EFFECTIVE AND SECURE USE OF HUMAN RELATION NETWORKS

by

NA LI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2012

To my parents Xiande Li and Juan Wang,

my American parents Hollis and Virginia Lackey,

and my uncle and aunt Roy and Cathie Steele,

for their love, care and support all the time.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Sajal K. Das who is very helpful and supportive in my pursuit of my Ph.D. degree. In addition to his exceptional guidance on my research, I want to thank him for giving me the opportunity of choosing research topics composed of my dissertation. It has been a great pleasure working with him these five years. His breadth of knowledge and his enthusiasm for research have been inspiring me. I also appreciate him for offering all kinds of opportunities to me, including engaging me into the writing of NSF proposals, encouraging me to attend Summer School on Trustworthy Computing at Carnegie Mellon University in 2010 and Illinois Wireless Summer School at UIUC in 2009, and helping me apply for several scholarships including many travel grants for attending conferences (WoWMoM 2010, ICDCN 2012, PerCom 2009 and 2011) as well as the award of Lockheed Martin Outstanding Contribution to the CSE Department at UTA. All of these made my Ph.D. life really colorful and wonderful!

I would also like to express my very deep appreciation to all of my committee members, Prof. Gautam Das, Prof. Manfred Huber, Prof. Yonghe Liu, Prof. Qilian Liang (from EE Dept. at UTA) and Prof. Nan Zhang (from CS at GWU), for their technical guidance in my research and for taking time to serve in my dissertation committee. I enjoyed the classes taught by them, which helped me build up strong technical background. In particular, I am very grateful to Prof. Zhang for his years of research guidance in a long distance. Additionally, I want to thank Prof. Gautam Das and Prof. Zhang for their support and help in my job search.

I am also very grateful to all the CSE faculty members at UTA, and particularly, Professors Donggang Liu, Matthew Wright, Mohan Kumar, Bahram Khalili, Ramez Elmasri, Darin Brezeale, Gergely Zaruba (UTA), as well as the CSE staff members, Camille Costabile, Pam McBride, and Sherri Warwick, for their encouragements and kind support. My special thanks go to Prof. Donggang Liu and Prof. Wright for their years of leading in our group project from which I got funded as a research assistant; Prof. Kumar who introduced me to the field of opportunistic networks by his class which helped me build up my background in the area of mobile social networks and paved the way for the research topics that my dissertation covers; Prof. Khalili for his constantly encouraging and great help in my job search; Prof. Elmasri for offering me the opportunity of teaching CSE 1311-003, Introductory Programming for Engineers and Scientists, in fall 2011, which let me taste the flavor of being an instructor and accumulate teaching experience; Dr. Darin for his sharing of his teaching experience and lecture materials which made my first piece of teaching experience quite manageable; Prof. Zaruba for his kind support to attend IEEE PerCom 2009 and 2011 conferences. Additionally, I appreciate all of the tutors at UTA Writing Center for helping me improve my writing skills. Particularly, I would like to thank Jessi Urban, who is a very helpful and responsible tutor, for her great help and encouragement all the time.

Moveover, I would like to thank all CReWMaN members, and particularly, some alumni including Drs. Preetam Ghosh and Jing Wang, Pradip De and Nirmalya Roy for their help in my research and my future career plan. Also, I want to thank Dr. Mario Di Francesco for introducing me to the project of multimodal sensing in heterogeneous and multimedia wireless sensor networks, and also for his discussion and insightful comments to my research work. I appreciate the study environment in CReWMaN directed by Prof. Sajal K. Das, where I have strengthened not only research skills but also some other abilities, like leadership. This experience would be invaluable for my entire life.

Last but not the least, my special thanks go to my very nice and supportive parents, Xiande Li and Juan Wang, for their constant encouragement, love and understanding of my years of study. Also, I would like to thank my American parents, Hollis and Virginia Lackey, for their unconditional love and care, as well as providing me a second family in US, which strongly supported me to finish my Ph.D. study. Additionally, I want to thank my uncle and aunt, Roy and Cathie Steele, for their great love and care. Without having got to know them and met them in Beijing, China, in 2007, I would have not been able to know my American parents, and my current life would be a different picture. In addition, I thank my grandparents and other relatives for their love and understanding.

This dissertation work was supported by the US National Science Foundation (NSF) under Awards IIS-0326505, CNS-0721951, CNS-0916221.

May, 2012

ABSTRACT

EFFECTIVE AND SECURE USE OF HUMAN RELATION NETWORKS

Na Li, Ph.D.

The University of Texas at Arlington, 2012

Supervising Professor: Sajal K. Das

With the advent of Web 2.0 and advanced techniques of wireless devices (e.g., smart phones or iPhones), Online Social Networks (OSNs) and Mobile Social Networks (MSNs) are becoming integral part of our lives as two main digital social communities. Data collected from people's communication on OSNs and MSNs contains valuable information which makes human relationships more visible as compared to their existence in our physical world. For instance, the friend list on a user's profile page on Facebook clearly tells us the user's friendships with other users. Moreover, the short-range wireless communication techniques (e.g., Bluetooth) also enable us to "sense" human relations in MSNs composed of wireless devices carried by human. On the one hand, human relation networks can facilitate socially intelligent computing, for example, the friend recommendation service provided by most of the OSNs; on the other hand, we must ensure the security in using them to avoid users' concerns in OSNs and MSNs. This dissertation addresses four research problems in the effective and secure use of human relation networks in OSNs and MSNs:

(1) How can OSN owners preserve users's relation privacy in sharing data with the third-parties? We model a more realistic attack against users' relation privacy in

publishing OSN data, where part of the users may be identified from the published OSN data. We propose a privacy preservation model, called ℓ-diversity, to protect users' relation privacy. Furthermore, we propose three graph-manipulation based anonymization techniques to convert an arbitrary graph to a graph of ℓ-diversity with less topology change as compared to the original graph.

(2) How can a third-party analyst efficiently detect a minimum subgraph that connects a group of target users on OSNs with the minimum number of web accesses needed for online discovery? Based on the topological properties of human relation networks on OSNs, we propose two searching techniques which can quickly discover the subgraph connectivity on OSNs. Furthermore, we observe that users on OSNs are very well connected as we can find the connectivity of any group of target users by a small number of web accesses in searching on OSNs.

(3) In MSNs, the approach to inferring a relation between two people is based on the encountering frequencies of their wireless devices. However, in a malicious wireless environment, one device can create redundant encountering records by tailgating another wireless device. The redundant records can be used by the former to launch a black hole attack to disturb data forwarding. We design a reputation-based framework guaranteeing the use of reliable relations in data forwarding in MSNs.

(4) Highly active nodes which frequently meet with other nodes in an MSN, also called high-centrality nodes, are introduced in design of data forwarding protocols; however, overusing these nodes may cause serious network congestion. To deal with this issue, we build a framework which effectively spreads the congestion condition at high-centrality nodes to the entire network by social influence to notify data sources of the congestion situation so that they can adjust data generation rate to relieve network congestion.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

We live not only in a physical world but also in a digital world in which mobile (wireless) communications, social technologies and the Internet are linking people together and forming one interconnected network. With the advent of Web 2.0 and advanced techniques of wireless devices (e.g., smart phones or iPhones), Online Social Networks (OSNs) and Mobile Social Networks (MSNs) are becoming integral part of our lives as two main digital social communities. Indeed, these technologies have changed the way we live and interact with others. The number of people surfing the Internet as well as the amount of time they spend online (via wired or wireless communications), and the amount of time they are engaged via mobile phones or smart phones are ever increasing. People's communication on OSNs and MSNs creates a valuable information pool containing data that make human relation networks more visible, even at widely apart geographical distances, as compared to their existence in our physical world (see in Figure 1.1). For instance, the friend list on a user profile page on Facebook.com clearly tells us the user's friendship with others. Moreover, the short-range wireless communication techniques (e.g., Bluetooth) also enable us to "sense" human relations in MSNs composed of wireless devices carried by human. Thanks to the device discovery module embedded in the short-range communication techniques, a wireless device can record its encountering with other wireless devices. So for two devices which encounter frequently, most likely, their owners are colleagues working in the same building.

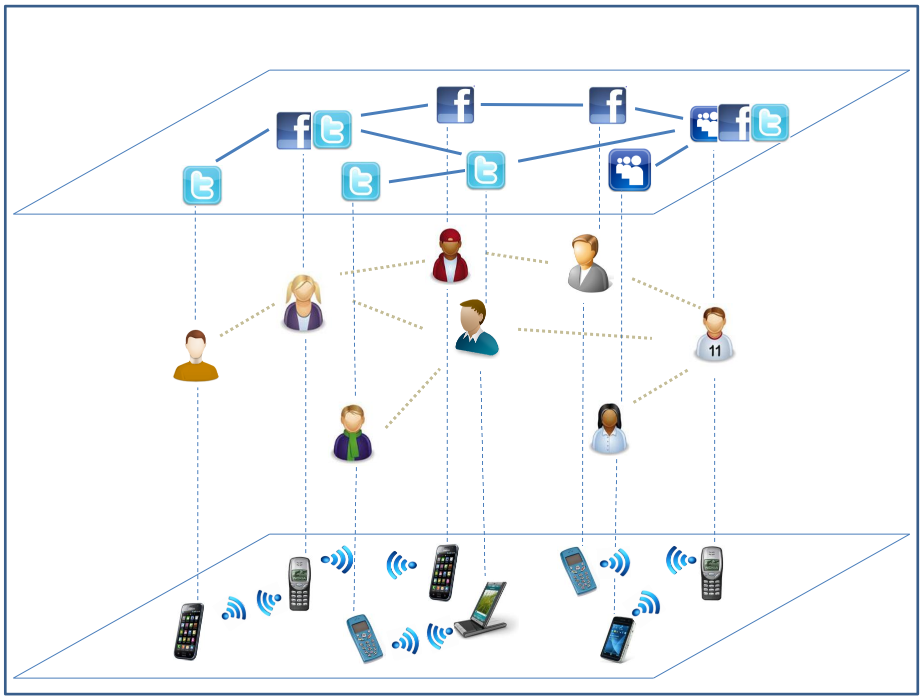


Figure 1.1: OSNs, MSNs, and Human Relaiton Networks

On the one hand, human relation networks can facilitate socially intelligent computing. As an example, the “friend” recommendation service provided by OSNs may recommend two users be friends if they have a considerable number of common friends in their friend lists. Another example is given the topology of the human relation network sensed by their wireless devices in an MSN, a message created at one person can be effectively delivered to another person based on the network topology, using devices’ opportunistic encounters.

On the other hand, using human relation networks also leads to significant concerns and challenges in both OSNs and MSNs. For example, when OSN owners share their user data to the third-parties, they should ensure the privacy of users’ sensitive relations will not be exposed. Similarly, there exist problems while using human relation networks in MSNs. Due to the way human relations are derived in

MSNs, such as by analyzing the contact/communication frequency of people's wireless devices, a fake relation can be easily created by tailgating another wireless device and collecting redundant contact records. An attacker may take advantage of the fake relations he creates to attract data in the network and drop them to disfunction the network. In that case, how to detect reliable relations and leverage them to facilitate data forwarding in MSNs becomes an issue required to be solved.

1.1 Scope and Contributions of this Dissertation

In this thesis, we particularly study four problems from the perspective of efficient and secure use of human relation networks in OSNs and MSNs as pictured in Figure 1.2, which include preserving users' relation privacy in publishing OSN data, detecting subgraph connectivity on OSNs solely based on local view, designing a reputation-assisted data forwarding framework in MSNs, and leveraging social-influence to handle traffic congestion in MSNs. Part of our work and results in this dissertation have been published at conferences and journals [44, 43, 42, 41]. In this section, we briefly state the challenges we are faced with in each of the problems and highlight our contributions.

1.1.1 Preserving Relation Privacy in Publishing OSNs

As we all know, OSN owners are sharing data collected from their users' online activities with the third parties, such as commercial companies or sociologists. The data collected by the OSNs are valuable for the third parties, because they can further mine the data to extract valuable information they need. For example, a company may use the data which form the basis for customer profiles to promote its products to the customers through an online recommendation system. Or sociologists may analyze the data to better understand the evolution of the social communities in our

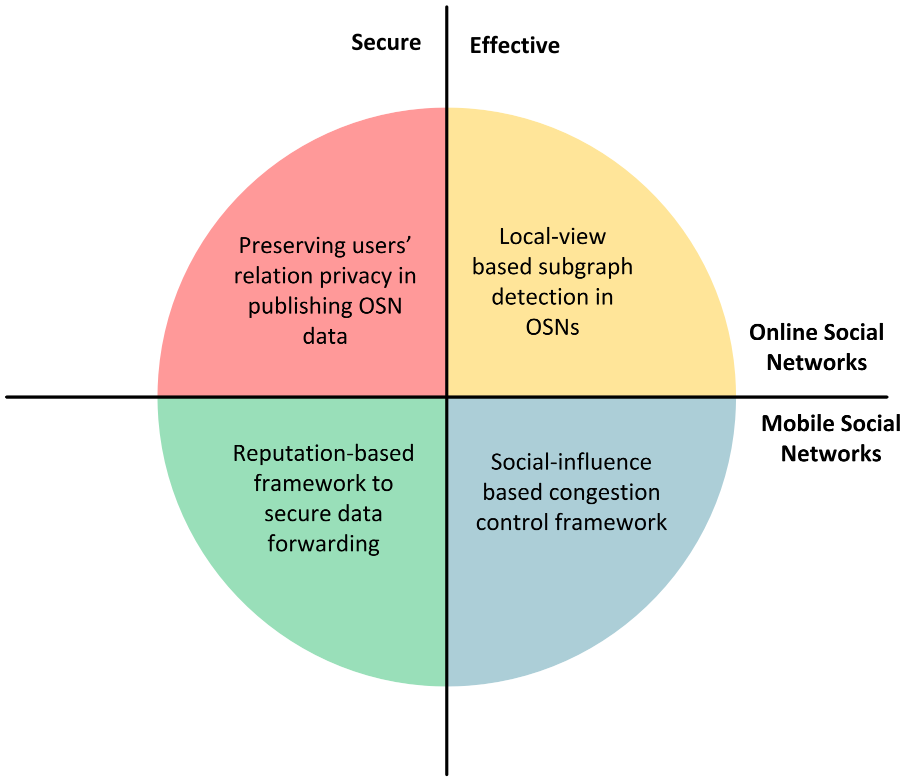

Secure
Effective
Preserving users' relation privacy in publishing OSN data
Local-view based subgraph detection in OSNs
Online Social Networks
Mobile Social Networks
Reputation-based framework to secure data forwarding
Social-influence based congestion control framework

Figure 1.2: The theme of this dissertation

physical world. However, in the procedure of publishing data to these third-parties, an OSN owner must ensure that the information which users are not willing to expose will be preserved, for example relationship privacy. The disclosure of relationship privacy becomes a growing concern for OSN users who usually want to keep it private. Nonetheless, it is quite challenging to protect such privacy as it involves multiple users and thus cannot be simply controlled through individual settings on the OSNs. In fact, the failure to protect user relationship privacy will cause serious consequences, not only severely undermining the popularity of OSNs, but also restricting the amount of data that OSN owners are willing to share with the third-parties. This dilemma cannot be solved by simply removing users' relations from the data before publishing.

Although doing so will completely hide all users' relations, it will also completely ruin the data utility to the third parties who usually need the relationship information contained in the data. For instance, a sociologist who is interested in researching the evolution of group dynamics cannot glean much practical insight from the published data without any details about users' relations. Therefore, the challenge for an OSN owner in this dilemma is how to satisfy both of the concerns, preserving user privacy as well as providing useful information to the third parties. In this work, we summarize our contributions in the following:

- (Problem Novelty) We first propose a novel attack model which is more practical against users' relationship privacy in publishing OSN data. Then we define the ℓ-diversity model for hiding the sensitive relationship between any two users one of whom can even be identified from the published data.
- (Solution Novelty) We develop three graph-manipulation based anonymization techniques which can ℓ-diversify any graphs. Two of them publish a subgraph with ℓ-diversity while the other publishes a supergraph with ℓ-diversity. We not only prove the correctness of our techniques in ℓ-diversifying an arbitrary graph but also analyze their time complexity.
- We conduct a comprehensive set of experiments on both synthetic and real-world social network data sets, and evaluate the performance of our techniques by measuring the utility loss caused by our anonymization techniques. Moreover, we demonstrate the tradeoff between privacy preservation and utility loss.

1.1.2 Local-View based Subgraph Detection in OSNs

OSNs collect data from their users' various activities including logging on/off, adding friends or constructing groups. Part of these data is visible on OSNs, for example, a friend list is available on each user's profile web page. The OSN site owners can picture the entire network of their users' friendships, and further develop new application services, for instance, friend recommendation. However, an interesting question is how people other than OSN owners can take advantage of the information provided by OSNs for their diverse purposes. The challenge here is the lack of the global view of the OSN network due to the limited knowledge available on each user's profile page. Apparently, crawling the entire OSN for analysis is not a realistic approach, considering the time it requires and the dynamic nature of the OSNs. Besides, nowadays the OSNs limit the number of web accesses from one (or a group of) IP address(es), which further increases the hardness of gleaning information from OSNs. Therefore, we need to design more effective and efficient techniques to mine the OSN graphs which are not fully presented in front of us.

Motivated by the aforementioned question, we study subgraph detection from the perspective of a third-party analyzer - discovering a small connected subgraph which pictures users' relations on OSNs. Here we illustrate two promising applications which are orthogonal to our subgraph detection in OSNs. First of all, an analyzer can leverage the online relationship network to plan a successful cocktail party. The "success" is referred to as having more fun by not only inviting a group of target people, but also making all attendees acquainted with each other directly or indirectly by inviting additional people. The selection of these additional people comes across the detection of a small subgraph from an OSN, such as Facebook.com. The other application of our subgraph detection problem is to facilitate US government to investigate a known list of terrorists by extracting from an OSN a subgraph which connects all these known terrorists together.

We should note that it is a profound limit for the third-party analyzers to conduct subgraph detection in OSNs, as the entire OSN graphs are not accessible to others than the OSN site owners. Furthermore, in an OSN the information that can be used for the third-party analyzers is limited on each user's profile web page, which further increases the challenge of detecting a subgraph. For instance, only the list of friends is visible on a single user's profile page on the Facebook, or the list of users that a user trusts and is trusted by are posted on the user's profile page on Amazon.com and Epinions.com, therefore, we call them *local view*. The local view is available by web querying which is referred to as accessing the profile web page of a user. A third-party analyzer can get a larger view of the entire OSN relation netwok by querying more users; however, issuing a query to an OSN is not cost free, for example, intensively accessing an OSN causes the heavy workload at the OSN server. Therefore, we need to design efficient and effective techniques for the third-party analyzer to conduct subgraph detection on OSNs. Our contributions to this research topic can be summarized as follows.

- (Problem Novelty) We propose a novel problem of detecting subgraph connectivity on OSNs from the perspective of a third-party analyzer. Particularly, we search for a minimum-size subgraph which covers a group of target users on an OSN, without knowing the entire OSN graph. Furthermore, we takes into consideration the cost of web accesses issued for online subgraph detection.
- (Solution Novelty) We design two algorithmic techniques to solve our proposed problem. Those two techniques, called Unbalanced Multiple-Subgraph (UMS) and Balanced Multiple-Subgraph (BMS), are designed based on well-known social network topological properties, such as small-world phenomenon, power-law distribution of vertex degrees, as well as well-connectivity of vertices of high degrees.
- (Insightful Experimental Study) We conduct a comprehensive set of experiments on large-scale real-world OSN data sets, and evaluate the performance of our algorithms with respect to their effectiveness and efficiency for subgraph detec-

tion. The experimental study shows the well-connectivity of OSNs is not limited to vertices of high degree, but for all of users as our proposed techniques, especially for BMS, can discover the connectivity of any group of arbitrarily selected vertices through a small number of web accesses to the OSNs.

1.1.3 Reputation-Based Data Forwarding in MSNs

Since the human relation networks formed in MSNs are quite different from those in OSNs, we are faced with different challenges in using human relation networks in MSNs. In our context, an MSN consists of several wireless devices, such as mobile phones, which are carried by people in our physical community. Each wireless device records its encounters with other devices, and the records are used to analyze the relations among people. In such a network, we are particularly keen on data forwarding depending on the short-range communication (e.g., Bluetooth) between the mobile devices rather than Short Message Services (SMS). In general, a wireless device that frequently encounters the destination device indicating the close relationship between their owners should be a good data forwarder.

Due to the way the human relation is derived in MSNs such as by analyzing the contact/communication frequency of people's wireless devices, fake relations can easily be created by manipulated encounters, for example, by tailgating a wireless device to collect redundant contact records. With the redundant records, a wireless mobile device carried by a malicious person (or called an attacker) can exaggerate his ability of encountering destination nodes so as to attract data. The attempt of the attacker is to drop data rather than further forward them aimed to disfunction the network. This type of attack is particularly named *Black Hole Attack*.

The black hole attack was first noticed in [40]. To counteract arbitrarily bloating, the Encounter Ticket (ET) is first proposed in [40] as the evidence of node encounters. However, a malicious node is still able to exaggerate its ability to meet a destination by one-time tailgate attack, in which that malicious node collects redundant ETs by tailgating the destination one time, and then moves around the data source to intercept data. Although the authors in [40] propose a technique to ignore redundant ETs generated within a short interval, it may not work effectively under multi-tailgate attack, where an attacker moves in and out of the communication range of the destination to collect valid yet redundant ETs with large intervals. The reason behind such ineffectiveness is that the data forwarding protocol still depends on the probability of meeting a destination to evaluate an encounter's competency of delivering data.

To deal with this problem, we propose a trust-based framework which can be flexibly integrated with a large family of existing single-copy data forwarding protocols in MSNs, aiming at providing comprehensive evaluation to an encounters ability of delivering data. With the help of our proposed framework, not only black hole attack but also arbitrarily forwarding attack can be defended against effectively. Our contributions are briefly summarized as follows.

- Positive Forwarding Message (PFM) is designed as the evidence of the forwarding behavior of a node. With the help of some fields in PFM, two kinds of attacks, deliberately dropping data and arbitrarily forwarding data, can be effectively counteracted.
- A trust-based framework is proposed to assist data forwarding in MSNs, where we point out the "Self- Trusting" principle in data forwarding in MSNs.
- As a case study, we integrate our trust-based framework with the existing data forwarding protocol, PROPHET [45]. Through experimental study, we demonstrate the effectiveness of our framework against black hole attacks as well as

evaluating some features in our framework, such as stabilization and the effect of "Self-Trusting".

