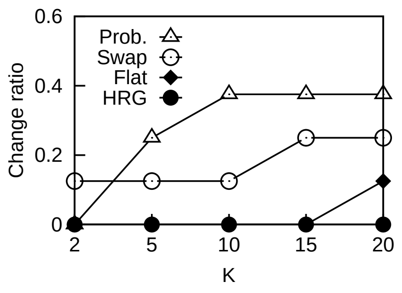


(a) Average change ratio

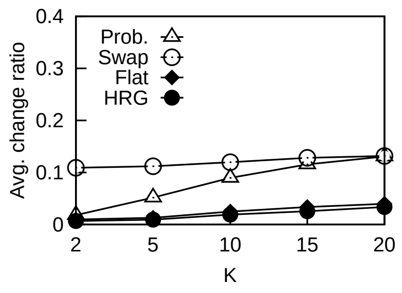


(b) SRCC

Figure 5.19: PageRank centrality utility changes with k on polblogs network.



(a) Diameter change ratio

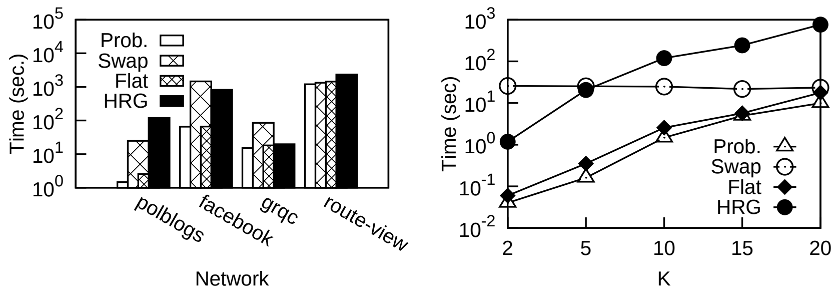


(b) Average change ratio of shortest path lengths

Figure 5.20: Shortest path utility changes with k on polblogs network.

Figure 5.21b reports the running time changes of the four algorithms on the **polblogs** network with increasing k. The results of similar trends on the other networks are omitted to avoid redundancy. From the figure, we find that the running time of the Prob., Flat, and HRG algorithms grows fast as k increases. This is because in order to achieve anonymity with larger k, usually more edges need to be modified. While the running time of Swap is not sensitive to changing k, but it always remains high. This is because the Swap method always needs to consider a large number of possible edge swap operations even with small k. Generally, the time performance of the Flat algorithm is good and comparable to that of the Prob. algorithm.

To sum up, the comprehensive experimental results clearly demonstrate that the high-utility k-anonymization algorithms (i.e. Flat and HRG) which are designed based on the community structure-based utility models can very well preserve network community utility as well as many other types of network utility (i.e., centrality utility and shortest path utility). The advantages of our algorithms in utility



(a) Running time on four tested networks ($k = 10$).
(b) Running time V.S. k on polblogs network.

Figure 5.21: Time performance.

are significant compared with the existing k-anonymization algorithms which control utility simply based on the number of edge modifications. The HRG algorithm has better utility performance than the Flat algorithm, however, incurs much more expensive time cost. Therefore, we recommend that in the cases where time performance is critical, people employ the Flat algorithm to achieve good utility with short running time, while the HRG algorithm is recommended when time performance is not the main concern to achieve the best utility.

5.7 Summary

In this chapter, we have discovered that existing k-anonymization approaches on social networks provide good protection for user identity privacy. However, with a naive utility model which evaluates utility loss based only on the number of edges modified during an anonymization process, they fail to generate anonymized networks with high utility. To solve this problem, we have proposed a high-utility social network anonymization framework to achieve good privacy protection with low utility loss. In our approach, we use a novel utility measure which directly evaluates the changes that an anonymization algorithm makes to network community structure. Our utility measure can work with both flat community structure and hierarchical community structure. We have performed experiments on real-world datasets and shown that our approach outperforms existing approaches in utility.

Chapter 6

Conclusion

In this chapter, we summarize the dissertation by reviewing the social network data preparation problem and the work we have accomplished. Then, we discuss some promising directions for future work.