1.1.4 Social-Aware Congestion Control in MSNs

In addition to the traditional data forwarding protocols mentioned in the previous problem, social-awareness has been incorporated into the design of data forwarding protocols to increase data delivery in opportunistic-encounter based Mobile Social Networks. The social-aware data forwarding protocols are based on the assumption that the nodes which more frequently meet with others in the network, namely high-centrality nodes, are more likely to encounter destination nodes, therefore, they should be good data forwarders. However, if all of the data traffic is forwarded towards these high-centrality nodes the number of which usually is limited in social community, they will get congested soon and thus become the bottleneck of network communication.

To date, although a multitude of work has addressed congestion control in MSNs from different angles, varying from migrating data to avoiding transmission along congested paths, we believe that slowing down the data generation rate at data sources should be the most robust approach to controlling congestion in MSNs. This is because even for employing migration-based or avoidance-based approaches, there always exists a condition where almost all of the nodes are completely congested, while data sources are unaware of it and still keep high data generation rate. A couple of techniques of adjusting traffic generation rate at data sources have been proposed for controlling congestion in our traditional networks, such as the Internet. The common mechanism of these techniques is to route a control message from a congested node back to the data source to notify the data source of the congestion at the node. However, directly applying these techniques to controlling the congestion in MSNs is impractical due to the lack of the contemporary path between any pair of nodes in MSNs. Furthermore, rather than notifying a particular data source, all data sources should be aware of the congestion at high-centrality nodes in MSNs as if the high-centrality nodes get congested, the entire network cannot function for data forwarding. Therefore, an efficient and effective mechanism is called for to control traffic congestion in MSNs. Our main contribution is to leverage social influence to spread congestion signal which captures the congestion condition at high-centrality nodes in the network to notify data sources so as to adjust their data generation rate to alleviate traffic congestion in the network.

1.2 Organization of Dissertation

Chapter 2 reviews existing literature and provide some necessary background on a few topics related to the four problems researched in this dissertation. Chapter 3 discusses the privacy preservation in publishing OSN data and propose three anonymization techniques to defend against the disclosure of user's relation privacy in data publishing even when users' identities may be partially exposed. Chapter 4 addresses efficiently detecting subgraph connectivity of a group of target users based on local view on OSNs. Chapter 5 researches on how to strengthen the robustness of human relation networks against black hole attack in MSNs. Chapter 6 discusses about leveraging social influence to control traffic congestion in MSNs. Finally, some conclusions are drawn in Chapter 7 along with future work discussion.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter mainly introduces related work to the four research problems to be addressed by this dissertation. The roadmap of this chapter is as follows. In Section 2.1 we investigate the existing techniques of preserving users' relationship privacy in publishing OSN data and propose a novel taxonomy of these techniques. In Section 2.2 we discuss some topological properties of social network graphs which play an important role in designing our techniques of detecting subgraph connectivity in OSNs from the perspective of a third-party analyst. Additionally, we review some literature relevant to subgraph detection in Section 2.3. Moreover, we introduce data forwarding protocols developed for Mobile Social Networks, followed by the discussion on the classic trust mechanisms in computing in Section 2.5 and many congestion control mechanisms in the networks in Section 2.6.

2.1 Relation Privacy in Publishing OSNs: A Survey

As online social networks (OSNs) become ubiquitous, more and more third-party enterprises are exploiting data from OSN websites. An OSN owner can either publish its collected data directly or provide a search interface, which third parties can use to issue queries as part of customer research or social experiments. Data can form the basis for a customer profile, for example, which the company can then use to promote its products through an online recommendation system.

In serving these third parties, however, an OSN owner must also respect its network members' privacy by preventing the disclosure of sensitive information in the data it provides. Relation privacy the protection of relationships that the user would like to remain private is a growing concern because it involves multiple users and thus cannot be controlled through individual settings. Moreover, such relationships are clearly foundational to a social network's success. According to a recent study [10], a Facebook user has more than 150 friends on average. Not being able to protect the privacy of these relationships could severely undermine these networks' popularity and restrict the amount of data that OSN owners are willing to provide to third parties.

The solution to this dilemma cannot be the straightforward removal of user relationships because relationships are often part of the data that third parties want. Sociologists interested in the evolution of group dynamics, for example, arent likely to glean much practical insight from OSN data published with no details about user relationships. Removing all relationship information from the data would certainly hide the social networks topology and thus protect the user. However, it also severely limits the datas usefulness to third parties. The challenge for an OSN owner then is how to satisfy both concerns.

Graphs are commonly used as a tool to present social network data, in which the vertices represent users and the edges denote relations. Each user's information included in his individual profile, such as his preference and affiliation, is categorized and labeled as vertex attributes. The strength of each relation can be represented by an edge weight. For example, the weight of a friendship edge between two users in a social network graph can distinguish two cases: either the two users just know each other, or they have been good friends for a long time. With the help of the graph presentation, we define the (relation) private information as the existence of an edge or a path between two vertices, as well as the sensitive edge weight in social network graphs.

There exist substantial literature on protecting user relationship privacy in publishing online social network data, in which the techniques can be categorized into two groups based on whether users' identities will be released in the procedure of data publishing. Specifically, one group of anonymization techniques preserve relation privacy by completely hiding users' identities, and the other group of techniques are intended to protect relation privacy while users' identities are released with full disclosure.

2.1.1 Complete Identity Anonymization

Intuitively, if we remove users' identities or assign pseudo-identities for them, although we can see edges in a published graph, we cannot tell who and who have a relationship. Thus, some research has been focused on preserving users' relationship privacy by hiding their identities. In the literature, a wide variety of techniques have been proposed to preserve users' identities in published social network graphs, which can basically be classified into two categories: (i) topology-preserving techniques and (ii) vertex-classifying and relabeling techniques. The former leverage on graph modification against topology-based attacks [46, 71, 72, 16, 14, 27, 70], where an adversary is assumed to master some topological properties of a user, such as its vertex degree or its neighborhood structure, while the latter preserve users' identity privacy by clustering the vertices and relabeling them without any graph manipulation. In the following, we will introduce these two categories of techniques in detail.

2.1.1.1 Topology Preserving

Knowledge of a user's topology, such as vertex degree the number of vertices adjacent to a vertex or neighborhood structure, makes that user and those with whom he or she interacts vulnerable to topology-based attack. The examples in Figures 2a through 2e make it clear that published social networks are not invulnerable to this kind of attack. In Figure 2.1, for example, if the attacker knows that Alice has two friends, he can easily identify Alice from the graph, even though he cannot distinguish Bob from Tom.

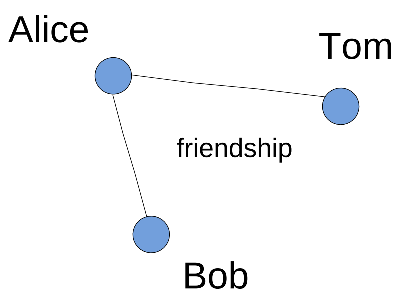


Figure 2.1: Naive identity removal

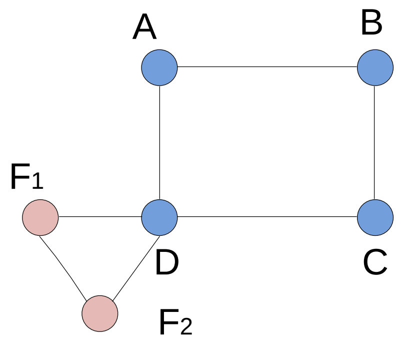


Figure 2.2: Active attack

An active attack [8] can also compromise user identities by actively injecting dummy users in an OSN. In Figure 2.2, an attacker successfully identifies User D using this method. After creating two fake user accounts on the OSN, F1 and F2, the attacker uses the fake accounts to identify real users on the basis of relations between F1 and F2 and real users. In this case, by knowing the dummy user profile details, the attacker could first identity F1 and F2 and then identify User D because of D's direct association with F1 and F2.

To protect user identities against topology based attacks, researchers have proposed k-anonymity and cluster-generalization techniques.

K-anonymity techniques. Techniques in this category aim to make each user indistinguishable from at least k C 1 others. The goal of the k-degree anonymity

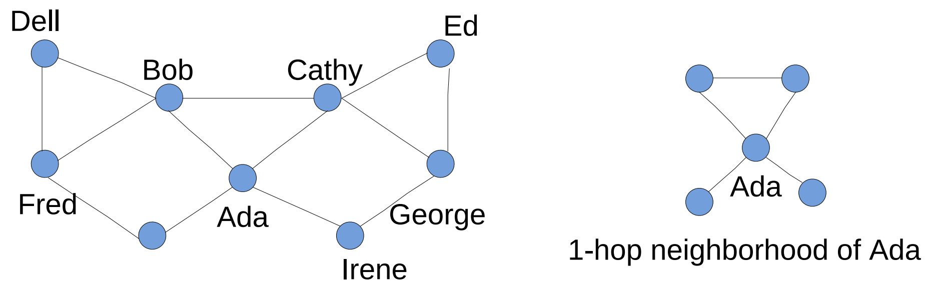


Figure 2.3: A failure of 2-degree anonymity

technique is to protect against an adversary who has some knowledge of the target user [46], which that adversary has gleaned through vertex degrees. On the basis of the original vertex degrees, the algorithm generates a sequence of revised degrees that satisfy k-anonymity, in which each vertex has at least k C 1 other vertices with the same degree. It then generates a new graph based on the revised degree sequence. A guiding principle for graph generation is to minimize the number of edges that change from the original graph [46]. These edges represent data utility, and each change in turn represents a loss of that utility.

The k-neighborhood anonymity technique guards against knowledge not only from the vertex degree but also from each vertex's neighborhood topology [71]. As Figure 2.3 shows, given Ada's neighborhood structure, an attacker can identify Ada, although the graph already satisfies 2-degree anonymity. This identification ability shows the insufficiency of the k-degree anonymity algorithm to thwart an attacker who knows the user's neighborhood topology.

The k-neighborhood anonymity technique first extracts and encodes all vertices neighborhood structures. It then forms groups of vertices with similar neighborhood codes, ensuring that each group is at least size k. Finally, it anonymizes each group by modifying the graph to achieve the same one-hop neighborhood topology, thereby satisfying k-neighborhood anonymity.

More recent techniques, notably k-automorphism [72] and k-isomorphism [16], can preserve user identity to an even greater degree. The k-automorphism algorithm guarantees k-anonymity even against an adversary who could know arbitrary hops in each users neighborhood. To achieve such a guarantee, the proposed algorithm produces a supergraph of the original graph that satisfies the k-different match principle.

To illustrate this principle, consider a published network $G^* = (V, E)$, where V is the vertex set and E is the edge set. For a subgraph $Q = (V_Q, E_Q)$, where $V_Q \subseteqq V$ and $E_Q \subseteqq E$, the k-different match principle requires the satisfaction of two conditions. The first is that at least k matches are isomorphic to Q under k different bijective functions $f : V_Q \rightarrow V$, which also maps each edge $(u, v) \in E_Q$ to a unique edge $(f(u), f(v)) \in E$. The second condition is that, for any two (of the k) matches, no matching maps the vertex in V_Q to the same vertex in V.

The k-automorphism algorithm consists of three steps. Graph partitioning first partitions the original graph into n blocks (subgraphs) and then clusters these blocks into m groups, each of which contains at least k blocks. The next step, block alignment, makes blocks isomorphic to each other in each group. Finally, block-crossing edge copying inserts block-crossing edges (if the original graph contains edges across different isomorphic blocks) into the published graph. As Figure 2.4 shows, the k-automorphism technique can still allow a breach in relation privacy even when user identity is completely anonymous. Because an adversary knows that User A has two friends and User B has three friends, he can infer that the two users are friends in the social network, although he isnt able to identify them individually. The vulnerability arises from the lack of structure-diversity for edges in the published graph in this case, all vertices of degree 2 are fully connected with those of degree 3.

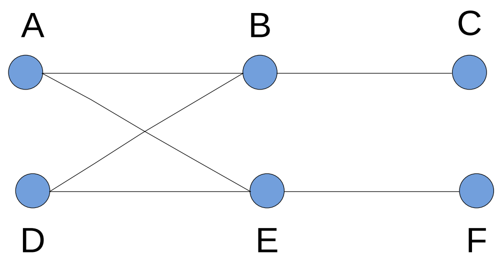


Figure 2.4: A failure of 2-automorphism

In contrast, the k-isomorphism algorithm [16] achieves k-secure graph anonymization. Thus, an adversary cant determine that two users are linked by a path of (at most) a certain length with a probability of more than 1/k, even given any subgraph of the original graph that includes two users of interest. The key technique is to partition the original graph G into k disjointed subgraphs and then anonymize each subgraph by inserting or deleting edges to make all k subgraphs isomorphic to each other.

Cluster generalization. Unlike the k-anonymity based techniques, clustering-based algorithms [14, 27, 70] use generalization methods to preserve user identity and structure against topology-based attacks. The techniques first cluster users into groups according to similarities of their attribute values or neighborhood topologies. Then, instead of publishing each user's detailed information, the technique posts a summary of each group, such as the number of users and internal edges. Figure 2.5 gives an example. A critical drawback of this technique, however, is the loss of data utility because detailed relationship information is no longer available.

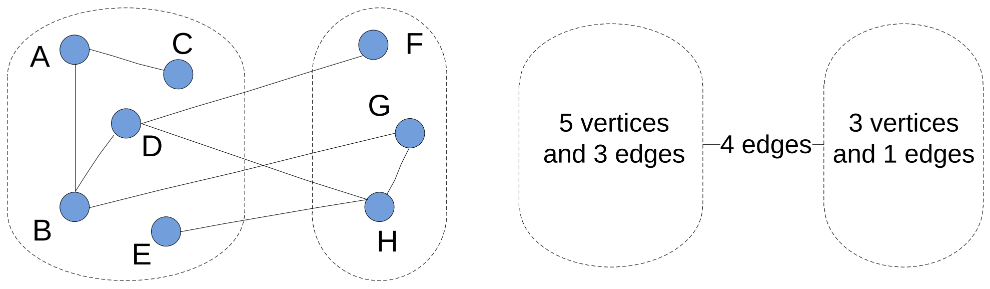


Figure 2.5: Cluster generalization

2.1.1.2 Vertex Classifying and Relabeling

The second category of techniques to completely anonymize user identities (and thus preserve the interactions between users and other entities) does not require graph modifications. Rather, the idea is to group users into classes and then assign a set of identities to each class, thus aiming to hide users real identities within a class. Most techniques in this category [11, 18] consider a bipartite graph, as opposed to an arbitrary graph. The bipartite graph consists of two subsets of vertices one denoting users and the other representing interactions, such as email, or entities, such as purchased goods. Edges between the subsets could disclose private information about the users and interactions in the form of some association.

The constraint on association diversity in each class governs the procedure for classifying users that is, any two vertices in each class must not share neighbors in other classes. Without the constraint on association diversity, if two users in a class happen to be the only ones involved in a specific interaction, that interaction will be disclosed, although the users' identities are hidden. Figure 2.6 shows that scenario for Users 1 and 2. The attacker is convinced that Users 1 and 2 have communicated through email, although he is not able to determine who's who. Obviously,

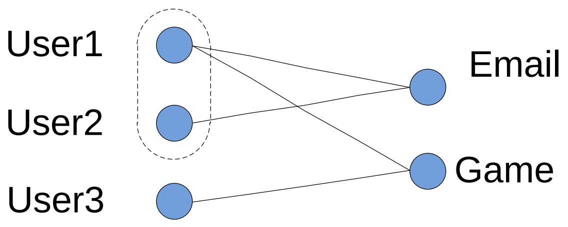


Figure 2.6: Vertex classifying

vertex classifying and relabeling effectively protects user identities without requiring graph manipulation. However, in a graph in which many users associated with one interaction, it might not be possible to classify so many users without violating the association diversity constraint. Moreover, vertex classifying and relabeling would have to rely on generalization techniques to defend against topology-based attacks.

2.1.2 Completely Disclosing User Identity

In some scenarios, maintaining real user identities in the published graph enhances data utility, and the edge weight becomes the sensitive information. For these scenarios, techniques must protect sensitive edge weights, and techniques based on manipulating OSN graph models no longer work.

One technique for protecting edge weights is Gaussian randomization multiplication [47], which uses value perturbation to manipulate the original weight of each edge by an i. i. d Gaussian random variable with mean 1 and variance σ. Because edge weights are tabular, not relational, data, OSN security engineers can adapt some existing techniques that the database and data mining communities use to preserve tabular data privacy.

Another technique models the weighted graph on the basis of the property to be preserved (shortest paths, minimum spanning trees, and so on), and then reassigns the edge weights to obtain an anonymous graph that satisfies the model [21]. The approach defines a linear property of a graph as one expressible in terms of inequalities involving linear edge-weight combinations. The desirable model captures the inequalities that the edge weights must obey to preserve the linear property. In other words, any solution from such a model would ensure the anonymization of edge weights while preserving the linear property being considered. The technique's authors also address how to select solutions that will keep an attacker from inferring even relative edge weights for example, which two vertices have the largest edge weight. One problem is deciding which properties third parties will find the most useful and thus which the OSN owner should preserve before publishing the data.

2.2 Social Network Topological Properties

Note that although human relation networks in OSNs and MSNs are established in terms of different types of data collected from our digital world, their graphs present similar topological features of social networks, such as small-world, scale-free and well-connectivity of nodes of high degrees.

Small-world is one of the well-known social network topological properties, which was first observed through a series of striking experiments conducted by Stanley Milgram and his coworkers in the 1960's [36, 49, 61]. This property is also translated into "six degrees of separation", referred to as the idea that everyone is at most six steps away from any other person on Earth, which indicates the small diameter of social networks. The small-world property ensures the existence of short path between any pair of vertices in social network graphs.

A scale-free network is a network with a power-law degree distribution, at least asymptotically. That is, the fraction $P(k)$ of vertices in the network with k direct neighbors for large values of k is given as $P(k) \sim k^{-\gamma}$, where γ is a constant typically in the range $2 < \gamma < 3$. The power-law degree distribution indicates that the number of vertices goes down quickly as their degrees go up, moreover, the number of vertices of highest degree is limited.

The well-connectivity among vertices of high degrees is also a topological property of social networks which was discovered by a comprehensive set of experiments on real-world social network data sets [64].

2.3 Subgraph Connectivity in Graph Mining

Research effort has been made on the topic of subgraph connectivity in the domain of graph mining. Faloutsos et al. [22] and Tong et al. [60] proposes solutions for finding a subgraph that connects a set of query nodes in a graph, where the proximity between nodes is defined depending on the global topology of the graph. Specifically, they extracted subgraphs including vertices as close to the query nodes as possible, where the closeness is quantified by the similarity measure between two vertices. In its subsequent work, Koren et al. [35] redefined the proximity measures based on "cycle-free effective conductance" (cfec) and proposed some algorithms for optimizing the cfec measure. Another work by Asur and Parthasarathy [7] suggests the concept of *viewpoint neighborhood* analysis to identify neighbors of interest to a particular source in a dynamically evolving network, associating their measure with heat diffusion. Cheng et al. [17] investigated the problem of connecting query nodes in a context-aware framework. They first employed modularity measure to partition the graph, and then studied the connectivity in both intra-community and inter-community levels. Kasneci et al. [32] proposed a random walk-based approach to find informative subgraphs associated with a group of query nodes in entity-relationship diagrams. Most recently, Sozio et al. [58] addressed the searching for the densest subgraph containing all query nodes with and without size constraint. However, the difference between the above line of research and our work in this dissertation is they assume the entire graph is given for detecting the subgraph connectivity, which makes it impossible to detect the subgraph on OSNs for a third-party analyst.

2.4 Data Forwarding in Mobile Social Networks

There has been extensive work in designing data forwarding protocols in MSNs. Some of them address how to effectively forward data to a single destination node or a group of nodes pre-known [12, 13, 45, 29]. In this category, a qualified forwarder is supposed to have stronger competency of successfully delivering data to the destination node, which has been estimated through the probability of meeting the destination in the near future. The more possible to meet the destination, the more competent to deliver data. Some metrics are proposed to predict that probability. For example, in [29], the role of people in social networks is exploited to estimate the probability of delivering data. Specifically, a person who actively moves in a community is a high-qualified data carrier from the perspective of data delivery, as he has a higher chance to meet others in the network, including the destination. Some other protocols leverage the encounter history to predict the delivery possibility. With the assumption that people's mobility pattern presents some repetitive features, two people who frequently meet before will more likely encounter each other again in the future. However, most of the existing protocols in this family ignore the difference between delivering data and meeting the destination in a hostile environment. In a malicious wireless environment, a adversarial node is motivated to arbitrarily boost its probability of meeting the destination to intercept data from other nodes, and then arbitrarily forward or even drop them, aiming at detrimentally degrading the network performance, which is called black hole attack.

Thanks to the novel structure of Encounter Ticket (ET) proposed in [40], which can only be generated based on real node encounters, arbitrarily bloating the ability of meeting a destination is avoided. However, a greedy node can still boost its times of meeting a destination node by collecting redundant ETs from one-time tailgate attack. In such an attack, a malicious node launches tailgate only once and then moves around the data source to intercept data, as illustrated in Fig. 2.7. The authors in [40] remove redundant ETs by ignoring those with similar generation time. However, the adversary may mount another kind of attack, where an attacker frequently moves in and out of the communication range of a destination node to collect non-redundant ETs and then wanders around the destination node rather than a source node to intercept data. We call such an attack multi-tailgating attack, as depicted in Fig. 2.7, which actually has more destructive impact on the network performance than one-time tailgate attack. Since the technique proposed in [40] still leverages the possibility of meeting the destination to estimate the delivery competency, it could not thoroughly counteract the black hole attack under multi-tailgating.

In addition, the authors in [54] propose a mechanism to detect black hole attacks in the same context. After any pair of nodes exchange data, they create receiving and forwarding records for each other and keep them in their memory. Then, whenever they meet again after further forwarding the data, they will validate the history records from each other to determine whether the black hole attack has been launched by peers. However, the verification can only be conducted when the same two nodes meet again, which may not be in a relatively real-time manner especially in MSNs characterized by long delays and frequent network partitions. Moreover, the authors did not address how to defend against arbitrarily forwarding data in MSNs. In a worst case, if a malicious node deliberately forwards data to a node which will never be able to forward data to others because of its isolation from the network, the

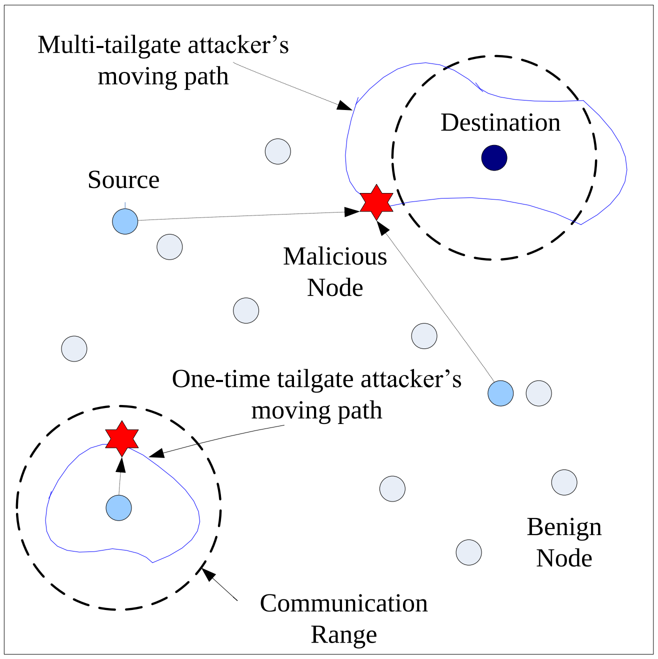

Multi-tailgate attacker's moving path
Destination
Source
Malicious Node
One-time tailgate attacker's moving path
Benign Node
Communication Range

Figure 2.7: One-time tailgate and multi-tailgate attacks

network performance will be degraded as badly as that under the black hole attack using the scheme proposed in [54].

More recently, social-awareness has been introduced into the design of data forwarding protocols in MSNs. Since it's human beings who carry the mobile devices as mobile phones roaming in our physical community, the mobility pattern of mobile devices presents some social community features, for example, the repetitiveness of daily traveling paths or the active interaction of some popular nodes with other nodes in the same community. Thus, putting social flavor into the protocol design will be able to aid data forwarding in MSNs.

A plethora of work has been conducted along this direction. In [30], the authors designed a protocol, called BubbleRap, addressing data forwarding across multiple communities. In BubbleRap, each node is labeled with two values to indicate its local and global popularity. A node's local popularity is referred to as the node's centrality in the local community, while its global popularity denotes its centrality in the whole network consisting of multiple communities. BubbleRap first continuously forwards a data message to its encountered nodes with higher global popularity until meeting a node existing in the same community as the destination node. Then, the message is forwarded further to the encounters with higher local popularity till the message either reaches the destination or is dropped due to the expiration.

Another work, called SimBet, introduced in [19] realized social-aware data forwarding by designing three metrics based on social analysis of a node's past interactions as well as a node's "betweenness" centrality (calculated using ego networks), a node's social "similarity" to the destination node, and a node's tie strength relationship with the destination node.

More recently, PeopleRank algorithm is proposed in [50] for choosing next-hop data forwarders. PeopleRank is essentially adapted from PageRank, a classic link analysis algorithm used by the Google Internet search engine, that assigns weights to webpages to rank its relative importance. By employing this idea, PeopleRank tags people as "important" when they are linked to many other "important" people in the social relation graph which is formed by the knowledge (e.g., friendship) publicly available on online social sites. In evaluating an encounter's capability of forwarding data, both a node's importance and its previous interaction with other nodes are considered. The assumption behind such evaluation is that nodes socially well connected in online social sites are better suited to forward messages towards any given destination in MSNs.

2.5 Trust in Computing

Trust-based data routing has been extensively studied in wireless networks, such as MANETs [5, 9, 59, 65]. As shown in Fig. 2.8, the framework of a Trust Management System (TMS) basically consists of two components, a Watchdog and a Reputation

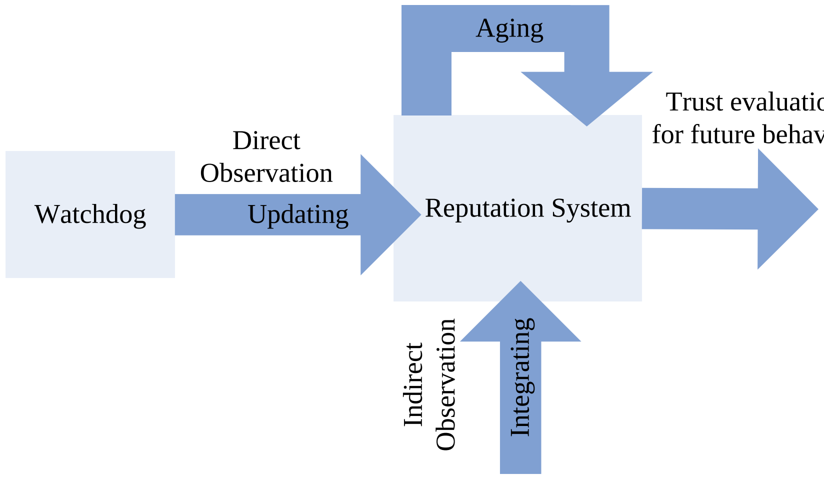


Figure 2.8: Structure of TMS

System (RS). The functionality of Watchdog is to monitor the real routing behavior of a node and then feed that information into RS to update the reputation of that node. In general, the RS conducts three tasks relevant to reputation: 1) updating reputation opinion based on the direct observation at the Watchdog (i.e., first-hand information), 2) integrating reputation by combining the indirect information (i.e., second-hand information) from other members with the first-hand information, and 3) aging reputation to refresh first-hand information.

Essentially, these trust-based data routing protocols in MANETs take advantage of contemporary routing paths between a source node and a destination node to monitor routing behaviors of intermediate nodes along the paths. For example, some protocols use the Watchdog of a data source node to collect the ACKs of the data sent out from the source as the evidence of the good routing behavior of intermediate forwarders. Specifically, the ACKs are generated at the destination node when data are delivered and then routed back to the data source along the same transmission path as the data come along to the destination. ACKs demonstrate that the nodes along the routing path indeed help forward data properly. In MANETs, the delay of the

ACKs can be ignored because of the assumption on the existence of the contemporary routing path between the source node and the destination node. Some other protocols allow the Watchdog to directly monitor wireless channels to check whether or not the next-hop forwarder forwards data with integrity. The successful channel monitoring relies on the assumption of the relatively slow mobility of nodes in MANETs. In particular, if the data message sent out from the next-hop forwarder is exactly the same as the original one sent out from the previous-hop node (i.e., the Watchdog owner), the next-hop forwarder is assumed to be a benign node which behaves well for data forwarding. However, the Watchdog mechanisms developed for MANETs can not be directly applied to MSNs because of the features of frequent network partitions and long delays in MSNs. As the contemporary path does not exist any more in MSNs, neither channel monitoring nor end-to-end ACK works effectively.

2.6 Congestion Control Strategies

Traffic congestion problem has been studied in wireless networks characterized by long delay and frequent partition. A storage routing protocol was proposed in [56], where the arrival of data at a congested node invokes the migration of a set of selected data from the node to a set of its neighboring nodes. The neighbors with the lowest migration cost are chosen for storing the migrated data, where the migration cost is referred to as the cost of transmission and storage. For the selection of data to be migrated, the authors introduced three approaches, picking up data with the oldest time stamp or data which are queued at the head of the queue, or at the tail of the queue. The limitation of the proposed routing protocol is there exists a boundary situation in which data will not be allowed to be migrated further because of the serious congestion condition at almost all of the nodes in the network.

In another work [28], the authors defined three states for each node, namely normal state, congestion adjacent state and congestion state, based on the preset thresholds of the utilization rate of node storage. Furthermore, such three states convert to each other based on the utilization rate of storage at each individual node. A node in congestion adjacent state will notify its neighbors of its congestion state by broadcasting a control message, which will label the links of that node with "Half-Hanged". The selection of a path for data forwarding is intended to avoid Half-Hanged links. The limitation of the proposed mechanism is that each node's status is determined solely by its own storage and bundle forwarding, therefore, it cannot accurately reflect the congestion occurring in the network. In addition, similar to the problem existing in migration-based scheme in [56], data sources may keep high data generation rate, while all other nodes in the network have been congested seriously.

Moreover, the work [52] first noticed the potential vulnerability of social-aware data forwarding in MSNs. Specifically, a large amount of data traffic is carried by a fewer nodes typically with high centrality or betweenness. Such bias on traffic distribution makes the network unsustainable, causing the quick depletion of constrained resource on the heavily-used nodes, the potential random failure, or even attacks against these nodes. To balance traffic distribution over the network, the authors designed a fair routing protocol [52]. The fairness is achieved by introducing assortative-based queue control mechanism, where the queue size is defined as social status, which indicates how useful a node is for data forwarding. Only nodes with high status can forward data to the nodes with lower status. The social status is integrated with another metric associated with long-term and short-term interactions between nodes on their pair-wise encountering. Through their experimental study, they demonstrated the effectiveness of their protocol on balancing traffic distribution in MSNs.

Following [52], some congestion-aware data forwarding strategies have been designed [26, 53]. Firstly, the authors in [26] demonstrated the biased load distribution towards high-centrality nodes in MSNs by conducting experiments in real-trace based data sets. Additionally, the authors proposed a congestion control framework which consists of two components, *Forwarding Heuristic* and *Congestion Control System*. The forwarding heuristic evaluates an encountering node by linearly combining smoothed centrality, neighbor similarity and history-based performance metric. In addition, two parameters, *Receptiveness* and *Retentiveness*, are leveraged in the congestion control framework to evaluate the availability of a node. Although the authors noticed the social impact in traffic congestion in MSNs, they did not take the impact into consideration in design of their framework which we, however, consider in our work.

CHAPTER 3

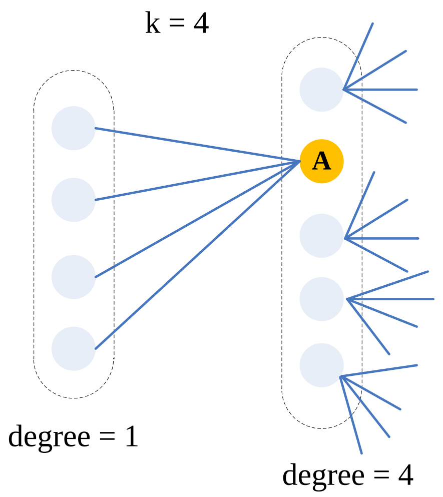
PRESERVING RELATION PRIVACY IN PUBLISHING OSNS

This chapter addresses preserving user's relation privacy in publishing OSN data while partial users' identities may be disclosed. As evidenced by recent work on de-anonymizing real-world OSNs [51], it is extremely difficult, if not impossible, to absolutely hide all users' identities from the OSN data. The fundamental reason is a user may be willing to publish his/her personal information to the public, through blog for example, which may be used by an adversary for identifying the user from the published OSN data. Note that the control on releasing such personal information is in the hands of individual users rather than OSN owners. Therefore, the critical challenge for OSN owners to preserve users' relationship privacy in publishing data is how to protect relationship privacy even when one of the associated users is identifiable from the published data. To address this challenge, we adapt the ℓ-diversity anonymity model which was originally proposed in [48] for preserving the privacy of tabular data to our context of protecting users' relationship privacy. Our model ensures that an attacker is prevented from successfully inferring the existence of a sensitive relationship between two users with confidence greater than $1/\ell$, even when one of these two users is identifiable from the published OSN data. In addition, we consider two schemes for developing the techniques of preserving users' relationship privacy: (1) removal-only scheme which only removes edges from the original OSN graph, and (2) insertion-only scheme which only inserts edges/vertices in to the OSN graph.

This chapter is organized as follows. In Section 3.1 we address our proposed relationship privacy problem with respect to attack model, privacy preservation model, the utility measurement and problem formulation. In Section 3.2, we propose three heuristic anonymization algorithms for protecting users' relationship privacy, followed by a comprehensive experimental study in Section 3.3 to demonstrate our algorithms performance. Finally, we conclude the chapter in Section 3.4.

3.1 Preliminaries

3.1.1 Attack Model and Privacy Guarantee



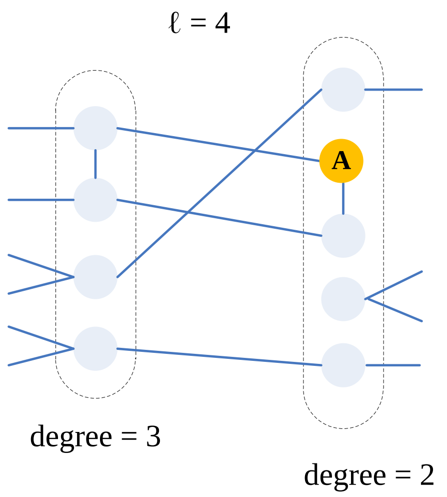


Figure 3.1: 4-degree anonymous graph   Figure 3.2: 4-diversity anonymous graph

We assume an attacker launches passive attack only by analyzing the published data, as opposed to active attacks defined in [8]. We do not consider active attacks in which an adversary can revise the topology of the OSN by creating dummy users and relationships before data publishing. In our attack model, the attacker attempts to compromise the relationship privacy between two target users, to see whether they have a sensitive relationship or not. The extra knowledge assumed to be equipped with by the attacker is the number of friends for each of the target users (i.e., the degree of the vertices corresponding to target users) in the OSN. This is common information publicly available on many OSN websites, like LinkedIn. Furthermore, in the worst case one of the target users can even be identified by the attacker based on some background knowledge which may be collected from the user's blog. Apparently, if the attacker can identify both of the target users from the published data, their relationship will be on the table, in which case we can do nothing to preserve the relationship.

The work in [46] has discussed how to preserve users' identity privacy under the attack model where the vertex degree of a target user is assumed to be available to the attacker. The authors argued that with the knowledge of a user's degree, the attacker may identify the target user even from an identity-removed OSN. To solve the problem, the k-degree anonymity model was proposed in [46], ensuring that each vertex shares the same degree with at least $k-1$ other vertices in the published OSN. However, such a model is insufficient to guarantee the relationship privacy we are concerned with in our model. A simple example is illustrated in Figure 3.1, where nodes are grouped in terms of their degrees. Suppose the attacker is curious about the relationship between user A and user B. In this 4-degree anonymous graph, if the attacker only knows user A's degree and user B's degree are four and one, respectively, he cannot successfully infer the existence of their relationship. However, under our attack model, if the attacker can identify user A somehow, although he is not able to identify user B solely with the knowledge of user B's degree, the sensitive relationship between the two target users has been exposed, due to the full connection of all vertices of degree one with user A. To defend against such an attack, we propose a new privacy anonymization model defined in Definition 3.1.1, adapted from the ℓ-diversity model which was initially developed in [48] for preserving tabular data privacy in the fields of databases.

A published network is ℓ-diversity anonymous if and only if given the degrees of any two users, the attacker cannot infer the existence of a relationship between them from the anonymized graph with the probability of greater than $1/\ell$, even when one of them is identifiable from the published social data.

Note that this problem definition is built based on but stricter than k-degree anonymity. In Definition 3.1.1, first of all, we need to ensure a user cannot be identified by his/her degree information, which is degree anonymity. One can see if a set of vertices in the published graph satisfy ℓ-diversity, there must be at least ℓ vertices sharing a unique degree, which is exactly required by degree anonymity model. Second, the ℓ-diversity model also ensures the relationship diversity between any two vertex groups each having vertices of the same degree. A toy example of a 4-diversity anonymous graph is given in Figure 3.2. In this graph, even when user A is exposed, the possibility of successfully inferring the relationship between user A and user B is 1/4, given user B's degree of three.

3.1.2 Utility Measures

As we mentioned in Section1, although completely removing the relationship information from the OSN data before publishing will protect users' privacy, it will significantly ruin the utility of the published data for the third-parties, who often want the relationship information for their data analysis. In fact, it is the third-parties who should specify data utility based on their applications, so it is hard for the OSN site owners to propose a generic metric to measure the utility of data and its loss caused by anonymization techniques.

Hence in this paper we use some topological measurements to study the impact of our proposed anonymization techniques on the utility loss of the published data. Particularly, we consider the number of edges/vertices removed/inserted in technique design. In addition, we provide some other measurements, such as the degree distribution and distance distribution between reachable vertices, through our experiments to be addressed in Section 3.3.

3.1.3 Problem Formulation

Given an arbitrary social network graph, $G : \langle V, E \rangle$, we intend to make the graph ℓ-diversity anonymous by graph-manipulation from two directions, (1) find a subgraph $G' : \langle V', E' \rangle$, where $V' = V$ and $E' \subseteq E$, and (2) create a supergraph $G^* : \langle V^*, E^* \rangle$, where $V^* \supseteq V$ and $E^* \supseteq E$. In addition to the result graph satisfying ℓ-diversity anonymity, the utility loss should be minimized, which is particularly quantified in terms of the number of changed vertices/edges changed as compared to the original graph.

3.2 Social Network Republishing Algorithms

In this section, we will first introduce our insight on ℓ-diversity anonymous graphs, which places a important role in our algorithm design, and then we will propose three techniques for anonymizing the users' relationship privacy in publishing OSN data. We start with a simple *Max-Subgraph-Basic* algorithm which is designed based on the intuition of starting with an empty graph and then gradually adding back a subset of edges in the original graph without violating ℓ-diversity. We shall point out the problem of Max-Subgraph-Basic which causes serious utility loss and motivates us to design an improved algorithm, called *Max-Subgraph.* Finally, we will propose our *Min-Supergraph* algorithm which belongs to the insertion-only scheme and constructs an ℓ-diversity supergraph by inserting dummy edges/vertices.

3.2.1 Insight on ℓ-Diversity Anonymous Graph

Given an ℓ-diversity anonymous graph, we know from Definition 3.1.1 that the number of vertices with a unique degree must not be less than ℓ, that is to say, if we suppose V_a is the set of vertices with degree of d_a, then $|V_a| \geq \ell$. Furthermore, the second critical point in Definition 3.1.1 - preventing the attacker from inferring the existence of a relationship between two users with confidence greater than $1/\ell$ even when one of the target users is identified - can be interpreted as the fact that in each group, the ratio of vertices that share a neighboring vertex to the total vertices in that group must not be greater than $1/\ell$. Suppose in V_a vertices linking to v_n (any vertex in the graph) form a set V'_a, then $|V'_a/V_a| \leq 1/\ell$.

Now we extend our vision to see what will happen if we further partition vertices which even have the same degree into smaller groups. In particular, we keep partitioning each group until each smaller group is reduced to be size of $|V_{a^\star}|$, where $\ell \leq |V_{a^\star}| < 2\ell$. In such a small group, the second point in Definition 3.1.1 can be interpreted as $|V'_{a^\star}| < 2$, which indicates that the ℓ-diversity model does not allow any two vertices in the same small group to share any vertex. In other words, edges across any pair of small groups should not share any vertices at their endpoints, which is exactly the definition of matching in Graph Theory. Matching is referred to as a set of edges without any common endpoints. We are inspired by this observation in designing our techniques for preserving OSN users' relationship privacy.

Note that the observation aforementioned is a sufficient but unnecessary condition to judge whether a given graph is ℓ-diversity anonymous. Specifically, if the vertices in a graph can be partitioned into small groups each having vertices of the

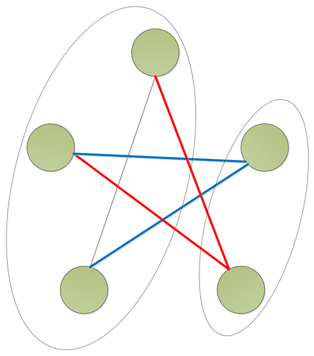


Figure 3.3: 2-diversity but not divisible graph

same degree, and also edges across any small groups form a matching, we ensure that it is an ℓ-diversity anonymous graph, but not all graphs of ℓ-diversity anonymity can be partitioned into such small groups with the matching property, for example, the graph shown in Figure 3.3. Since all vertices have the same degree, we can regard them as one group of size five. Note that each vertex is a common neighbor of two other vertices. Therefore, in a social network with such a topology, if one user is identified, an attacker has a chance of 2/4 to infer the relationship between the identified user and another target one, so it is 2-diversity anonymous. But apparently we can not split the vertices further into smaller groups of size two or three without violating the matching property, because in doing so any two vertices in a group will share a neighbor in the other group, as shown in Figure 3.3. Although the two features, small group and matching only form a sufficient condition for an ℓ-diversity anonymous graph, they at least give us a direction of how to convert an arbitrary graph to a graph with ℓ-diversity anonymity.

3.2.2 Max-Subgraph-Basic Algorithm (MaxSubB)

We start by describing Max-Subgraph-Basic (in short, MaxSubB), a simple version of the subgraph-publishing algorithm. The pseudocode for MaxSubB is given in Algorithm 1. Given an arbitrary graph, $G : \langle V, E\rangle$, the basic idea of the MaxSubB is to start from a graph with all isolated vertices of degree zero, $G_0 : \langle V_0, E_0\rangle$, where $V_0 = V$ and $E_0 = \emptyset$, and then gradually add back a subset of edges from the original graph for publishing, while ensuring ℓ-diversity at *anytime* during the edge-recovery process. Such a procedure is conducted iteratively, where each iteration consists of three steps: *Matching-based Partitioning*, *Partition Adjustment*, and *Edge Recovery*, which are detailed as follows.

Step 1. (Matching-based Partitioning): We first arbitrarily select from the original graph a maximal matching, which is referred to as a maximal set of edges without any common endpoints. We start with randomly selecting an original edge, and then remove its two endpoints and their associated edges from the original graph. This edge-picking-up procedure continues until no edge is left. Then, all selected edges form a maximal matching. Based on the matching, all vertices can be partitioned into at least two groups, which are called "matched" groups, each containing one of endpoints of each edge in the matching. If there are vertices left unmatched, they constitute the third group, named "unmatched" group.