6.1 Dissertation Summary

Raw social network data have many undesirable features such as vast, noisy, distributed, and sensitive, which challenge data analysis and mining research. In order to enhance data quality and usability and enable successful data analysis and mining tasks, data preparation is indispensable. Social network data preparation is a complex process which consists of data collection, data cleaning, data reduction, and data conversion steps. Each of these steps deals with various distinctive challenging problems. In this dissertation, we study three important problems in social network data preparation. The first one is the sampling problem of social network content data in the data collection step, and the second and third ones are the privacy protection issue and the utility issue respectively in the data conversion step.

In Chapter 3, we perform an exploratory study of user-generated Twitter data samples obtained from the Twitter stream API, which is the data source for a variety of research and commercial applications. We compare the sample data with the corresponding complete dataset in many different aspects including basic tweet statistics, content representativeness, user coverage, and user interactions in order to understand the nature of the sample data, their bias, and how well they represent the complete data stream. We make some interesting observations. For example, we find that the Twitter data samples with a low sampling rate of 0.96% are good enough for studying general Twitter user activity patterns, analyzing certain tweet contents e.g., text terms and url domains; however, for some other tweet contents such as hashtags, larger samples are preferred in order to obtain accurate results. We also find that the sample data are biased towards active users and miss lots of user interaction information, but extending the sampling period and increasing the sampling rate both help to improve the coverage of the user base and the accuracy of the interaction-based user popularity estimation. Our findings provide insights about the sample data obtained from the Twitter stream API and provide incentives for people to use or not to use them for their research.

In Chapter 4, we discover a new type of privacy attacks on social networks called connection fingerprint (CFP) attacks, in which attackers utilize the connection information of an anonymized user to some known public users in a social network to re-identify the user and compromise identity privacy. We formally analyze the risk of CFP attacks on real-world social networks and demonstrate these attacks can seriously damage user identity privacy. We propose two efficient k-anonymity-based network conversion algorithms to protect social networks against CFP attacks and preserve the utility of converted networks. One algorithm is based on adding dummy vertices. It can resist powerful attackers with the connection information of a user with the public users within n hops ($n \geq 1$) and can preserve the centrality utility of public users. The other algorithm is based on edge modifications. It is only able to resist attackers with the connection information of a user with the public users within 1 hop but preserve a rich spectrum of network utility.

In Chapter 5, we find that existing k-anonymity based algorithms convert networks to protect privacy by modifying edges, and they preserve utility by minimizing the number of edges modified. We find this simple utility model cannot reflect real utility changes of networks with complex structure. Thus, relying on it, existing k-anonymization algorithms cannot guarantee generating social networks with high utility. To improve utility performance, we propose a novel utility benchmark which directly evaluates the changes that a network conversion algorithm makes to network community structure. In addition, we design a general network conversion algorithm framework with the new utility benchmark which can be used with various k-anonymity schemes, e.g., k-degree anonymity and k-neighborhood anonymity. Experimental results on real-world networks demonstrate that the community structure-based utility benchmark can better reflect real community utility of converted social networks compared with the existing naive utility model. Our network conversion algorithms using the new utility benchmark significantly outperform existing algorithms in utility.

To summarize, in this dissertation we consider three important problems in social network data preparation including sample representativeness, privacy protection, and utility preserving. Our study motivates people to use sampled social network data more carefully and provides people methods to better preserve the privacy and utility of social network data. It benefits data mining and analysis tasks by providing more reliable social network data.

6.2 Future Work

We conclude this dissertation by outlining several interesting and promising research problems for further work.

In the study of sampled Twitter data, we realize that many Twitter data analysis applications use the Twitter stream API to obtain the same sampled Twitter data for various analysis tasks. However, the sample data are good for certain analysis tasks such as user activity pattern characterization and sentiment analysis, but may not be suitable for some other tasks such as user network analysis. We think it is meaningful to study task-specific sampling techniques to generate tweet streams that are suitable for different analysis applications. Note that the sample datasets obtained from the Twitter Stream API are sampled sets of tweets. Although we can extract user ids and interaction information from tweet data, tweets do not contain the "follower" and "friend" relationship information declared by Twitter users, which is important metadata. It is an interesting open question whether these relationships can be inferred from tweets and the user interaction information that is available in sample data streams.

In the study of network conversion algorithms against CFP attacks, we find that in some cases, our algorithms have different utility performance on different networks. We think it is interesting to explore the relationship between network structure and utility performance in future work. Another direction of improvement is the utility performance of the algorithms on networks with a large number of public users or when the privacy parameter k is large.

In the study of utility benchmarks in social network privacy protection, we propose an utility benchmark based on network community structure and demonstrate its advantage for reflecting real network community utility. It is interesting to study other practical utility benchmarks which can reflect other different types of network utility.