Step 2. (Partition Adjustment): Note that all vertices in each of the groups formed by the maximal matching have the same degree and do not share any neighbor through the edges in the matching. Thus, the graph formed by the matched edges will satisfy ℓ-diversity as long as each group contains at least ℓ vertices. The step of partition adjustment mainly checks whether the partition formed by the matching can be adjusted so that all groups have size at least ℓ. If all groups have at least ℓ vertices, we go to the next step directly. If the unmatched group has a size greater than ℓ, but each matched group has less than ℓ vertices, while the sum of them has vertices greater than ℓ, we combine the two matched groups into one, and stop looking for additional matching in it from then on. If the size of the unmatched group is smaller than ℓ, we may adjust the matching by removing the minimum number of matched edges such that the number of unmatched vertices is at least ℓ. Note that at the same time of the adjustment, the number of vertices in each matched group decreases. Therefore, we need to ensure after the adjustment each matched group has at least ℓ vertices, or the union of the two matched groups has vertices not less than ℓ. In the latter case we merge the two groups into one and stop iteratively checking additional matching in it. For all other situations, the matching fails and is aborted without any edge publishing.

Step 3. (Edge Recovery): For each edge maintained in the matching after the procedure of partition adjustment, we add it back to the graph for publishing. The previous steps will continue iteratively in each of the groups which are newly formed and not terminated, until no matching can be found to further split the group into smaller groups of size not less than ℓ.

To illustrate the process of executing MaxSubB, an example is given in Figure 3.4(A-D), where $\ell = 2$. Based on a maximal matching {(1,2), (3,4), (5,6), (7,8)}, all vertices are partitioned into three groups, $match_1$ $\{v_1,v_3,v_5,v_7\}$, $match_2$ $\{v_2,v_4,v_6,v_8\}$, and *unmatch* $\{v_9,v_{10}\}$, as shown in Figure 3.4(B). Since each group has at least ℓ vertices, all edges in the matching are published. For $match_1$ and $match_2$, because they contain vertices greater than ℓ, we continue look for a maximal matching in each of them. For $match_1$, all vertices can be matched further, which leads to another partition, $\{v_1,v_3\}$ and $\{v_5,v_7\}$. Furthermore, one more matching is found between v_5 and v_7. For $match_2$, since no matching exists, the iteration terminates without additional edges being published. The graph manipulated by MaxSubB

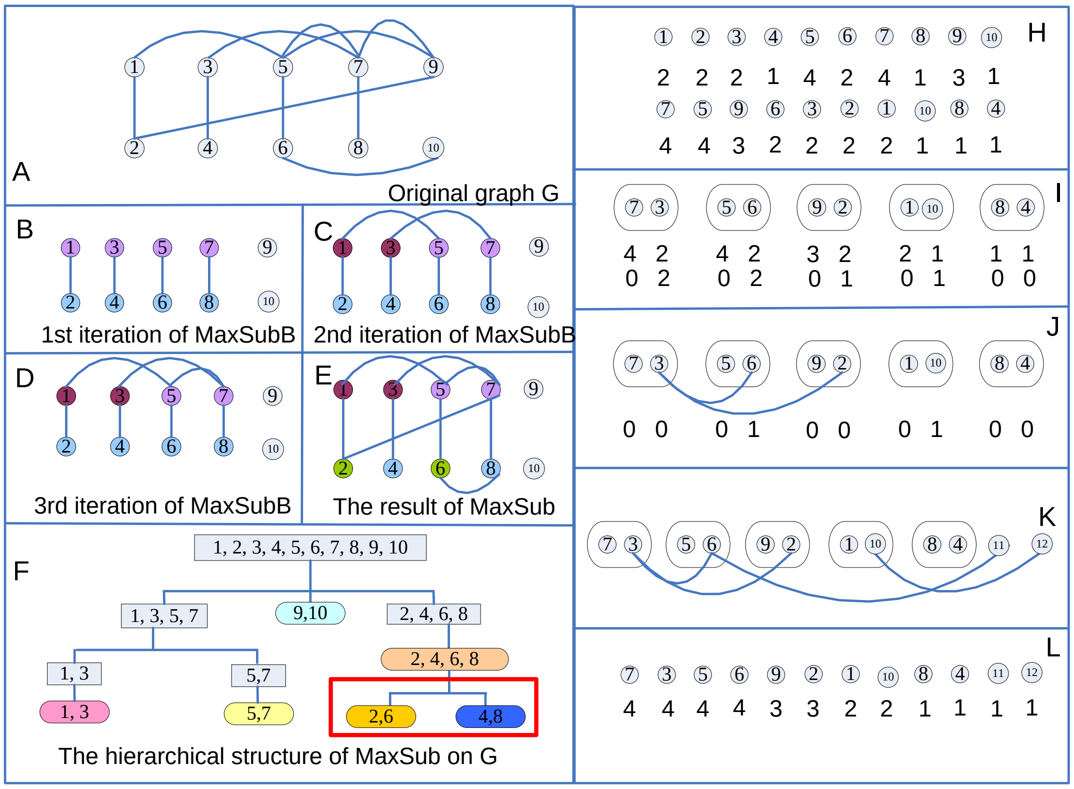


Figure 3.4: An example for our algorithms

is given in Figure 3.4(D). Note that the procedure of splitting the original vertices into smaller groups by iteratively matching and partitioning can be presented as the growth of an upside-down tree which is rooted as the original vertex set. The leaf-nodes of the tree are groups which have size equal to or greater than ℓ, but are not divisible by additional maximal matchings. The corresponding tree to the illustrated example is presented in Figure 3.4(F) without the branches circled in red. Each leaf-node of the tree forms a set of equal-degree vertices in the graph published by MaxSubB.

3.2.2.1 Correctness and Complexity Analysis

Theorem 1. *Given an arbitrary graph, its subgraph published by MaxSubB satisfies ℓ-diversity.*

Proof. We will prove Theorem 1 from two aspects of ℓ-diversity anonymity: (1) degree anonymity - each final group contains at least ℓ vertices of the same degree, and (2) relationship diversity - neither of two vertices in a group share a neighboring vertex in the published graph. One can see that the combination of the two properties guarantees ℓ-diversity anonymity as defined in Definition 3.1.1.

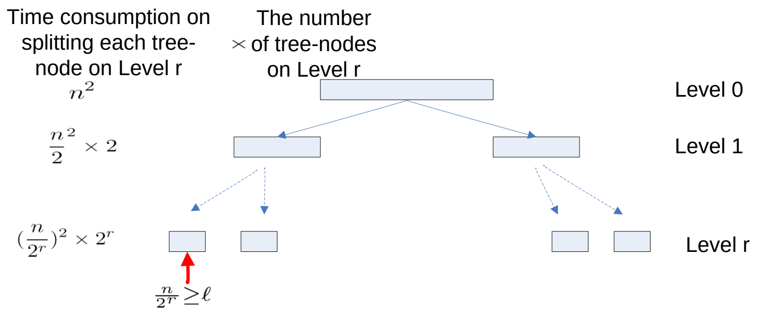
Degree anonymity: The matching-based partitioning procedure of MaxSubB guarantees that, starting from the initial all-isolated-vertex graph, each iteration produces groups with vertices of equal degree. Besides, the step of partition adjustment assures that each group in the output has at least ℓ vertices. Therefore, MaxSubB achieves degree anonymity.

Relationship diversity: We prove each of leaf-node groups formed by MaxSubB satisfies the relationship diversity by contradiction. Suppose that two vertices in the same group share a neighbor v, then both of their edges connecting with the common neighbor must be added at the same iteration when v is separated from those two vertices by a matching. This contradicts the fact that all added-back edges in one iteration forms a matching. Thus, all leaf-node groups published by MaxSubB satisfies the relationship diversity requirement.

Accordingly, the graph finally published by MaxSubB is ℓ-diversity anonymous.

□

Theorem 2. *Given an arbitrary graph of n vertices, the time complexity of MaxSubB $T(n) = O(n^2)$.*



Time consumption on splitting each tree-node on Level r
$\times$ The number of tree-nodes on Level r
n^2
$\frac{n^2}{2} \times 2$
$(\frac{n}{2^r})^2 \times 2^r$
Level 0
Level 1
Level r
$\frac{n}{2^r} \geq \ell$

Figure 3.5: Worst case for MaxSubB

Proof. Basically, the complexity of finding a maximal matching is $O(|g|^2)$ in a group with $|g|$ vertices. In the worst case of MaxSubB, all vertices can be matched in each group, so that each matched group has the maximum vertices to participate the next iteration of the matching-partitioning procedure. Therefore, this procedure can be presented in a binary tree, as shown in Figure 3.5. The time complexity of the MaxSubB can be derived from summing the time consumption on partitioning each tree-node on all tree levels, as formalized in Equation 3.1, where n is the number of original vertices. $\square$

$$T(n) = \sum_{r=0}^{r=\lfloor \log_2 \frac{n}{\ell} \rfloor} (\frac{n}{2^r})^2 \times 2^r = O(n^2) \tag{3.1}$$

3.2.2.2 Observation for Further Improving MaxSubB

Given a complete graph G_c with $2^p\ell$ vertices, where p is an arbitrary positive integer, we consider its maximum ℓ-diversity subgraph $G_{c\ell}^{max}$. First, we will introduce Theorem 3 in the proof of which we present the construction of $G_{c\ell}^{max}$. Then we will compare $G_{c\ell}^{max}$ with the graph published by MaxSubB with respect to the number of the published edges, highlighting our observation for further improving MaxSubB to more utility of published data which is particularly measured by the number of the original edges published.

Theorem 3. *Any ℓ-diversity graph with $2^p\ell$ vertices must have edges less than the total number of edges in $G_{c\ell}^{max}$.*

Proof. Given an ℓ-diversity graph with $2^p\ell$ vertices, no vertices can have degree greater than 2^p; otherwise, it requires more than $2^p\ell$ vertices to form a vertex group of size ℓ with degree of 2^p to meet the ℓ-diversity constraint. Therefore, the maximum possible number of edges in an ℓ-diversity graph with $2^p\ell$ vertices is $2^{2p-1}\ell$.

$G_{c\ell}^{max}$ with $2^p\ell$ vertices can be constructed as follows: first, we divide all vertices into 2^p unit groups each consisting of exactly ℓ vertices; then we find a perfect matching saturating all vertices in each unit group as well as between any pair of unit groups. As such, each vertex has two types of edges: one intra-edge connecting vertices in the same unit group and $2^p - 1$ inter-edges connecting vertices in different unit groups. Because all vertices have the same degree of 2^p, this graph satisfies ℓ-diversity with exactly $2^{2p-1}\ell$ edges. Therefore, $G_{c\ell}^{max}$ achieves the maximum possible number of edges, thus reaching an upper bound for all ℓ-diversity graphs with $2^p\ell$ vertices. □

Given G_c with $2^p\ell$ vertices, we now compare $G_{c\ell}^{max}$ with the output graph of G_c by MaxSubB. For the sake of simplicity, we assume ℓ to be even, which however is sufficient to highlight the point from which we could improve MaxSubB. Based on G_c, MaxSubB conducts a maximum matching saturating all vertices in each iteration until the size of each group is reduced to ℓ. Since ℓ is assumed to be even, all ℓ vertices in each group can be paired up by their edges. Again consider the execution of MaxSubB as the growth of a tree. In this case, we have a perfectly balanced binary tree where each level corresponds to one iteration, as depicted in Figure 3.5. Note

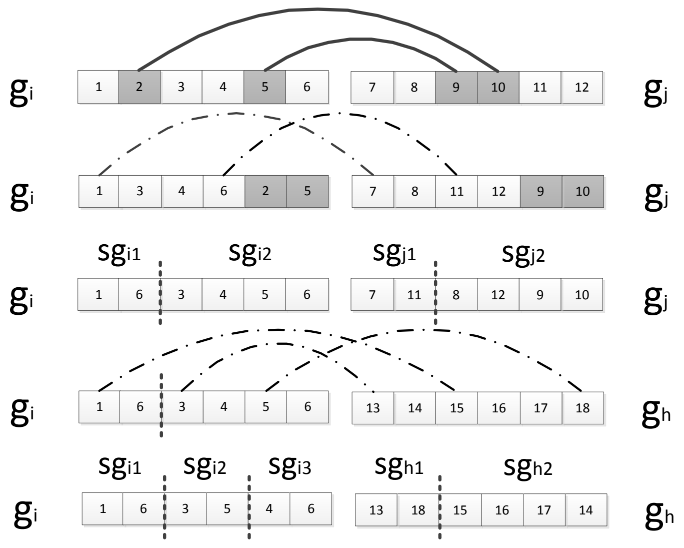


Figure 3.6: An example for the MaxSub algorithm

that the number of edges recovered in each level is always $2^{p-1}\ell$. Thus, the total number of edges recovered by MaxSubB is $2^{p-1}\ell p + 2^p\ell/2$, where $2^p\ell/2$ is the number of edges recovered between vertices in the same group. One can see that the result graph of MaxSubB has fewer edges than $G_{c\ell}^{max}$.

The reason behind the less-than-optimal utility performance of MaxSubB is its design of never recovering any additional edges between two groups of vertices after they are split. For example, given the graph published by MaxSubB in Figure 3.4(D), we could actually find another maximal matching, $\{(v_2,v_9), (v_6,v_{10})\}$, between two leaf-node groups, $\{v_2,v_4,v_6,v_8\}$ and $\{v_9,v_{10}\}$. Such a matching leads to an additional splitting of a leaf-node group, as marked in red circle in Figure 3.4(F), which turns out to publish one more original edge without violating the ℓ-diversity constraint.

3.2.3 Max-Subgraph Algorithm (MaxSub)

Motivated by the aforementioned observation, we design our algorithm Max-Subgraph (MaxSub) which improves upon MaxSubB by publishing more original edges across the leaf-node groups. The algorithm MaxSub first follows the same procedure as MaxSubB, and then finds additional maximal matchings across leaf-node groups in the result graph of MaxSubB to publish more original edges, as shown in the pseudocode of MaxSub in Appendix 1. Basically, the following three steps are performed for every pair of leaf-node groups, g_i and g_j, produced by MaxSubB.

Step 1. (Inter Matching): An arbitrary maximal matching is first selected between g_i and g_j subject to one constraint: if some vertices have been matched together across the two groups previously, they are not considered any more in the new matching. This constraint guarantees that the finally published edges across any pair of groups are still a matching, thereby satisfying ℓ-diversity anonymity. An example is illustrated in Figure 3.6, where $\ell = 2$. Suppose (v_2, v_{10}) and (v_5, v_9) are two edges published in MaxSubB. So the four vertices would be marked as "unmatched" vertices, and thus excluded from the new matching between g_i and g_j.

Step 2. (Partition Adjustment): Based on the new matching, g_i and g_j are further partitioned into small groups which contain matched and unmatched vertices, respectively. Partition adjustment will be conducted by removing edges from the matching, if necessary, to ensure each newly formed group does not have fewer than ℓ vertices. As shown in Figure 3.6, due to the matching of two edges (v_1, v_7) and (v_6, v_{11}), both g_i and g_j are divided into two small groups. Later on, upon checking the possibility of publishing edges between g_i and another group, say g_h, if a maximal matching is found between g_h and the entire group g_i after excluding matched vertices in MaxSubB, we then check each small group in g_i to see whether the newly matching can split the small group further. In our example, three edges, (v_1, v_{15}), (v_3, v_{13}) and

(v_5, v_{18}), form the new matching between g_i and g_h. However, since only one vertex is matched in sg_{i_1}, further splitting sg_{i_1} will violate the constraint of group size in ℓ-diversity anonymity. Therefore, we abort the edge in the matching. For sg_{i_2}, there are two edges matched, so we split sg_{i_2} into two smaller groups further. Meanwhile, vertices in g_h are partitioned into two groups.

Step 3. (Edge Recovery): The edges remained in the inter-matching will be published finally. Then, we repeat Steps 1-3 between any pair of leaf-node groups formed by MaxSubB.

3.2.3.1 Correctness and Complexity Analysis

Theorem 4. *Given an arbitrary graph, its subgraph published by MaxSub satisfies ℓ-diversity.*

Proof. We prove the graph published by MaxSub subject to ℓ-diversity constraint with respect to degree anonymity and relationship diversity.

Degree anonymity: The starting point of MaxSub is the output of MaxSubB, where each leaf-node group contains only vertices with the same degree. The inter-matching step in MaxSub splits the leaf-node groups into smaller groups in each of which vertices still share the same degree. Furthermore, the size of each smaller group is at least ℓ. Therefore, the degree anonymity holds for MaxSub.

Relationship diversity: We prove MaxSub could achieve relationship diversity by contradiction. Suppose two vertices in a smaller group s has a common neighbor v from another smaller group s'. Since MaxSub only conducts one matching between any pair of leaf-node groups generated by MaxSubB, one of the two edges must have been added in MaxSubB. However, this contradicts the design of MaxSub where vertices of the two leaf-node groups which have been matched together in MaxSubB are excluded from any new matching across the two groups. Thus, the result graph by MaxSub is ℓ-diversity anonymous. □

Theorem 5. *Given an arbitrary graph of n vertices, the time complexity of MaxSub $T(n) = O(n^2)$.*

Proof. Let h be the maximum size of a leaf-node group generated by MaxSubB. The time complexity for MaxSub to process each pair of leaf-node groups is $O(h^2)$. Since the size of each leaf-node group is at least ℓ, the number of pair-wise matchings across groups is $(n/\ell)(n/\ell - 1)/2$ at most. Thus, the processing of the output of MaxSubB takes $O(n^2 \cdot h^2/\ell^2)$. In addition, since MaxSub leverages MaxSubB as the first part of the algorithm, its time complexity is $O(n^2)$. Consequently, the overall time complexity of MaxSub is $O(n^2)$. □

Essentially, the fact of more original edges being published in MaxSub mainly depends on the success of inter-group matchings. In particular, MaxSub works most effectively when the leaf-node groups generated by MaxSubB are large, in which more matchings can be found. This is exactly our observation after applying MaxSubB to our data sets, as shown in Table 3.1, which we shall further elaborate in Section 3.3.

3.2.4 Min-Supergraph Algorithm (MinSuper)

MinSuper is a two-step algorithm of expanding a graph G to its supergraph of ℓ-diversity anonymity: 1) forming ℓ-groups, where ℓ-group is referred to as a group of ℓ vertices without sharing any neighboring vertex in G; 2) normalizing vertex degrees in each ℓ-group. The pseudocode of MinSuper is given in Algorithm3. To illustrate MinSuper, we shall use the same example (with $\ell = 2$) as that for MaxSubB and MaxSub, as shown in Figures 3.4(H-L).

Step 1. (Forming ℓ-groups): All vertices in G are first sorted in a non-ascending order of their degrees, as shown in Figure3.4(H). From the first vertex in the ordered sequence, MinSuper looks for ℓ vertices with no common neighbors to form an ℓ-group by checking sequentially and skipping the conflicting ones. If a group includes only fewer than ℓ non-conflicting vertices after going through the whole sequence, dummy vertices with degree of zero are padded into it aiming at ensuring group size not less than ℓ. In the running example, the final ℓ-groups are listed in Figure 3.4(I).

Step 2. (Normalizing each ℓ-group): Each vertex is first labeled with the difference between its degree and the maximum vertex degree in its group, as shown in Figure 3.4(I). Then, dummy edges are inserted across any pair of the ℓ-groups, particularly between vertices of nonzero degree difference which have not been linked with any other vertex in each other's group. Such a constraint in the procedure of injecting dummy edges essentially requires that both original and dummy edges across any pair of ℓ-groups do not share vertices at their ends, thus meeting the feature of matching for ℓ-diversity anonymity. For a group which still has at least one vertex with nonzero degree difference after pairing up with any other group, dummy vertices and edges will be inserted into the graph for completing the degree normalization in that group. In particular, dummy edges will be constructed between those dummy vertices and real vertices of nonzero degree difference aiming at reducing the degree difference to zero. For example, as shown in Figure 3.4(J), v_{11} and v_{12} are created for normalizing the degrees of v_6 and v_{10}. One can see that the procedure of sorting in the first step ensures that vertices in the same ℓ-group have similar degrees, thus minimizing the number of dummy edges required for degree normalization.

3.2.4.1 Correctness and Complexity Analysis

We start by demonstrating the necessity of inserting dummy vertices in MinSuper to create a supergraph with ℓ-diversity anonymity by Theorem 6.

Theorem 6. *Solely relying on the insertion of dummy edges may not work to expand an arbitrary graph to its ℓ-diversity anonymous supergraph.*

Proof. We know that given an ℓ-diversity anonymous graph with $\alpha\ell$ vertices, no vertices can have degree greater than α; otherwise, it requires the graph have more than $\alpha\ell$ vertices to ensure there are at least ℓ vertices with a degree greater than α. Therefore, if an arbitrary graph with $\alpha\ell$ vertices, G, originally has at least one vertex of degree greater than α, solely depending on dummy edges will not help to convert the graph to be ℓ-diversity anonymous. $\square$

Theorem 7. *Given an arbitrary graph, its supergraph published by MinSuper is ℓ-diversity anonymous.*

Proof. First of all, forming ℓ-groups guarantees the size of each group of ℓ. Secondly, due to the degree normalization procedure, all vertices in each group share the same degree. Finally, because of the conflict-checking in forming ℓ-groups as well as the group-crossing matchings when normalizing vertex degrees in each group, we ensure no vertices in the same group share any neighboring vertex. Therefore, the graph published by MinSuper is guaranteed to be ℓ-diversity anonymous. $\square$

Theorem 8. *The time complexity of MinSuper is $O(n^3)$ for a graph of n vertices.*

Proof. The time complexity of vertex sorting is $O(n\log(n))$ by Merge Sort Algorithm. In the worst case of forming ℓ-groups, where each vertex conflicts with all other vertices in a complete graph, it takes each vertex $O(n^2)$ time for checking, and thus $O(n^3)$ for all vertices. For degree normalization, each group requires at worst to check all other groups for edge construction. The total number of groups is n in the worst case of a complete graph, and the checking between any pair of groups requires $O(\ell^2)$. Therefore, the time complexity of normalizing degrees is $O(\ell^2 \times n^2) \approx O(n^2)$. As a result, the time complexity of MinSuper is $O(n^3)$. □

3.3 Experimental Study

To evaluate our algorithm performance, we conducted a comprehensive set of experiments on both synthetic and real-world data sets. In this section, we will mainly discuss utility loss of the published social data caused by our anonymization techniques with respect to both theoretical and empirical measurements. In addition, we will investigate the impact of social network topology on our algorithm performance. All of our proposed algorithms, MaxSubB, MaxSub and MinSuper, were implemented on MATLAB. Besides, we used a software package called Pajek [1] for analyzing the topological properties of anonymized graphs.

3.3.1 Data Sets

1) Real-World Data sets:

- **Co-author graph**: The co-authors data set consists of 7955 authors of papers in database and theory conferences. It is available at the collection of Computer Science Bibliographies (http://liinwww.ira.uka.de/bib-liography). The co-authors graph is formed by constructing undirected an edge between any authors who have co-authored papers. Totally, there are 10055 edges in the graph.
- **SIGMOD graph**: The data set was crawled from DBLP (http://dblp.uni-trier.de/xml/) on December 5, 2009, and contains co-authorship information

for all previous SIGMOD conferences. There are 3791 vertices (i.e., authors) and 10003 edges (i.e., co-authorships) in the graph.

2) Synthetic Data sets:

- **Erdos Renyi Model (ER)**: ER is a classic random graph model, which randomly constructs edges in the graph to connect nodes. We generated a random graph with 5000 vertices and the average degree of 5 using the software Pajek [1], which already implemented the ER model.
- **R-MAT Model (R-MAT)**: We used the R-MAT graph model [15] to generate another synthetic data set. The basic idea behind R-MAT is to recursively subdivide the adjacency matrix of a graph into four equal-sized partitions, and then distribute edges within these partitions with an unequal probabilities, a, b, c and d. R-MAT can generate graphs characterized by scale-free and small-world features, the two most important properties for many real-world social networks. We followed the configuration given in [71], and set the input parameters as follows, $a = 0.45$, $b = 0.15$, $c = 0.15$ and $d = 0.25$, to generate social network graphs with 5000 vertices and the average vertex degree of 5.

3.3.2 Experimental Results

We will analyze the utility loss of the anonymized data in terms of both theoretical and empirical measurements: (1) *Theoretical measurements* - the number of changed edges/vertices as compared to the original graph. (2) *Empirical measurements* - the number of pairs of unreachable vertices, the average distance between any pair of reachable vertices (i.e., the length of the shortest path between each pair of vertices), the distribution of vertex degrees, and the distance distribution of reachable vertex pairs.

3.3.2.1 Theoretical Metrics

Figures 3.7(a-d) show that in our four data sets the number of fake vertices created by MinSuper is quite small, as compared to the number of the original vertices. In Figures 3.8(a-d), MaxSub publishes more original edges than MaxSubB does, which is consistent with our algorithm design - MaxSub is extended from MaxSubB by returning more group-crossing edges. For MinSuper, as compared to the large number of the original edges, only a few fake edges are constructed so that the number of edges in the graphs published by MinSuper is almost the same as the number of original edges.

In addition, we can also observe from these figures *the tradeoff between privacy preservation and utility loss*. Technically, according to Definition 3.1.1, the larger the ℓ value, the securer the privacy preservation; however, a larger ℓ will incur more utility loss in the published graph, because it blocks the procedure of further splitting vertices into smaller groups to and prevents more edges from being published. From Figures 3.8(a-d), one can see the plot of MaxSub decreases slightly with the value of ℓ. However, the plot of MaxSubB hardly changes with ℓ. The reason is that as shown in Table 3.1, the average size of a leaf-node group is large, as compared to ℓ, so that the change of a small ℓ value will not influence substantially on the matchings in MaxSub, thus the number of edges being similar. For MinSuper, the plot rises up with the value of ℓ. This is because a nonconflicting group of a larger size makes a larger difference between the maximum and minimum vertex degree in the group greater, which requires more fake edges to be created for degree normalization.

3.3.2.2 Empirical Metrics

Degree distribution: In Figures 4.5(a-d), the degree distribution of the graph published by MaxSub is more similar to the distribution of the original graph than the distribution of the graphs anonymized by MaxSubB is. MinSuper shows the best result in which its degree distribution almost overlaps with the original distribution.

Average distance between any pair of reachable vertices and the number of pairs of unreachable vertices: The results relevant to average distance between two vertices and the number of pairs of unreachable vertices are presented in Figures 3.10(a-d) and Figures 3.11(a-d), respectively. For data sets of ER, R-MAT and SIGMOD, not only the average distance but also the number of pairs of unreachable vertices in the graphs published by MaxSubB is much larger than the two measurements in the graphs anonymized by MaxSub. For Co-Author data set, although the two plots for MaxSubB and MaxSub intersect at a certain point in Figure 3.10(c), it does not indicate that the connectivity of the graph anonymized by MaxSubB is better than that published by MaxSub with varying ℓ values after their intersection point. Because from Figure 3.11(c), one can see the number of pairs of unreachable vertices in MaxSubB is still much larger than that in MaxSub, which means the poor connectivity of the graph published by MaxSubB. Consequently, the two metrics, the number of pairs of unreachable vertices and their average distance, have to be combined together to evaluate our algorithm performance. For MinSuper, both of the two metrics are similar to those in the original graphs, which shows a good maintenance on the utility of publishing data.

Distance distribution: In Figures 3.12(a-d), we can still see that distance distribution of the graphs published by MinSuper presents the best performance, almost

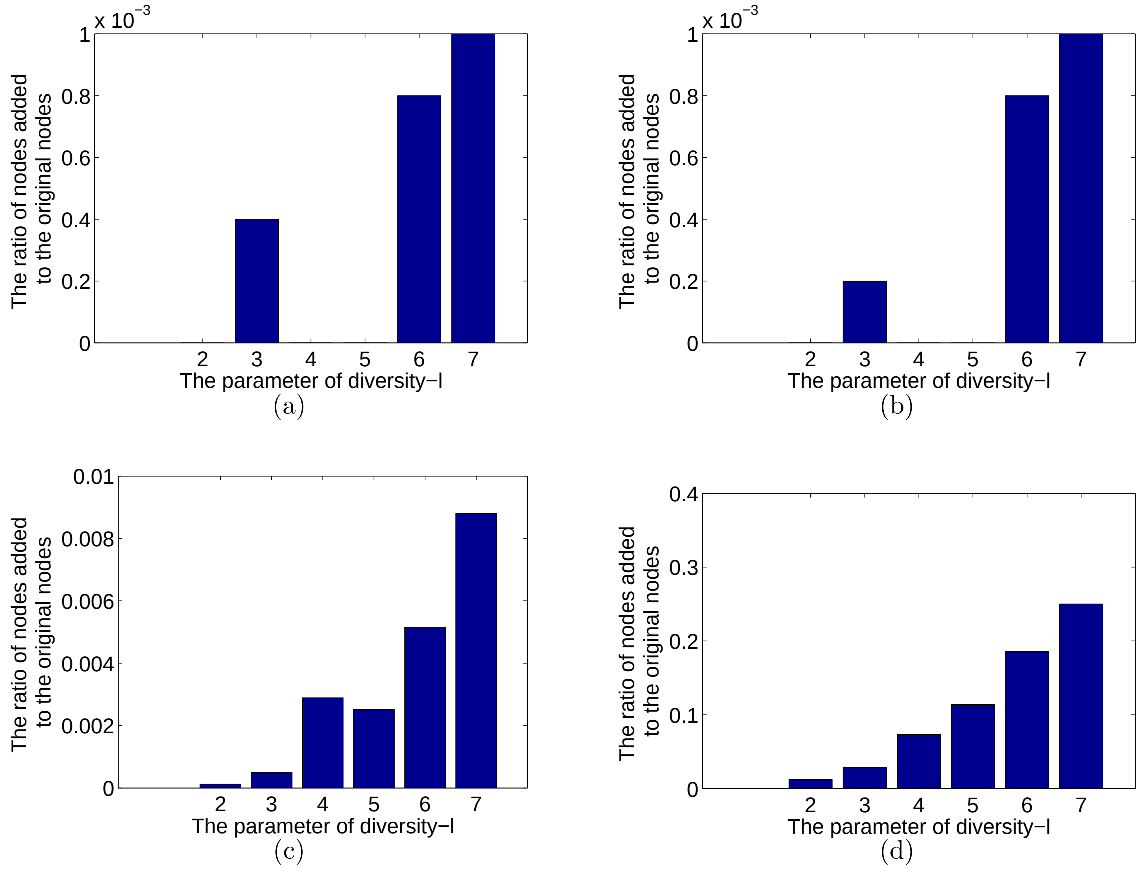


Figure 3.7: The ratio of dummy nodes to the original nodes in MinSuper (a) ER (b) R-MAT (c) Co-Author (d) SIGMOD

overlapping with that of the original graphs. In addition, the curve of MaxSub is more similar to the curve of the original graph than that of MaxSubB is.

3.3.3 The Influence of Network Topology

We investigated the influence of social network topologies on our algorithm performance with respect to two aspects, changing average vertex degree and network scalability. We conducted our experiments on synthetic social network graphs generated according to R-MAT model [15].

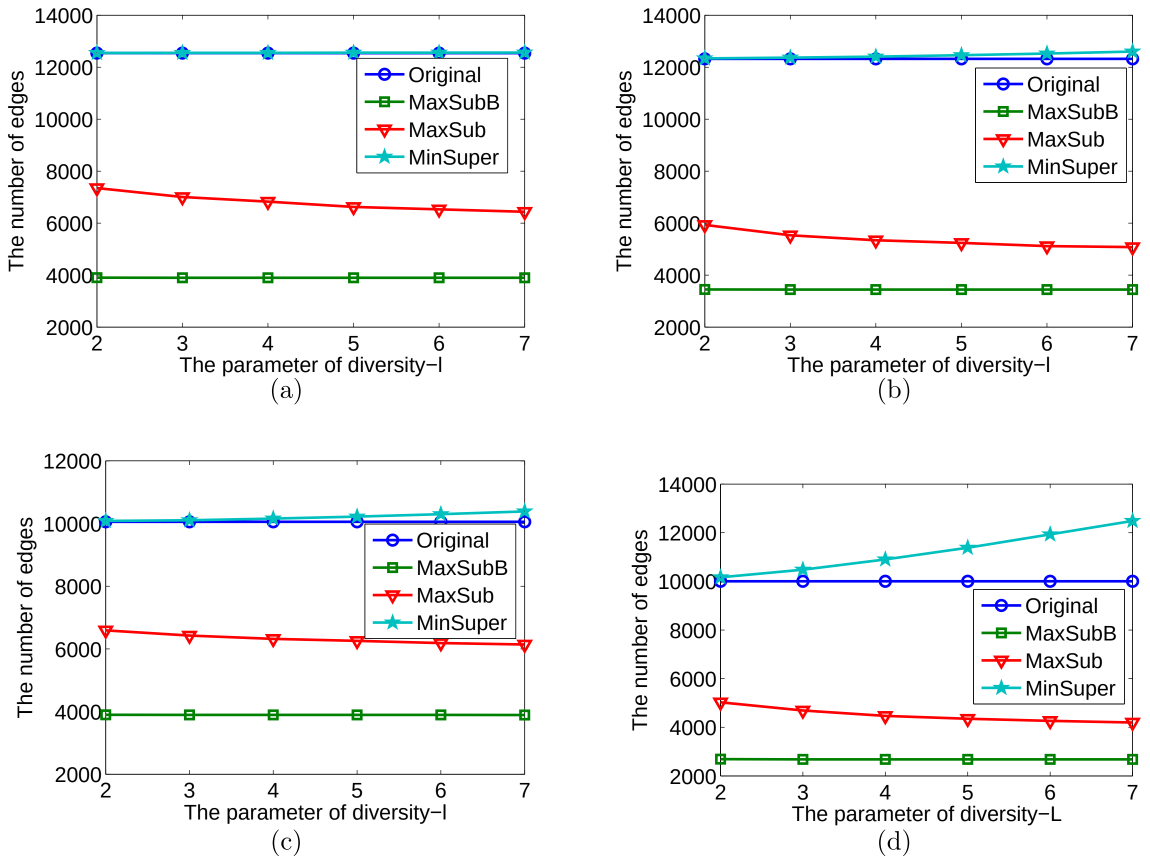


Figure 3.8: The number of published edges (a) ER (b) R-MAT (c) Co-Author (d) SIGMOD

3.3.3.1 Average Vertex Degree

In this group of experiments, we created some synthetic data sets based on R-MAT model. We set $n = 3500$ and $\ell = 5$, and increased the average vertex degree from 2 to 16 with the increment of 2. First, one can see from Figure 3.13(a) that despite the increasing of the average vertex degree, the number of dummy vertices created in MinSuper is quite small.

In Figure 3.13(b), increasing the average vertex degree leads to enlarging the difference of the number of edges in the graph anonymized by MinSuper and the

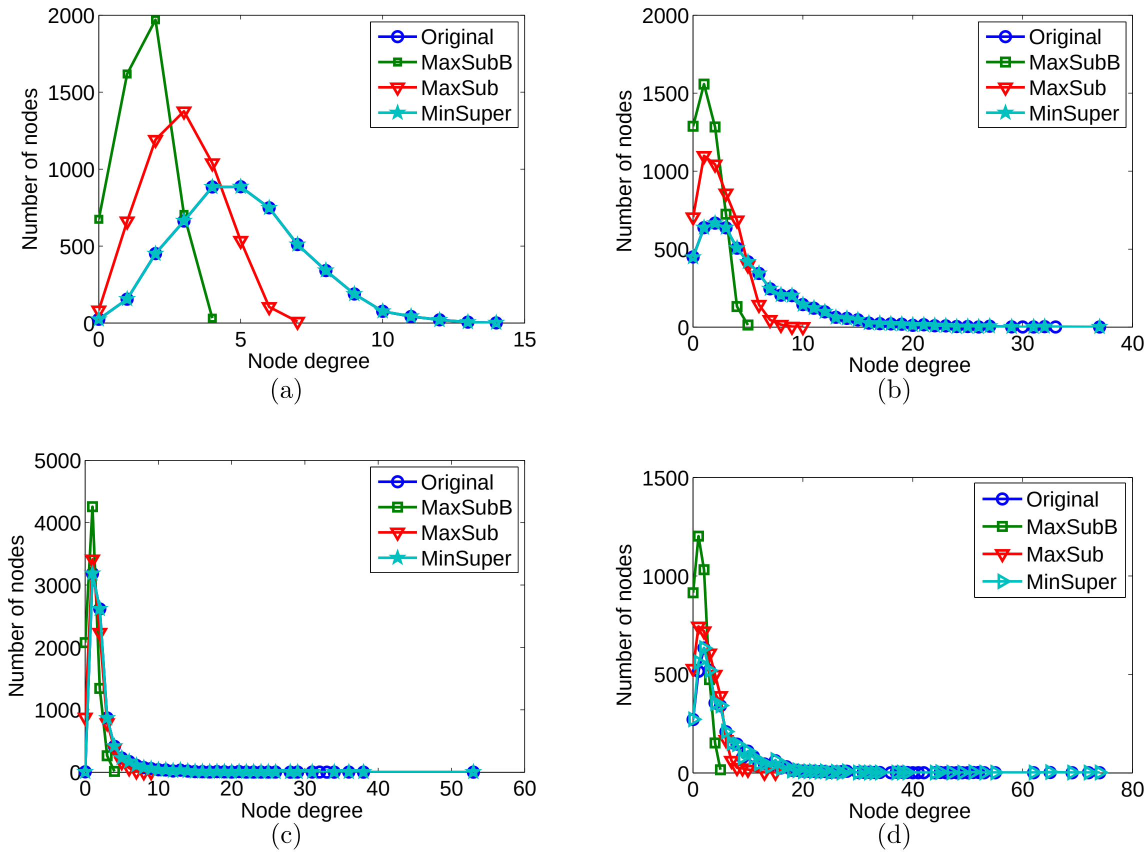


Figure 3.9: The degree distribution in the anonymized graphs, when ℓ=2 (a) ER (b) R-MAT (c) Co-Author (d) SIGMOD

number of the original edges. The reason is that given a fixed number of vertices, a larger average degree implies a vertex shares more neighbors with other vertices, which leads to a higher possibility for the vertex to conflict with other vertices. Therefore, in the phase of forming an ℓ-group, more vertices may need to be checked and may even select vertices of very low degree. In that case, the degree difference in the group is very large, therefore, more fake edges are needed for degree normalization.

Additionally, as shown in Figure 3.13(b), with the average vertex degree the plot of MaxSub increases much slower than the plot of the original graphs, which indicates more utility loss. The reason is although there are more edges in a graph

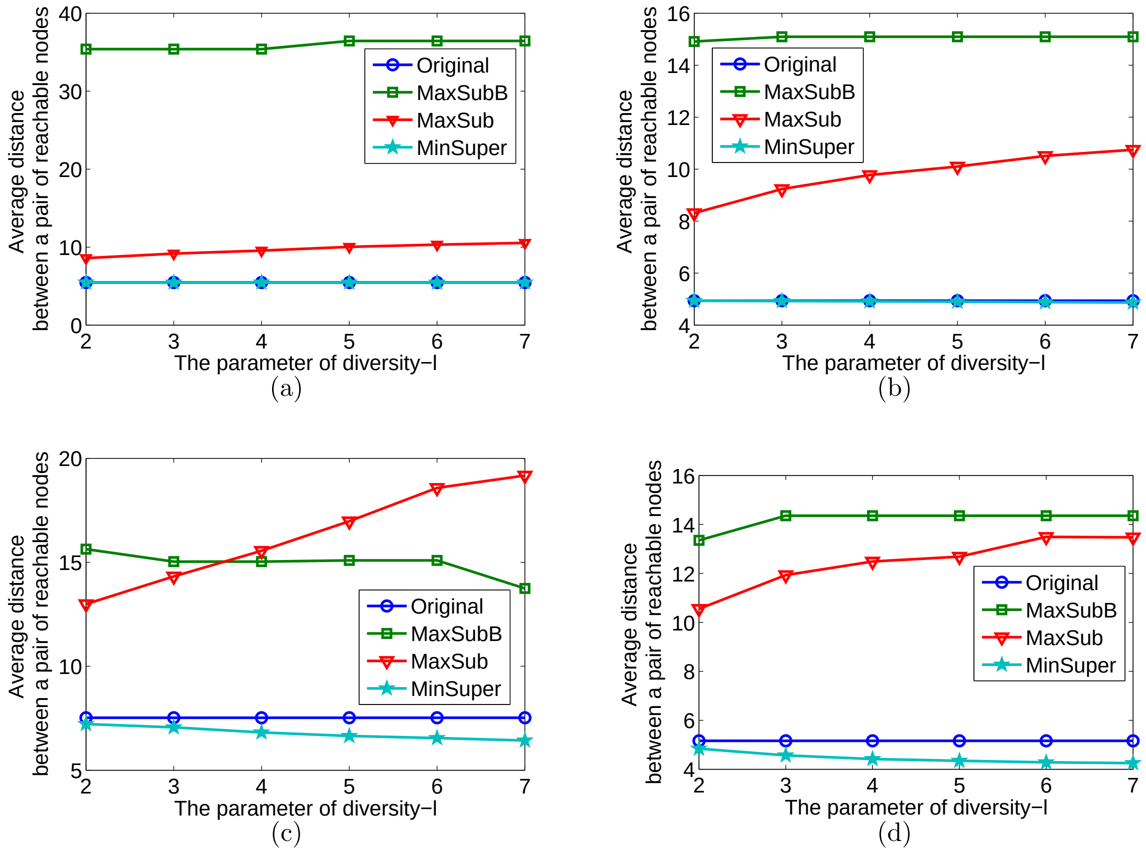
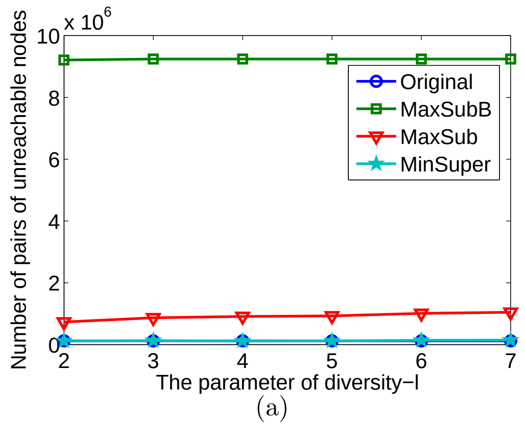


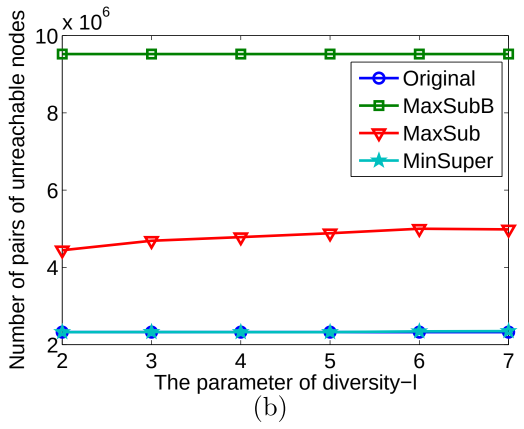
Figure 3.10: The average distance of any reachable vertices in the anonymized graphs (a) ER (b) R-MAT (c) Co-Author (d) SIGMOD

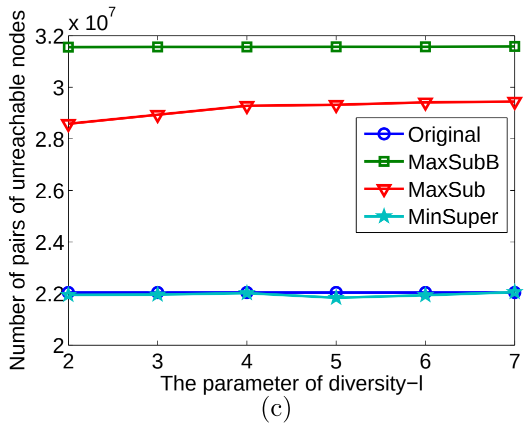
with a higher average vertex degree, since the number of vertices are fixed, and the topological features of social networks (e.g., small-world, scale-free) are still presented in the graphs, thus, the result of maximal matching will not be influenced greatly. Therefore, the plot for MaxSub does not increase so fast as that for the original graphs. MaxSub performs better than MaxSubB, as we expected.