Bibliography

[1] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD'05, pages 36--43, 2005.

[2] Charu C. Aggarwal. *Social Network Data Analytics*. Springer Publishing Company, Incorporated, 2011.

[3] Charu C. Aggarwal and Philip S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer Publishing Company, Incorporated, 2008.

[4] Rakesh Agrawal and H. V. Jagadish. Algorithms for searching massive graphs. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):225--238, 1994.

[5] Faraz Ahmed, Rong Jin, and Alex X. Liu. A random matrix approach to differential privacy and structure preserved social network graph publishing. *CoRR*, abs/1307.0475, 2013.

[6] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th International Conference on World Wide Web*, WWW'07, pages 835--844, 2007.

[7] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. 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, pages 181--190, 2007.

[8] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Everyone's an influencer: Quantifying influence on twitter. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 65--74, 2011.

[9] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC'09, pages 49--62, 2009.

[10] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *VLDB Endowment*, 2(1):766--777, 2009.

[11] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Prediction promotes privacy in dynamic social networks. In *Proceedings of the 3rd Wonference on Online Social Networks*, WOSN'10, pages 6--6, 2010.

[12] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computer Science*, 2(1):1--8, 2011.

[13] Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology*, 2(3):22:1--22:37, 2011.

[14] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97--130, 2001.

[15] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM SIGMOD*

International Conference on Management of Data, SIGMOD'99, pages 479--490, 1999.

[16] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC'07, pages 1--14, 2007.

[17] Shixi Chen and Shuigeng Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD'13, pages 653--664, 2013.

[18] James Cheng, Ada Wai-chee Fu, and Jia Liu. K-isomorphism: Privacy preserving network publication against structural attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD'10, pages 459--470, 2010.

[19] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'09, pages 219--228, 2009.

[20] Aaron Clauset, Cristopher Moore, and M E J Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98--101, 2008.

[21] Aaron Clauset, Cristopher Moore, and Mark E. J. Newman. Structural inference of hierarchies in networks. In *Proceedings of the 2006 Conference on Statistical Network Analysis*, ICML'06, pages 1--13, 2007.

[22] Aaron Clauset, M. E. J. Newman, , and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, pages 1-- 6, 2004.

[23] Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. *ACM Transactions on Data Privacy*, 6(2):161--183, 2013.

[24] Brittany Darwell. Facebook platform now encompasses more than 50m pages and 10m apps. WWW page, 2013.

[25] Xuan Ding, Lan Zhang, Zhiguo Wan, and Ming Gu. De-anonymizing dynamic social networks. In *Proceedings of the Global Communications Conference*, GLOBECOM'11, pages 1--6, 2011.

[26] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC'06, pages 265--284, 2006.

[27] Emarketer. Worldwide social network users: 2013 forecast and comparative estimates. WWW page, June 2013.

[28] Experian. Experian marketing services reveals 27 percent of time spent online is on social networking. WWW page, April 2013.

[29] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici. Link prediction in social networks using computationally efficient topological features. In *Proceedings of the IEEE 3rd International Conference on Social Computing*, SocialCom'11, pages 73--80, 2011.

[30] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 -- 174, 2010.

[31] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):14:1--14:53, 2010.

[32] Jun Gao, Jeffrey Xu Yu, Ruoming Jin, Jiashuai Zhou, Tengjiao Wang, and Dongqing Yang. Neighborhood-privacy protected shortest distance comput-

ing in cloud. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD'11, pages 409--420, 2011.

[33] Gabriel Ghinita, Yufei Tao, and Panos Kalnis. On the anonymization of sparse high-dimensional data. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, ICDE'08, pages 715--724, 2008.

[34] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC'07, pages 15--28, 2007.

[35] Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, pages 2498--2506, 2010.

[36] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 9th IEEE International Conference on Data Mining*, ICDM'09, pages 169--178, 2009.

[37] Michael Hay, Gerome Miklau, and David Jensen. Anonymizing social networks. Technical report, UMass Amberst, 2007.

[38] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *VLDB Endowment*, 1(1):102--114, 2008.

[39] H. S. Heaps. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, Incorporated, 1978.

[40] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. Discovering geographical topics in the twitter stream.

In *Proceedings of the 21st International Conference on World Wide Web*, WWW'12, pages 769--778, 2012.