3.3.3.2 Network Scalability

We set the average vertex degree to 3 and ℓ to 5, while varying the number of original vertices from 500 to 7500 with the increment of 500. The number of







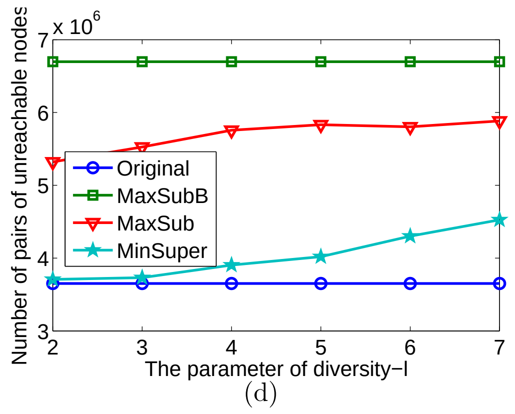


Figure 3.11: The number of unreachable vertex pairs in the anonymized graphs (a) ER (b) R-MAT (c) Co-Author (d) SIGMOD

fake vertices created for anonymization is quite small, regardless of ℓ, as shown in Figure 3.14(a).

In Figure 3.14(b), the plots of MinSuper and the original graphs almost overlap, and with the increasing of the network scalability, the number of edges published by MaxSub and MaxSubB also increases tangibly, due to the impact of the scalability on the maximal matchings. Specifically, given an average vertex degree, the larger the network scalability, the more edges will be involved in the matchings, which leads to an increase of edges published by MaxSubB and MaxSub. Furthermore, the group-

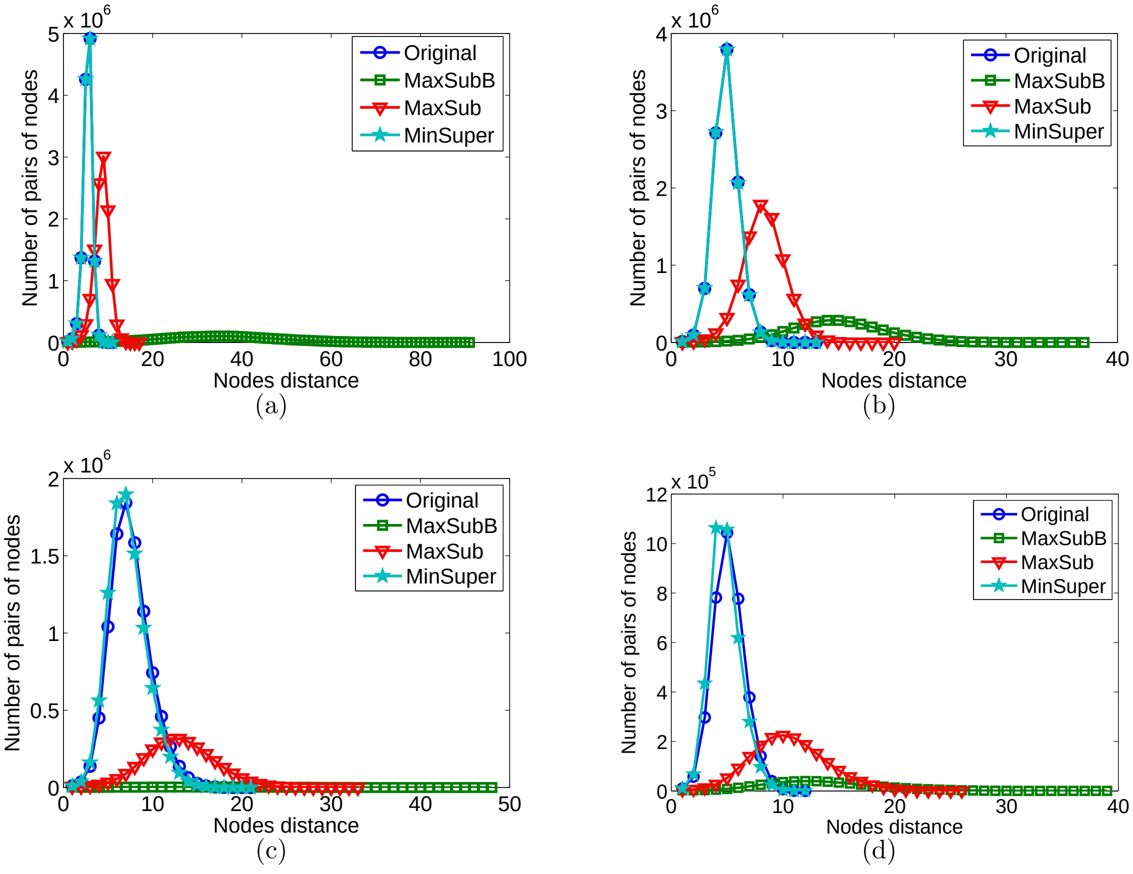


Figure 3.12: The distance distribution in the anonymized graphs, when ℓ=2 (a) ER (b) R-MAT (c) Co-Author (d) SIGMOD

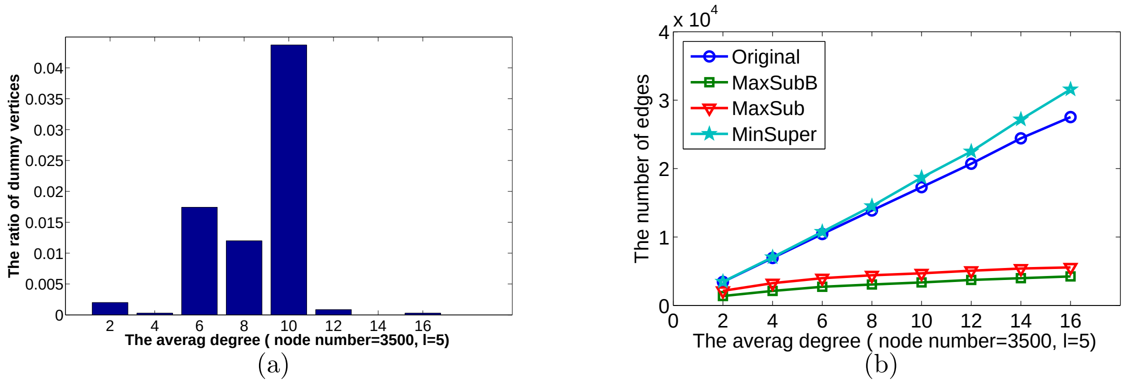
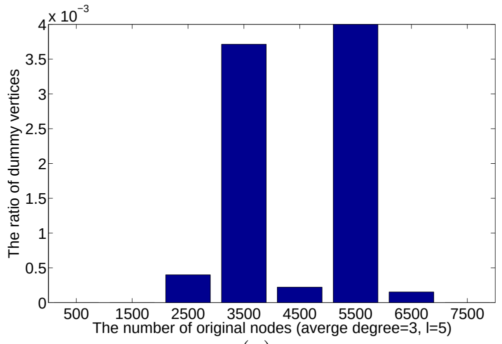
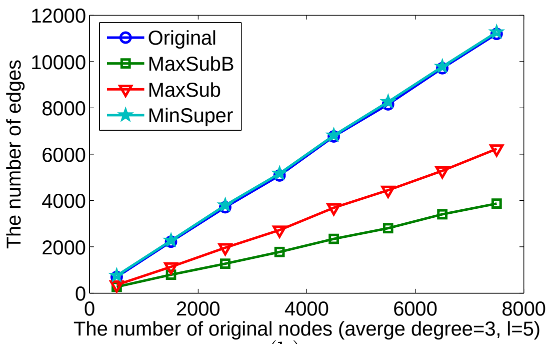


Figure 3.13: The influence of average node degree (a)The ratio of the dummy nodes in MinSuper (b) The number of published edges



(a)



(b)

Figure 3.14: The influence of the number of original nodes (a)The ratio of the dummy nodes in MinSuper (b) The number of published edges

crossing matchings in MaxSub will be more effective with larger scalability, and hence the edges published by MaxSub are more than those published by MaxSubB.

3.4 Summary

In this chapter, we defined a practical attack compromising sensitive relationship privacy from a published (and k-degree anonymized) social network. This attack motivated us to define an ℓ-diversity privacy guarantee for publishing social network data. To achieve this privacy guarantee, we developed three graph manipulation based heuristic algorithms, MaxSubB, MaxSub and MinSuper, each of which either only deletes edges or only inserts edges/vertices. We conducted a comprehensive set of experiments on both synthetic and real-world social network data sets to evaluate our technique performance in terms of both theoretical and empirical utility metrics.

We note that if the intended purpose of social network publishing is to disclose the empirical measures, then MinSuper is certainly a better approach than MaxSubB and MaxSub, because it offers more accurate approximations of empirical measures while achieving the same privacy guarantee. Nonetheless, we would like to note that this does *not* mean MinSuper should always be preferred over MaxSubB and MaxSub. This is because there are also many applications of social network publishing where the intended purpose is not to disclose the empirical measures, but to learn the correlation between edge existence and certain attribute of vertices in the OSNs, for example, learning rules such as the same affiliation$\rightarrow$edge connection with support of at least 10% and confidence of at least 75%. In this case, MaxSubB and MaxSub may be more appropriate solutions for data publishing as they guarantee that every published edge must exist in the original graph, thereby every learned rule holding in the original graph. In comparison, MinSuper may require the addition of fake edges, and hence cannot provide a similar guarantee to the accuracy of rules learned.

In fact, our definition of ℓ-diversity can also be applied to cases where the adversary holds other type of external knowledge except vertex degree, the design of which will be addressed as future work. Another direction of our future work is currently in MaxSubB and MaxSub we arbitrarily choose a maximum matching, we may find a better way to look for a maximal matching such that more original edges could be published. Of course, this will increase the complexity of our algorithms which may not be suitable for publishing data from large-scale social networks.

Algorithm 1: MaxSubB

```
Input   : $G(S, E_s)$ - a subgraph of the original graph
Output: $E_G$: published edges, where $E_G \subseteq E_s$
begin
1   Find a maximal matching in $S$ forming $S_{match1}$, $S_{match2}$ and $S_{unmatch}$
2   if $|S_{match1}| + |S_{match2}| \geq 2\ell$ then
3       if $|S_{unmatch}| < \ell$ then
4           Adjust the matching
5       if Adjustment succeeds then
6           $E_G$={edges maintained in the matching}
7           if $|S^{adjust}_{match1}| \geq \ell$ then
8               $MaxSubB(G_1, S^{adjust}_{match1})$ and $MaxSubB(G_2, S^{adjust}_{match2})$
9   else
10      if $|S_{match1}| + |S_{match2}| \geq \ell$ then
11          if $|S_{unmatch}| < \ell$ then
12              Adjust the matching
13          if Adjustment succeeds then
14              $E_G$={edges maintained in the matching}
```

Algorithm 2: MaxSub

```
Input  : The result graph published by MaxSubB
Output: a heuristic minimum $\ell$-diversity anonymous supergraph of $G$
begin
1   for any pair of leaf-node groups, $g_i$ and $g_j$, from MaxSubB do
2       Mark vertices matched between $g_i$ and $g_j$ by MaxSubB
3       Find a maximal matching among vertices without marks
4       for each SmallGroup $sg_k$ in $g_i$ and $g_j$ do
5           Split $sg_k$ into $sg_k^{match}$ and $sg_k^{unmatch}$
6           if $sg_k^{match} < \ell$ or $sg_k^{unmatch} < \ell$ then
7               Adjust the matching for $sg_k$
8           if Adjustment is successful or ($sg_k^{match} \geq \ell$ or $sg_k^{unmatch} \geq \ell$)
            then
9               Add back edges associated with $sg_k^{match}$ in the new matching
10          else
                Abort the maximal matching
```

Table 3.1: Number of leaf-level groups in MaxSubB

Project	ER	R-MAT	Co-Author	SIGMOD
Nodes	5000	5000	7955	3791
Groups	22	36	21	39
Average Group size	227	139	379	97

Algorithm 3: MinSuper

```
Input  : $G(V, E)$ the original graph, and its vertex set $T$, where $T = V$
Output: a heuristic minimum $\ell$-diversity anonymous supergraph of $G$
begin
1   Sort vertices in $T$ in the order of their degrees; $groupnum = 0$
2   while $T \neq \emptyset$ do
3       $t$=next non-grouped node in $T$; Remove $t$ from $T$; $counter = 0$;
4       while $counter < \ell$ and not reach the end of $T$ do
5           Find a non-conflicting node $v_{nc}$; Remove $v_{nc}$ from $T$
6           $counter++$
7       if $counter == \ell$ then
8           An $\ell$-group is formed
9       else
10          An $\ell$-group is formed with $(\ell - counter)$ fake nodes
11      $groupnum++$
12  Calculate degree difference $\Delta d_v$ for each node $v$
13  for $i$=1: $groupnum - 1$ do
14      for $j$=$i$ : $groupnum$ do
15          Create fake edges across $g_i$ and $g_j$
16          if $g_i$ still has a set of nodes with nonzero $\Delta d_v$, $S_i$ then
17              while $S_i \neq \emptyset$ do
18                  $s$ =next node in $S_i$
19                  Remove $s$ from $S_i$; Create $\Delta d_s$ fake nodes
20                  Construct $\Delta d_s$ fake edges linking $s$ with fake nodes
```

CHAPTER 4

LOCAL-VIEW BASED SUBGRAPH DETECTION IN OSNS

In this chapter, we mainly address the problem of detecting a small connected subgraph which covers a group of target users in OSNs from the perspective of a third-party analyzer. The subgraph connectivity problem has been researched in the domain of graph mining [22, 60, 35, 32, 58]. Nonetheless, the main difference between our proposed problem and the above line of research is two-fold: (1) The existing work addressed subgraph connectivity with the assumption of knowing the topology of the entire graph; however, we instead consider subgraph detection from the perspective of a third-party analyzer who only has a local view for a particular user in OSNs. (2) In addition to the objective considered in the traditional subgraph connectivity problem - minimizing the discovered subgraph that connects all of the target vertices, we also care for the cost in searching the subgraph on OSNs, which is specified by the number of queries (i.e., web accesses) issued in OSNs for the subgraph detection, as many OSN web sites limit the number of web accesses per IP address per day to avoid the intense workload at OSN servers.

In addition, there has been some work which researches on efficiently searching on social network graphs when the topologies of the entire graphs are unknown. Adamic et al. [3, 4] studied searching the shortest path between any two nodes on network graphs with power-law link distributions. They noticed that high connectivity nodes play an important role in communicating and networking, which is exploited in design of their searching technique. However, rather than searching for a shortest path between a pair of vertices, our work is focused on searching the connectivity of a target group of vertices in OSNs, which are usually far more than two.

The roadmap of this chapter is as follows. Section 4.1 introduces preliminaries including system model and problem definition. We propose two searching techniques in Section 4.2 which seeks a connected subgraph by a small number of queries to link all target vertices and introduce a heuristic algorithm for detecting a small subgraph which connects all of the target vertices given the entire graph in Section 4.3. We combine the techniques to be discussed in Sections 4.2 and 4.3 to solve our LMSD problem. In Section 4.4, we conduct a comprehensive set of experiments over large-scale real social network data sets to evaluate the effectiveness and efficiency of our proposed techniques. Finally, conclusions are drawn in Section 4.5 with some discussions on future research based on our finding.

4.1 System Model and Problem Definition

We model an OSN by an undirected graph, $G(V, E)$, in which the vertex set V represents users and the edge set E denotes the relationships between users, such as friendships. In this chapter, we only consider one type of relationships in an OSN for our model. The procedure of discovering the local view of a user on an OSN through web access can be modeled as querying a vertex on the OSN graph and responding with the list of its neighboring nodes. The number of a vertex's neighbors in the OSN corresponds to the vertex degree in the OSN graph.

Given the local-view discovery model, the procedure of our subgraph detection is based on the interaction with the OSN by a sequence of queries. We keep track of not only vertices already queried but also a list of vertex candidates to be queried in the near future. A vertex is said to be a *candidate* when it has not yet been queried while already seen in the local-view of at least one queried vertex. As more vertices

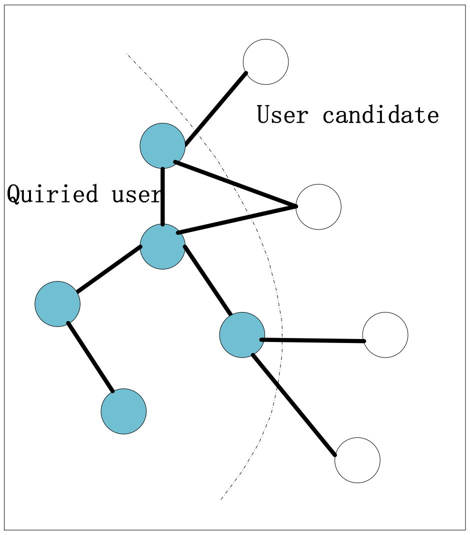


Figure 4.1: The procedure of querying on an OSN

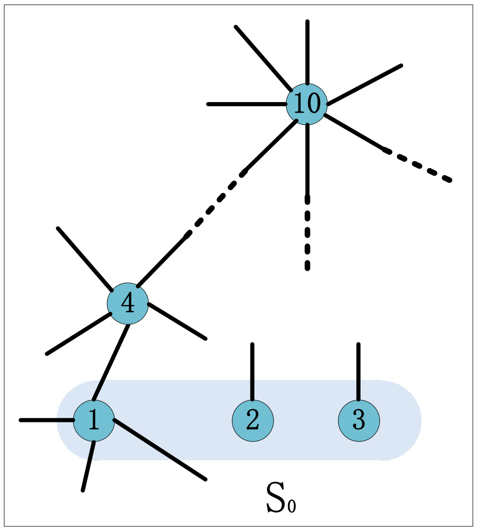


Figure 4.2: Example of the inefficiency of UMS

are queried, the third-party analyzer's view on the OSN graph becomes larger by stitching the local-views of queried vertices. Figure 4.1 illustrates a generic querying procedure for our subgraph detection in an OSN.

Given the system model, our local-view based minimum subgraph detection problem is defined in Problem 1. Note that the LMSD problem requires both the size of the detected subgraph and the number of queries be minimized, which is hard to achieve at the same time. To cope with this challenge, we heuristically interpret the problem in a way of sequentially handling the two concerns. Specifically, we wish to first discover a connected subgraph on the OSN which contains all target vertices by querying a small number of vertices, and then detect the minimum subgraph over the connected subgraph discovered in the first step. The underlying support to the effectiveness of our interpretation to solve the LMSD problem is if the number of vertices queried in the first step is small, the size of the finally detected subgraph should not be that large. One can see given the connected subgraph discovered in the first step, the minimum subgraph detection becomes Centralized Minimum Subgraph Detection problem (CMSD) - given the graph topology, looking for a minimum subgraph which connects all target vertices. The CMSD problem is a hard problem as proved in Theorem 9. The complexity of the CMSD indirectly indicates the hardness of the LMSD problem, because the former is part of the latter. Based on our aforementioned interpretation, now we can shift our attention from directly solving the LMSD problem to efficiently searching a connected subgraph which links all target vertices in the OSN graph by a small number of queries, which will be addressed particularly in the following section.

Problem 1. *Local-view based Minimum Subgraph Detection (LMSD): Given a set of target vertices S_0 in a graph $G(V, E)$ the full topology of which is unknown initially, find the minimum number of vertices from $V \setminus S_0$ to make all target vertices connected by querying the minimum number of vertices for local-view discovery.*

Theorem

Theorem 9. *The Centralized Minimum Subgraph Detection problem (CMSD) is NP-hard.*

Proof. We will prove the NP-hardness of CMSD by a reduction to the Steiner Tree problem (ST). The definition of ST is : Given an unweighed graph G and a set of vertices V_t in it, find a tree with minimum number of edges in G, which make any two vertices in V_t reachable to each other either directly or indirectly via other vertices in G. As is well known, the ST problem is NP-hard [24]. The decision version of ST is that given an unweighed graph $G(V, E)$, a set of vertices $V_t \subseteq V$ and an integer k, we are looking for a tree which involves all vertices in V_t and contains at most k edges from E. The decision version of CMSD problem is that given an unweighed graph

$G'(V', E')$, a set of vertices $V'_t \subseteq V'$ and an integer k', we are seeking for a subgraph of G' which includes all vertices in V'_t and covers at most k' vertices from $V' \setminus V'_t$.

Now we will demonstrate that there is a solution for ST if and only if there is a solution for CMSD. Evidently, the vertices in any steiner tree with at most k edges will be the solution of CMSD, where $k' = k + 1 - |V_t|$. On the other hand, any spanning tree of the subgraph found in CMSD will form a steiner tree with at most $k' + |V'_t| - 1$ edges. Here the spanning tree is referred to as a tree composed of all the vertices and some (or perhaps all) of the edges of a given graph. Therefore, the CMSD problem is NP-Hard. □

4.2 Social Network Feature Based Searching

The traditional graph searching techniques, such as *Depth First Search* (DFS) or *Breadth First Search* (BFS), can be applied to discovering a subgraph which covers all target vertices; however, their cost on querying the OSN is non-trivial without knowing the topology of the OSN graph. Therefore, we are motivated to design more efficient searching techniques to discover the subgraph connecting all target vertices.

Both of the two techniques we are going to propose in this section start the searching by querying all target vertices on the OSN graph. Each target vertex and its corresponding neighboring nodes returned from its query form a subgraph. At this point, these subgraphs are most likely disjoint with each other due to the sparse topology of social network graph. Each of the subgraphs has its own set of vertex candidates which have been “seen” on the OSN graph, while not queried yet. The initial vertex candidate set of a subgraph aforementioned solely contains the neighbors of the target vertex in that subgraph. However, a subgraph will grow as a vertex selected from its candidate set is queried. Now the problem of efficiently discovering a connected subgraph which connects all target vertices becomes how to select vertices to query so that the subgraphs of all target vertices can be merged into one piece quickly. Apparently, the selection of vertices to query is critical in our searching. In the following, we will first define a metric to evaluate the importance of a vertex in searching and then introduce our proposed searching techniques which are based on the well-known topological properties of social networks to ensure the efficiency of our searching on OSNs.

4.2.1 The Importance of Vertex Candidates

We see that the efficiency of merging the subgraphs associated with target vertices is determined by the selection of vertices to query in our searching procedure. In a dense graph ($|E| \gg |V|$), a straightforward criteria to select a vertex candidate is the number of target vertices which will become reachable after querying the vertex. A vertex should be queried if it can make more target vertices connected. However, solely leveraging such a criteria to select a vertex candidate on a sparse graph, such as social network graphs, may not be sufficient, and may even lead the searching process to terminate with failure. This is because in a sparse graph it is highly likely that none of the vertex candidates can directly improve the reachability among target vertices. Therefore, we need additional measurement to evaluate the importance of a vertex candidate based on our final goal of merging all of the subgraphs.

Inspired by the critical role of high-degree vertices in searching on social network graphs, which is discovered in [3, 4], we also prioritize vertex candidates of high degrees to query in our searching. However, the real degree of a vertex candidate is unknown until we query it, therefore, we need to predict the degree for each vertex candidate. We define *pre-degree* for each vertex candidate, which is referred to as *the number of edges having been seen associated with that vertex upon that status of searching.* For example, in Figure 4.1 the pre-degree of node 3 is two at that time point of searching, as we only see its connections with node 1 and node 2. As we query more vertices which directly connect with node 3, its pre-degree will increase. Given the measurement to evaluate the importance of vertex candidates, we will detail two searching techniques in the next subsection to quickly merge the subgraphs of all target vertices.

4.2.2 Algorithmic Techniques for Subgraph Merging

We propose two algorithmic searching techniques, called *Unbalanced Multiple-Subgraph (UMS)* and *Balanced Multiple-Subgraph (BMS)*, to connect the subgraphs of all target vertices. The main difference between these two techniques is how to decide the set of vertex candidates from which to choose a vertex to query next. Our delicate selection of the candidate set plays a critical role in efficiently merging subgraphs, as its design incorporates the topological properties of social networks. Before we dive into the details of our techniques, let us first introduce a terminology, *subgraph degree*, which is referred to as the maximum degree of vertices already queried in the subgraph. The subgraph degree critically determines from which subgraph we select a vertex candidate to query in the next searching step.

4.2.2.1 Unbalanced Multiple-Subgraph(UMS)

Basically, the UMS searching technique consists of three steps:

Step 1. Query all target vertices on the OSN graph and form their subgraphs individually.

Step 2. Select the subgraph with maximum subgraph degree as the *target subgraph.*

Step 3. Query the vertex with the highest pre-degree from the vertex candidate set of the target subgraph (break tie arbitrarily).

After querying a vertex, the target subgraph, its vertex candidate set and the pre-degrees of its vertex candidates are updated according to the return list of neighboring nodes of the queried vertex. If the query results in a subgraph overlapping with the target subgraph, they will be merged together as the target graph. Correspondingly, their sets of vertex candidates and their sets of vertices already queried will be merged. Additionally, the pre-degrees of vertex candidates will be updated, if necessary. The Step 3 will be repeated until the subgraphs of all target vertices are merged together. The reason we call this algorithm *Unbalanced* Multiple-Subgraph Searching is once the target subgraph is determined at the beginning of our search, it will not be changed as the search goes on, which seems to bias to the subgraph with maximum degree which is obtained in the Step 1.

Let's see an example illustrated in Figure 4.3. After querying nodes 1, 2 and 3, three subgraphs, Subg1, Subg2 and Subg3, are correspondingly formed with degrees of 2, 1, 3, separately. Based on the maximum degree rule in the UMS searching technique, Subg3 is chosen as the target subgraph. Node 4 is randomly selected as the first vertex to query from the candidate set of Subg3 as there are three vertex candidates with the same degree. Subg3 grows up. Then, node 5 becomes the candidate with the highest pre-degree in Subg3, therefore, we query node 5, which leads to the merging of Subg2 with Subg3. The search continues by querying vertices selected from the candidate set of Subg3 until Subg1 is also merged with Subg3.

4.2.2.2 Balanced Multiple-Subgraph (BMS)

The inspiration for designing BMS came from our concern over the efficiency of searching by UMS. One can see in essence UMS prioritizes vertices of high degrees in searching, which may not be able to efficiently reach out to the target vertices of low degrees. For example, in Figure 4.2 the subgraphs initialized with the target

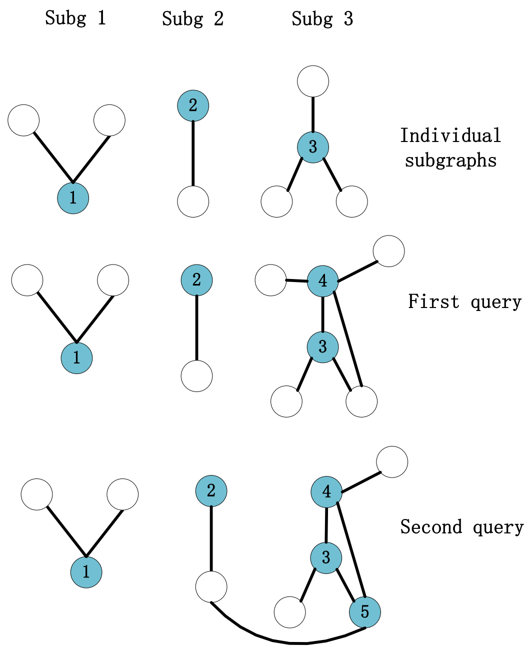


Figure 4.3: The example of using UMS technique

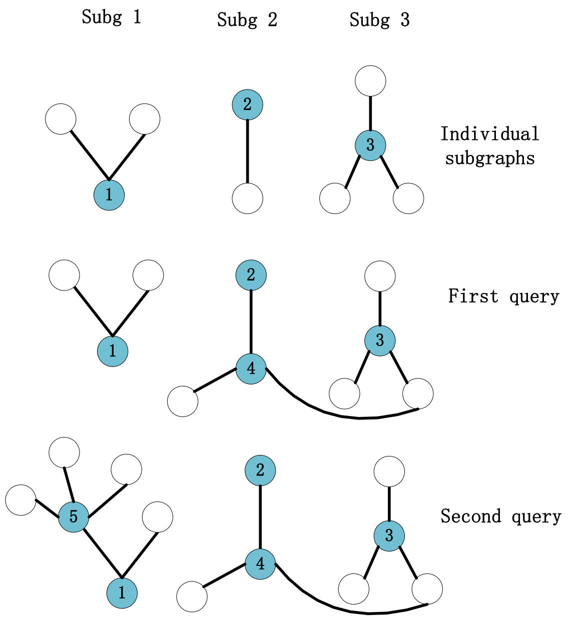


Figure 4.4: The example of using BMS technique

vertices of very low degrees can do nothing but wait for the reaching out of the subgraph with the maximum degree. However, since the vertices of high degrees in social networks are well connected as we introduced in Chapter 2, if we could reduce the degree difference among subgraphs by prioritizing the searching on the subgraphs of low degrees, the procedure of merging subgraphs should become faster. Based on this idea, we design the BMS searching technique which also consists of three steps:

Step 1. Query all target vertices on the OSN graph and form individual subgraphs.

Step 2. Select the subgraph with minimum subgraph degree as the *target subgraph*.

Step 3. Pick up a vertex candidate of the highest pre-degree from the candidate set of the target subgraph for querying (break tie arbitrarily).

Similar to UMS, in BMS if querying a vertex leads any subgraphs to overlapping, they will be merged by combining their vertex candidate sets, and the pre-degrees of their vertex candidates will be updated. Note after querying a vertex, the subgraph degree will be updated if the newly queried vertex has a degree higher than the current degree of that subgraph. One can see that the degree of the target subgraph may be increased as the search goes on, so that when it becomes greater than the minimum subgraph degree among all subgraphs, the target subgraph will be reassigned. In our searching procedure, whenever two subgraphs merge together, the degree of the newly formed subgraph will be assigned with the larger degree of the two subgraphs. Steps 2 and 3 will be repeated until the subgraphs of all target vertices are connected together. Unlike UMS, in BMS the target subgraph is dynamically changed to take care of low-degree subgraphs in our searching procedure, therefore, we call this technique Balanced Multiple-Subgraph Searching.

Let us run BMS on the toy example we used before for UMS, as shown in Figure 4.4. Initially, the subgraphs formed by querying target vertices have degrees of 2, 1, 3, separately. Since Subg2 has the minimum degree, it is defined as the target subgraph in the first query. Then the only vertex candidate in Subg2, node 4, is queried, which causes the merging of Subg2 and Subg3. The newly merged subgraph has degree of 3, which is larger than the degree of Subg1, therefore, the target subgraph is reassigned Subg1. In Subg1, node 5 is selected to query, which grows Subg1 and increases its degree to 4. At this point, the target subgraph will be reassigned again. The search continues until all subgraphs are merged. One can see the difference between our proposed UMS and BMS techniques is how to select the target subgraph.

Here we stress the unique topological properties of social networks used in design of BMS to ensure the efficiency of merging the subgraphs of target vertices in OSNs. First of all, based on the literature [4], we know that high-degree vertex based search can reach vertices of highest degree with a few steps of about $O(n^{\frac{\gamma-2}{\gamma-1}})$ in social network graphs, where γ is the exponent of the power law distribution, and n is the number of vertices in the networks. Therefore, in our search each subgraph will reach a vertex of highest degree with a few steps of searching. Second, because of the limited number of vertices of highest degree in scale-free social networks as well as their well-connectivity, searching multiple subgraphs along high-degree vertices will let the subgraphs grow towards the vertices of highest degree and merge together. Third, due to the well-connectivity among not only highest-degree vertices but also vertices of high degrees, balancing the search to take care of low-degree subgraphs will speed up the merging of all individual subgraphs associated with target vertices. Incorporating all of these topological properties in design of BMS ensures its efficiency of searching.

4.3 Centralized Minimum Subgraph Detection (CMSD)

Given the connected subgraph searched by our aforementioned techniques, we can solve the LMSD problem by looking for a even smaller subgraph which links all target vertices in the given subgraph. Considering the association between the CMSD problem and the Steiner Tree problem (ST) as we discussed in Section 4.1, we apply a classic approximate ST algorithm proposed in [37] to detect the minimum subgraph in the subgraph finally merged by our searching techniques on the OSN graph.

There are two main reasons for us to employ the ST algorithm in [37]: (1) It can guarantee the size of the detected subgraph is no larger than $2(1-1/\ell)$ times the size of the optimal subgraph, where ℓ is the number of leaves in the optimal tree. (2) It runs faster with the time complexity $|S_0||V|^2$, which is a critical concern when running algorithms on large-scale OSN data sets. Given an undirected and unweighed graph $G(V, E)$ and a set of target vertices $S_0 \subseteq V$, there are four steps to find a heuristic minimum steiner tree in [37]:

Step 1. Construct the complete undirected graph $G_1(V_1, E_1)$ by creating an edge between each pair of vertices in S_0 with a label of the length of their shortest path on G.

Step 2. Find the minimal spanning tree T_1 of G_1.

Step 3. Construct the subgraph G_s of G by replacing each edge in T_1 by its corresponding shortest path in G.

Step 4. Find the minimal spanning tree T_s of G_s. Delete from T_s edges with leaves which are non-steiner points.

4.4 Experimental Study

We evaluate the performance of our proposed techniques of solving the LMSD problem by a comprehensive set of experiments on large-scale real-world data sets of OSNs. We first introduce the data sets used in our experiments, and then define the measurements intended for evaluating our technique performance. We implemented the algorithms of UMS and BMS as well as the classic heuristic Steiner Tree algorithm (ST)[37] in Python (including NetworkX 1.5 package).

We conducted multiple groups of experiments for each data set by varying the number of the target vertices we selected, ranging from 10 to 60 in increments of 10. Furthermore, given a specific size of the target vertex set, we took five trials of experiments by selecting target vertices uniformly at random from the input graph associated with each data set.

4.4.1 Real Data Sets

We chose three data sets for our experimental study. The Facebook data set [63] is the data crawled from Facebook.com, capturing the friendship between users, which can be modeled as an undirected graph. The other two data sets, the Epinion data set [55] and the Slashdot data set [38], are collected in [2]. The Epinion data set shows the trust relationship between users on the customer review site Epinion.com. This data set is originally mapped to a directed graph, as the trust relationship is asymmetric. A user A trusts another user B does not ensure B also trusts A. However, since we consider undirected graphs in our problem formulation and algorithm design, we converted this data set to an undirected graph. Specifically, if there originally exists at least one edge between two vertices, regardless of their direction, we create an edge between the two vertices in the corresponding undirected graph. The Slashdot data set contains the friend/foe links between the users of Slashdot, where the links are also directional. The data set does not distinguish friendship from foeship between users. Similar to the Epinion data set, we also convert the Slashdot data set to an undirected graph for our experimental study. In Table 6.1, we list the number of edges/vertices in the undirected graph of each data set.

Furthermore, since we are concerned with how to connect a group of target vertices together by searching on an OSN, we need to ensure all of the vertices are indeed reachable to each other in the OSN graphs, that is to say, the modeled undirected graphs should be connected. Therefore, we pre-processed those original data sets by extracting the largest connected component from each data set. In Table 6.1, we also list the number of edges/vertices in the largest component of each data set. In our experiments we use the largest connected component of each data set as the input graph for evaluating our algorithms performance. Note that the data sets we

Table 4.1: Real data sets for our experiments

Date Sets	**Vertices**	**Edges**	**Largest Component (V, E)**
Facebook [63]	63731	817090	(63392, 816886)
Epinion [55]	75888	405740	(75877, 405739)
Slashdot [38]	82168	504230	(82168, 504230)

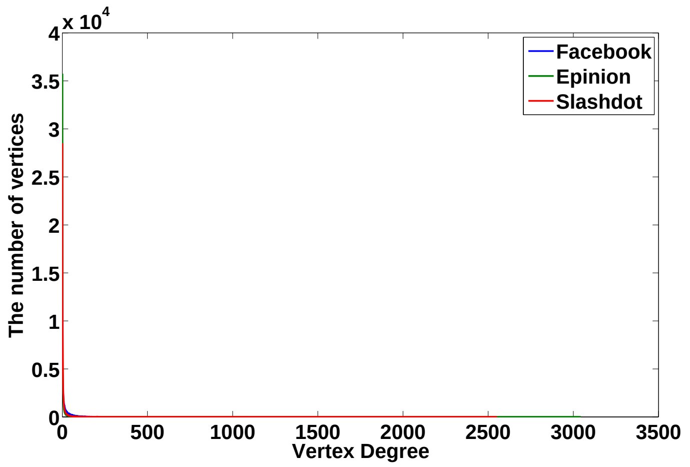


Figure 4.5: Degree distribution

will mention in the following mean the processed data sets unless otherwise stated.

4.4.2 Degree Distribution

We plotted the degree distribution for each data set. Figure 4.5 shows the vertex degree distributions of our OSN graphs almost follow the power-law distribution, which indicates the limited number of highest-degree vertices, thus ensuring the merging of individual subgraphs of the target vertices in BMS.

4.4.3 Connectivity of Vertices of High Degrees

To confirm the well-connectivity among vertices of high degrees in OSN graphs, we further analyze our three data sets. For each data set, we first extracted the

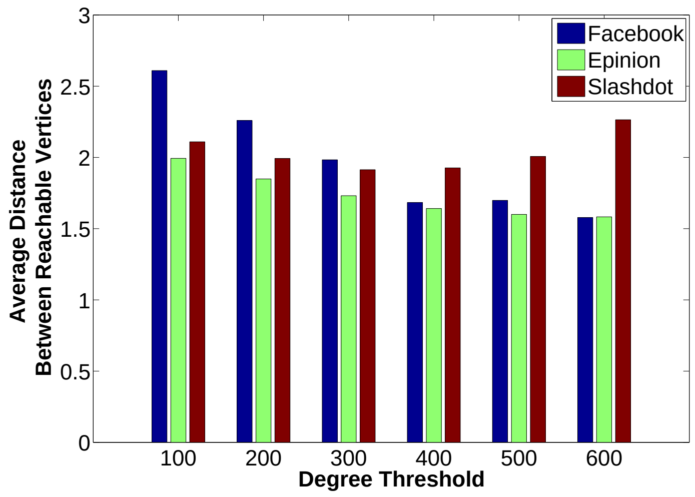


Figure 4.6: Average distance of reachable vertex pairs

subgraph formed by vertices with a degree more than a threshold and their associated edges. The threshold ranges from 100 to 600 in increments of 100. To analyze the connectivity among vertices of high degrees, we derived the number of the connected components and the average length of the shortest paths (i.e., distance) between any pair of reachable vertices in each extracted subgraph. From Table 4.2, we can see that although the number of vertices decreases as the degree threshold goes up, the vertices of high degrees still almost form a connected component. Furthermore, the average distance between any reachable pair of vertices is about 2, as shown in Figure 4.6. These results demonstrate the well-connectivity among vertices of high degrees in our OSN data sets, which ensures the good performance of our BMS technique.

4.4.4 Effectiveness

The effectiveness of our techniques to solve the LMSD problem is evaluated by the size of the finally detected subgraph. We not only applied the heuristic ST algorithm [37] to the subgraph finally merged by the searching techniques of UMS

Table 4.2: Subgraphs of high-degree vertices

$\geq$ **Degree**		100	200	300	400	500	600
Facebook	**Vertices**	3307	461	106	46	26	11
	Components	1	1	1	1	1	2
Epinion	**Vertices**	1684	630	290	147	86	52
	Components	2	1	1	1	1	1
Slashdot	**Vertices**	1916	757	235	115	61	39
	Components	1	2	2	1	3	3

and BMS, but also ran it on each data set directly in order to compare our local-view based subgraph detection algorithms with the subgraph detection given the global graph. For the sake of simplifying our reference to the algorithm applied on the global graph, we name it the *Global Steiner Tree* algorithm (GST). Moreover, when we mention UMS (BMS) in the following discussion, we mean the entire technique which first uses UMS (BMS) to search a connected subgraph containing all target vertices by a few queries and then applies ST on the connected subgraph to find the minimum-size subgraph connecting the target vertices.

From Figures 4.8, 4.10 and 4.12, we can see that the sizes of subgraphs finally detected by UMS and BMS are fairly acceptable, as compared to the number of target vertices. Furthermore, the sizes of those detected subgraphs are similar to the size of those detected by the GST algorithm, which demonstrates the effectiveness of our techniques to solve the LMSD problem. One may also notice that in some cases the subgraphs detected by UMS and BMS are even smaller than those discovered by the GST algorithm. This happens because the heuristic ST algorithm [37] has the approximation ratio of $2(1 - 1/\ell)$, thus it is possible for our techniques to discover a subgraph smaller than the subgraph detected by GST.

In addition to the size of detected subgraphs, we also plotted the number of queried vertices by BMS in Figures 4.8, 4.10 and 4.12, which is particularly named

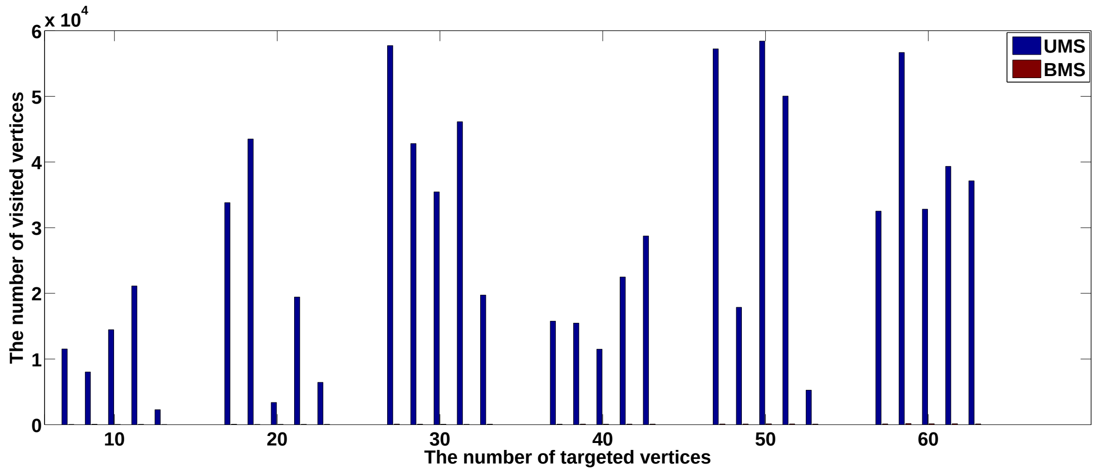


Figure 4.7: The cost for Facebook set

BMS-N. By comparing BMS and BMS-N, we notice that the number of queried vertices has the same order of magnitude as the size of the subgraph detected by BMS, which validates our interpretation to the LMSD problem mentioned in Section 4.1 - a small number of queries will not lead to detecting a large subgraph which connects all target vertices. It also demonstrates the effectiveness of the BMS technique on merging individual subgraphs of the target vertices in our searching.