[41] Bernardo A. Huberman, Daniel M. Romero, and Fang Wu. Social networks that matter: Twitter under the microscope. *First Monday*, 14(1), 2009.

[42] Mathias Humbert, Théophile Studer, Matthias Grossglauser, and Jean-Pierre Hubaux. Nowhere to hide: Navigating around privacy in online social networks. In *Proceedings of the 18th European Symposium on Research in Computer Security*, ESORICS'13, pages 682--699, 2013.

[43] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: Understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, WebKDD/SNA-KDD'07, pages 56--65, 2007.

[44] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *VLDB Endowment*, 4(11):1146--1157, 2011.

[45] Vishesh Karwa and Aleksandra B. Slavkovic. Differentially private synthetic graphs. *CoRR*, abs/1205.4697, 2012.

[46] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 457--476, 2013.

[47] Daniel Kifer and Johannes Gehrke. Injecting utility into anonymized datasets. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD'06, pages 217--228, 2006.

[48] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M. Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd Inter-*

national ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'09, pages 195--202, 2009.

[49] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *Proceedings of the 1st Workshop on Online Social Networks*, WOSN'08, pages 19--24, 2008.

[50] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J. H. Cui, and A. G. Percus. Reducing large internet topologies for faster simulations. In *Proceedings of the 4th IFIP-TC6 International Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems*, NETWORKING'05, pages 328--341, 2005.

[51] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'06, pages 611--617, 2006.

[52] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW'10, pages 591--600, 2010.

[53] David Lazer, Alex Pentland, Lada Adamic, Sinan Aral, Albert-Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. Social science: Computational social science. *Science*, 323(5915):721--723, 2009.

[54] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md. Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. Twitter trending topic classification. In *Proceedings of the IEEE 11th International Conference on Data Mining Workshops*, ICDMW'11, pages 251--258, 2011.

[55] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'06, pages 631--636, 2006.

[56] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[57] Tiancheng Li and Ninghui Li. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'09, pages 517--526, 2009.

[58] Kun Liu, Kamalika Das, Tyrone Grandison, and Hillol Kargupta. Privacy-preserving data analysis on graphs and social networks. In *Next Generation of Data Mining*, chapter 21, pages 419--437. 2008.

[59] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD'08, pages 93--106, 2008.

[60] Lian Liu, Jie Wang, Jinze Liu, and Jun Zhang. Privacy preservation in social networks with sensitive edge weights. In *Proceedings of the SIAM International Conference on Data Mining*, SDM'09, pages 954--965, 2009.

[61] Arun S. Maiya and Tanya Y. Berger-Wolf. Sampling community structure. In *Proceedings of the 19th International Conference on World Wide Web*, WWW'10, pages 701--710, 2010.

[62] Arun S. Maiya and Tanya Y. Berger-Wolf. Benefits of bias: towards better characterization of network sampling. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'11, pages 105--113, 2011.

[63] Hossein Maserrat and Jian Pei. Neighbor query friendly compression of social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'10, pages 533--542, 2010.

[64] Michael Mathioudakis and Nick Koudas. Twittermonitor: Trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD'10, pages 1155--1158, 2010.

[65] Julian Mcauley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data*, 8(1):4:1--4:28, 2014.

[66] Amin Milani Fard and Ke Wang. Neighborhood randomization for link privacy in social network analysis. *World Wide Web*, pages 1--24, 2013.

[67] Darakhshan Mir and Rebecca N. Wright. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT'12, pages 167--176, 2012.

[68] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC'07, pages 29--42, 2007.

[69] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, WSDM'10, pages 251--260, 2010.

[70] Fred Morstatter, J ürgen Pfeffer, Huan Liu, and Kathleen M Carley. Is the sample good enough? comparing data from twitter's streaming api with twit-

ter firehose. In *Proceedings of the 4th International Conference on Weblogs and Social Media*, ICWSM'10, 2013.

[71] J. L. Myers and A. D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum Associates, 2003.

[72] Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. Is it really about me?: Message content in social awareness streams. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW'10, pages 189--192, 2010.

[73] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004.

[74] B. Ohana and B. Tierney. Sentiment classification of reviews using sentiwordnet. In *Proceedings of the 9th IT & T Conference*, page 13, 2009.

[75] Olga Peled, Michael Fire, Lior Rokach, and Yuval Elovici. Entity matching in online social networks. In *Proceedings of the 2013 International Conference on Social Computing*, SOCIALCOM'13, pages 339--344, 2013.

[76] Alexei Pozdnoukhov and Christian Kaiser. Space-time dynamics of topics in streaming text. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, LBSN'11, pages 1--8, 2011.

[77] Vishal Bhatnagar Preeti Gupta. Data preprocessing for dynamic social network analysis. In *Data Mining in Dynamic Social Networks and Fuzzy Systems*. IGI Global, 2013.

[78] Elie Raad, Richard Chbeir, and Albert Dipanda. User profile matching in social networks. In *Proceedings of the 2010 13th International Conference on Network-Based Information Systems*, NBIS'10, pages 297--304, 2010.

[79] Carlos G. Rodríguez Penagos Rafael E. Banchs. Mining user-generated content for social research and other applications. In *Small and Medium Enterprises: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2013.

[80] Vibhor Rastogi, Dan Suciu, and Sungho Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd International Conference on Very Large Databases*, VLDB'07, pages 531--542, 2007.

[81] P. Krishna Reddy, Masaru Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *Proceedings of the 2nd International Workshop on Databases in Networked Information Systems*, DNIS'02, pages 188--200, 2002.

[82] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW'10, pages 851--860, 2010.

[83] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC'11, pages 81--98, 2011.

[84] Fortunato Santo. Community detection in graphs. *Physics Reports*, 486:75--174, 2010.

[85] Christian Sengstock and Michael Gertz. Latent geographic feature extraction from social media. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'12, pages 149--158, 2012.

[86] Craig Smith. By the numbers: 220 amazing twitter statistics. WWW page, 2014.

[87] Mustafa Sofean and Matthew Smith. A real-time architecture for detection of diseases using social networks: Design, implementation and evaluation. In *Proceedings of the 23rd ACM Conference on Hypertext and Social Media*, HT'12, pages 309--310, 2012.

[88] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC'10, pages 1--9, 2010.

[89] Xiaoxun Sun, Hua Wang, Jiuyong Li, and Jian Pei. Publishing anonymous survey rating data. *Data Mining Knowledge Discovery*, 23(3):379--406, 2011.

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

[91] Chih-Hua Tai, Peng-Jui Tseng, Philip S. Yu, and Ming-Syan Chen. Identity protection in sequential releases of dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):635--651, 2014.

[92] Enhua Tan, Lei Guo, Songqing Chen, Xiaodong Zhang, and Yihong (Eric) Zhao. Spammer behavior analysis and detection in user generated content on social networks. In *Proceedings of the IEEE 32nd International Conference on Distributed Computing Systems*, ICDCS'12, pages 305--314, 2012.

[93] Global Times. Media, govt, organizations get hooked on weibo: Report. WWW page, 2013.

[94] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *Advances in Knowledge Discovery and Data Mining*, pages 329--340. Springer, 2013.

[95] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[96] Sina Weibo. Maximize your brand power with weibo. WWW page, 2013.

[97] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP'10, pages 223--238, 2010.

[98] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB'07, pages 543--554, 2007.

[99] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, and Zhihui Wang. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT'10, pages 111--122, 2010.

[100] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'14, pages 911--920, 2014.

[101] Qian Xiao, Zhengkui Wang, and Kian-Lee Tan. Lora: Link obfuscation by randomization in graphs. In *Proceedings of the 8th VLDB International Conference on Secure Data Management*, SDM'11, pages 33--51, 2011.

[102] Xiaowei Ying and Xintao Wu. Randomizing social networks: A spectrum preserving approach. In *Proceedings of the SIAM International Conference on Data Mining*, SDM'08, pages 739--750, 2008.

[103] Sooyeon Yoon, Sungmin Lee, Soon-Hyung Yook, and Yup Kin. Statistical properties of sampled networks by random walk. *Physical Review E*, 75(4), 2007.

[104] Wang Yue and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *ACM Transactions on Data Privacy*, 6:127--145, 2013.

[105] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 338--349, 2011.

[106] Elena Zheleva and Lise Getoor. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th International Conference on World Wide Web*, WWW'09, pages 531--540, 2009.

[107] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, ICDE'08, pages 506--515, 2008.

[108] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations Newsletter*, 10:12--22, 2008.

[109] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *VLDB Endowment*, 2(1):718--729, 2009.

[110] L Zou, Lei Chen, and MT Özsu. K-automorphism: A general framework for privacy preserving network publication. *VLDB Endowment*, 2(1):946--957, 2009.