4.4.5 Efficiency

We exploited the number of queried vertices to evaluate the efficiency of our proposed algorithms. Evidently, issuing a query in an OSN requires the interaction with the OSN server, thus increasing the workload at the server and also costing time. Furthermore, nowadays many OSNs have limited the number of web accesses from one (or a group of) IP address(es). Therefore, the fewer queries needed for subgraph discovery the better. We straightforwardly use the number of queried vertices to

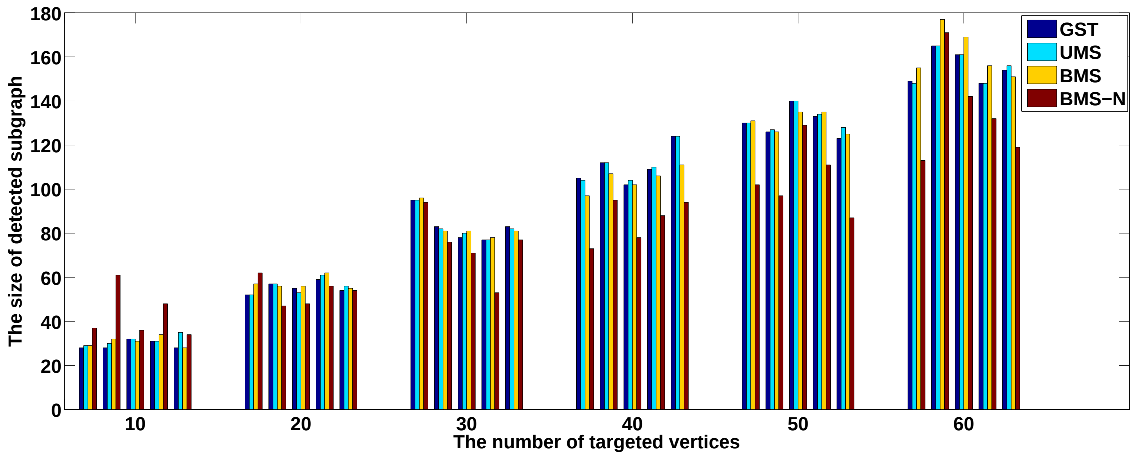


Figure 4.8: The effectiveness for Facebook set

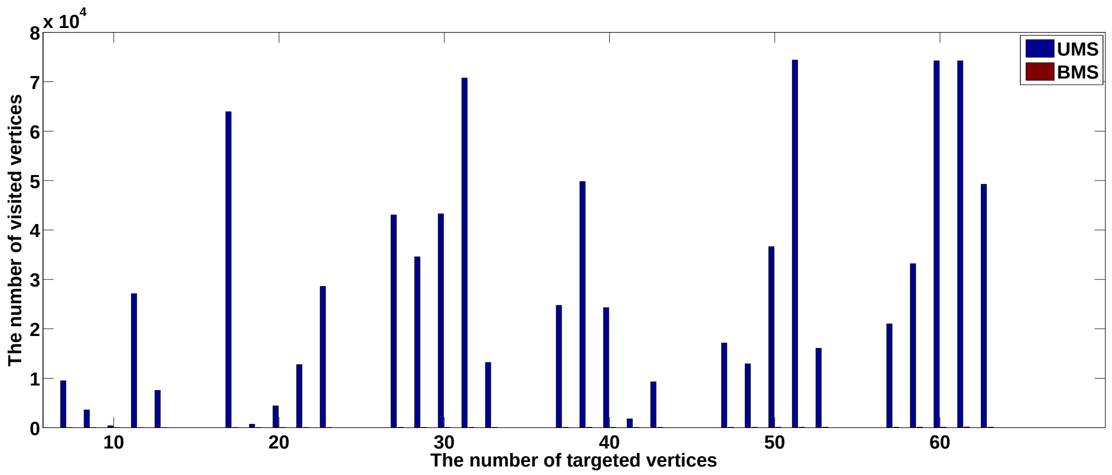


Figure 4.9: The cost for Epinion set

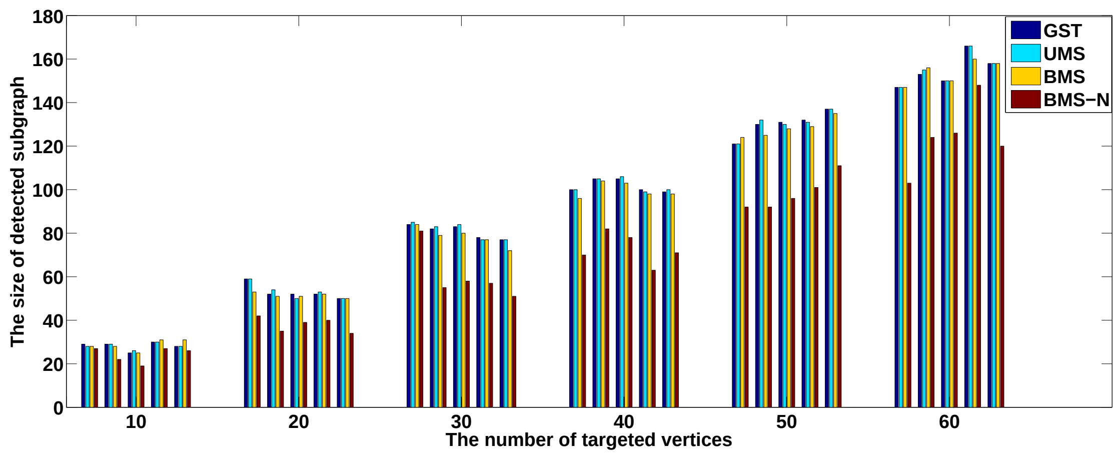


Figure 4.10: The effectiveness for Epinion set

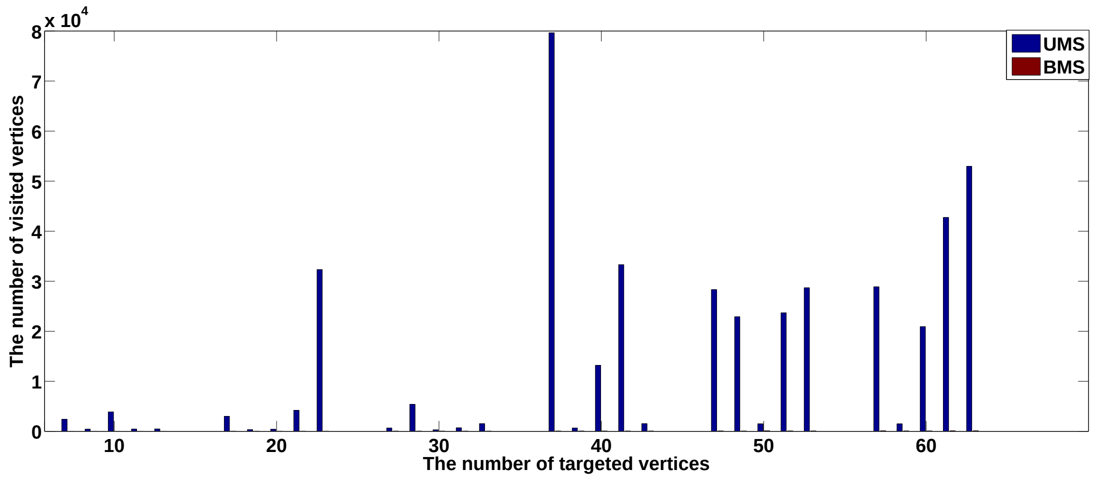


Figure 4.11: The cost for Slashdot set

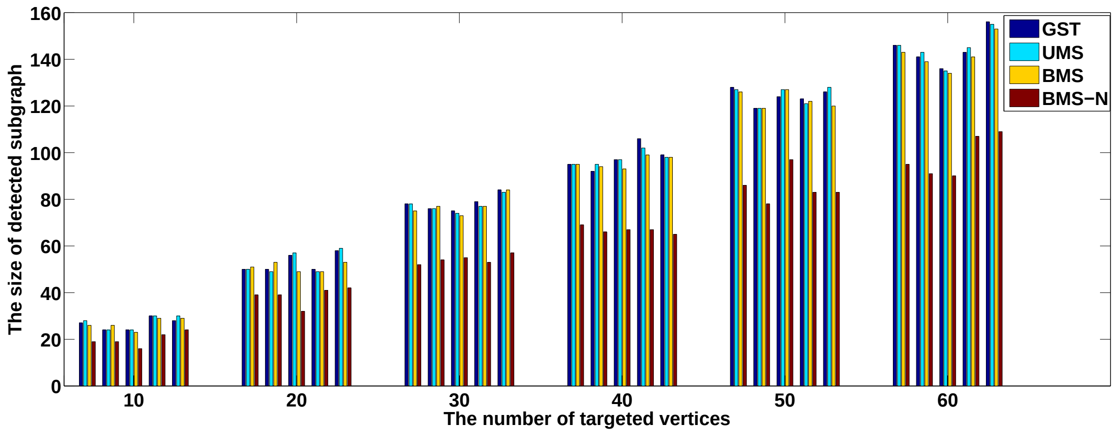


Figure 4.12: The effectiveness for Slashdot set

quantify the efficiency of our searching techniques. Figures 4.7, 4.9 and 4.11 compare the number of the queried vertices in UMS and BMS. However, since the order of magnitude of the number of queried vertices in UMS is far more than the order of magnitude of the number of queried vertices in BMS, the bar plot for BMS is not seen well in the figures. But one can look at the bar plots of BMS-N in Figures 4.8, 4.10 and 4.12, which are exactly the number of vertices queried by BMS. We can see the BMS technique outperforms the UMS technique. We owe the remarkable efficiency provided by BMS to our design principle, which delicately balances the degrees of subgraphs in searching and also leverages the well-connectivity of vertices of high degrees. Furthermore, the good performance of BMS on efficiency in turn demonstrates the well-connectivity among vertices is not limited to high-degree vertices, rather, all vertices on the OSNs are connected well, as we have found a way to discover the connectivity of a group of arbitrarily given vertices within a few number of queries in OSNs.

4.5 Summary

In this chapter, we proposed a novel problem of discovering a minimum connected subgraph to a given group of vertices from the perspective of a third-party analyzer on OSNs, namely local-view based minimum subgraph detection (LMSD). To solve the LMSD problem, we proposed two searching techniques, called Unbalanced Multiple-Subgraph (UMS) and Balanced Multiple-Subgraph (BMS), which are particularly based on the well-known topological properties of social networks, including small-world phenomenon, power-law vertex degree distribution and the well-connectivity of vertices of high degrees.

Through a comprehensive set of experiments over the large-scale real-world data sets from OSNs, we evaluated the performance of our proposed techniques. The BMS technique displays a remarkably good performance, as compared to UMS, which demonstrates all users on OSNs are well connected on OSNs as we can detect their connectivity by a small number of vertex queries. We believe that the discovery of the well-connectivity feature in OSNs will help improve some existing random-walking based searching techniques [20, 25, 33, 66, 67, 68, 69]. An OSN is so different from a random graph that searching techniques should be designed based on its unique topological features to enhance the efficiency of searching.

Furthermore, the design principle in BMS of searching from low-degree subgraphs shows great impact on the efficiency in solving the LMSD problem, particularly making the subgraphs of all target vertices merge together quickly in searching. In fact, given the minimum subgraph detected by our techniques, a lot of applications can be developed on it, which explores the scope of applications the third parties can develop in OSNs.

CHAPTER 5

REPUTATION-BASED DATA FORWARDING IN MSNs

In this chapter, we propose a trust-based framework which can be flexibly integrated with a large family of existing single-copy data forwarding protocols in MSNs, aiming at providing comprehensive evaluation to an encountering node's ability of delivering data and preventing a malicious node from constructing fake relations in MSNs. With the help of our proposed framework, not only black hole attack but also the attack of arbitrarily forwarding data can be counteracted effectively.

The remainder of this chapter is organized as follows. Section 5.1 introduces the network scenario and the assumptions behind our framework. After giving an overview of our framework in Section 5.2, we first design a control message in Section 5.3, called the Positive Feedback Message (PFM), and then present our proposed trust-based framework in detail in Section 5.4. Section 5.5 describes the simulation setup and the analysis of experimental results. Section 5.6 concludes the chapter.

5.1 Network Assumption

1) *Scenario*: The scenario we are concerned with is a network with several wireless devices (i.e., nodes) moving in a community, which are either held by people or fixed on buses. In such a network, data are forwarded in a store-and-forward manner, in which nodes will not forward data until encountering another node with a stronger competency of delivering data. Admittedly, multi-copy data transmission is more robust in wireless network than single-copy transmission, because although one copy of a data message is lost, other copies may still get delivered to the destination.

However, in order to demonstrate the disastrous impact of malicious nodes on the network performance (i.e., delivery rate) in MSNs, we solely consider the single-copy data transmission, in which each node will remove the data message after forwarding it to another node.

2) *Node capability*: Assume nodes have the same capability of computing, communicating and storing. Their communication ability is limited by specific wireless techniques. For instance, through Bluetooth, only when two nodes move into each other's communication range (e.g., 10$\sim$100 m) could they detect each other and start communication. Additionally, in order to secure data transmission over the wireless network, each node is assigned a unique ID and a pair of public/private keys for encrypting and decrypting data, as well as with a public key certificate issued by some trustable Public Key Infrastructure (PKI).

3) *Attack model*: Similar to benign nodes in the network, malicious nodes are assumed to be wireless devices as well, while they may have stronger capability of computing and storing as compared to those benign nodes. Furthermore, malicious nodes independently mount passive attacks - deliberately dropping or arbitrarily forwarding data which they intercepted by bloating their ability of meeting the destination nodes. Note that active attacks, such as tampering with data, can also be launched by malicious nodes; however, the traditional cryptographic techniques developed for wireless networks perform well against those attacks, and hence we will not stress them here.

5.2 Overview of Trust-Based Framework with PFM

Similar to the traditional Trust Management Systems (TMS), we include a Watchdog component into our framework. The functionality of the Watchdog embedded at each nodes is to monitor the forwarding behavior other nodes to whom it forwards data. In our framework, node A will evaluate node B's forwarding behavior in terms of the evidence, Positive Feedback Message (PFM), created by another node, say node C, to whom node B further forwards the data that is forwarded by node A. Specifically, after receiving the data, node C generates a PFM destined to node A with the intention of convincing node A that node B has helped forward data successfully. Each node (e.g., node A) who sends out data to its next-hop forwarders keeps recording for each of those forwarders how many PFMs corresponding to data sent out have and have not come back using two counters, respectively. In the simplest case, a forwarder is suspicious, if the corresponding PFM is never received by the node who forwarded the data message to the forwarder, because there is no positive evidence to demonstrate the good forwarding behavior of the forwarder. However, in addition to malicious behaviors, the non-arrival of a PFM possibly results from such uncertainty in MSNs as network partitions, which will be particularly addressed in our trust/reputation formulation in Section 5.4.

PFMs, which are collected at the Watchdog and fed into the main body of our trust-based framework, help form the reputation and trust opinion to another node's forwarding behavior in the future. Specifically, at each node, the direct reputation of the other node's forwarding behavior is formed based on the two counters used to record the forwarding behaviors of the latter node. In addition to the direct reputation evaluation, whenever two nodes meet together, they will exchange their counting records, which will be used to calculate the indirect reputation. Based on the direct and indirect information, a trust value will be derived and integrated with the probability of meeting the destination node to comprehensively evaluate the forwarding competency of a node.

In the following, we will first introduce the PFM structure in Section 5.3 and then discuss the main body of our trust-based framework in Section 5.4.

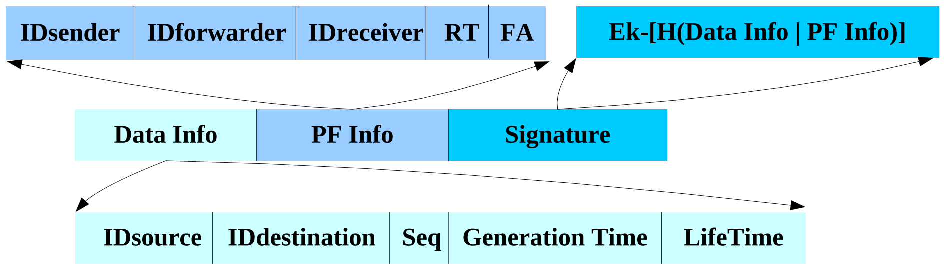


Figure 5.1: Structure of Positive Feedback Message (PFM)

5.3 Positive Feedback Message (PFM)

As the key part in our Watchdog mechanism, the *Positive Feedback Message* (PFM) is designed to demonstrate that a node already truthfully forwarded data to another node rather than dropping them or arbitrarily forwarding them to a node with weaker competency of delivering data. In the following, we will first introduce the structure of PFM and then explain its functionality in detail by an example illustrated in Fig. 5.2.

5.3.1 Structure of PFM

As illustrated in Fig. 5.1, a PFM consists of three fields: *Data Information*, *PF Information* and *Signature of PFM creator*. The combination of *Data Information* and *PF Information* uniquely represents one-hop forwarding behavior of a data message.

1) Data Information: A data message is specified by the identities of the source/destination and the message sequence number. In addition, *Lifetime* and *CreationTime* of a data message determine the lifetime of its PFMs. When a data message expires, its PFMs will be removed from the network to save storage resource.

2) PF Information: *PF Information* represents a one-hop behavior of a forwarder identified with *IDforwarder*. The field of *ForwardingAbility* (*FA*) quantifies

the goodness of the forwarder's behavior. Specifically, a high value of FA demonstrates that the data message has been forwarded to a node (i.e., the PFM's creator) with more competency of delivering data to the destination. *FA* is designed to prevent malicious nodes from arbitrarily forwarding data while still earning good reputation. Without *FA*, the destination node of a PFM can not distinguish a desirable forwarding behavior from a bad one after receiving the PFM.

3) Signature: The *Signature* of the PFM creator prevents a malicious node, who attempts to earn reputation of forwarding behavior, from generating fake PFMs or tampering the original information in PFMs.

5.3.2 An Example of the Functionality of PFM

Essentially, the functionality of PFM is to convince the node who sent a data message to a forwarder that the forwarder has further forwarded the data message to a more competent node (i.e., the PFM creator).

To create a PFM, the PFM creator requires some information about the previous hop forwarding of the corresponding data message. Thus, some information needs to be attached to the raw data message updated after each forwarding, such as *Sender*, *Forwarder*, *Receiver*, *Receiving Time* (RT) and *Sending Time* (ST). Apparently, signature and encryption techniques are required to secure data transmission and prevent malicious node from tampering the data message. However, the signature details will not be discussed here, since we mainly emphasize the function of PFM. An example is given in Fig. 5.2 to illustrate how to generate, forward and verify a PFM.

Step 1: PFM generation: Upon encountering node A and comparing with node A's delivery competency, node S sends the data message to node A who is more competent for delivering data, meanwhile, node S sets a timer to wait for the arrival of the corresponding PFM. Node A carries the data message until meeting another more competent node, say node B, and then sends it to node B after updating the fields attached to the raw data message. Once receiving the data message, node B creates a PFM destined to node S.

Step 2: PFM transmission: The PFM is transmitted over the network by the global/local epidemic forwarding [62]. Admittedly, PFMs can be regarded as regular data and forwarded by some existing data forwarding protocols, which will bring in less overhead than epidemic forwarding [62]. However, since the size of a PFM is quite small, the epidemic routing will not introduce a large amount of overhead into the network. Furthermore, many PFMs can be compressed together for one transmission in practice. Therefore, we simply employ epidemic forwarding for PFM transmission in our framework.

Step 3: PFM verification: After receiving a PFM, other nodes except node S will continue to forward it, while node S stops forwarding it further and verifies the goodness of the forwarding behavior of node A in terms of the FA field. If it is proved to be a good forwarding, one of the two counters mentioned in the Section 5.2 will be increased by one. If a bad forwarding is proved, or the timer expires before the arrival of the PFM, the other counter will be increased by one. These two counters have a critical influence on updating a forwarder's reputation.

Compared to the end-to-end ACK mechanism in MANETs, the advantage of applying PFM is that the feedback of a forwarding behavior comes faster, which is of primary importance for the reputation updating in MSNs characterized by long delays and frequent partitions. More specifically, rather than waiting for the arrival of ACKs generated from the data destination node, intermediate nodes (data senders) are able to receive PFMs earlier to update the reputations of the next-hop forwarders in a relatively real-time manner. In addition, for the end-to-end ACK mechanism, if a data message is lost on the way to the destination, the reputation of all intermediate forwarders will be impacted negatively, even though they indeed help with forwarding. However, in the PFM-based mechanism, the loss of a data message only influences the reputation of the last forwarder who forwarded the data.

5.4 Our Trust-Based Framework

After introducing the special structure and functions of PFM, we will discuss the main body of our trust-based framework for data forwarding in MSNs. Technically, there exist three modules in our proposed framework, *Reputation Module, Trust Evaluation Module and Forwarding Decision Module.* The Fig. 5.3 not only comprehensively shows the integration of the three modules but also presents some related events happening in those modules. In the following, we will detail the functions of these modules and corresponding events that occur in the modules.

5.4.1 Reputation Module

The reputation module focuses on how to exploit the collected information to derive the reputation of a node for data forwarding. Such information basically includes the direct observation (i.e., first-hand information) and the indirect observation (i.e., second-hand information) on an encounter's historical forwarding behaviors.

• Direct Observation: The direct observation is provided by the Watchdog component embedded in each node. Specifically, in MSNs, the Watchdog scrutinizes a forwarder's forwarding behavior by collecting PFMs which are exploited as the evidence to demonstrate the effective forwarding behaviors of the forwarder.

Based on the traditional concept in TMS [23], we also exploit the Beta distribution, Beta(α,β), to estimate the reputation to the forwarding behavior of an encounter, where α and β count behaviors of good forwarding and bad one (i.e., not

Table 5.1: Terminology used in our proposed framework

Notations	**Definitions**
Δt	The interval of aging reputation
α	Number of good forwardings
β	Number of droppings or inefficient forwardings
α_{ij}^{dir}	Number of node j's good forwardings by directly observing at node i
β_{ij}^{dir}	Number of node j's droppings or inefficient forwardings by directly observing at node i
$b_{ij}^{dir/ind/com}$	Node i's direct/indirect/comprehensive belief to node j on good forwarding
$d_{ij}^{dir/ind/com}$	Node i's direct/indirect/comprehensive disbelief to node j on good forwarding
$u_{ij}^{dir/ind/com}$	Node i's direct/indirect/comprehensive uncertainty to node j on good forwarding
b_{ij-k}^{ind}	Node i's indirect belief to node j on good forwarding provided by node k
d_{ij-k}^{ind}	Node i's indirect disbelief to node j on good forwarding provided by node k
u_{ij-k}^{ind}	Node i's indirect uncertainty to node j on good forwarding provided by node k
S_{ij}	The set of nodes which have met node i and offered their direct observations about node j
T_{ij}	The value of trust from node i to node j for good forwarding behavior
γ	A node character factor [39]
σ	The relative atomicity [31]

forwarding or ineffective forwarding), respectively. Specifically, after the sender of a data message gets the PFM that it is waiting for and further confirms the effectiveness of that forwarding, α will be increased by one. Otherwise, if the timer times out before the PFM arrives, or the received PFM indicates the data message has been forwarded to an incompetent node, β will be increased by one.

Note that even based on direct observation, one node's evaluation to the other node's forwarding behavior should diminish slowly, if it has been a long time since the former directly observed the forwarding behavior of the latter. The intuition is that a forwarder may change its forwarding behavior between good and bad over time, so that the reputation should be aged and evaluated in a real-time manner. Specifically, at a fixed interval, Δt, node i updates α_{ij}^{dir} and β_{ij}^{dir} by adding the number of the new forwarding behaviors of node j during Δt to the old values aged by a factor w_{dir}, as shown in Equations (5.1) and (5.2), where p_{ij} and q_{ij} represent the numbers of good forwarding behaviors and bad ones recorded by the two counters during Δt, respectively.

$$\alpha_{ij(new)}^{dir} = w_{dir} \times \alpha_{ij(old)}^{dir} + p_{ij} \tag{5.1}$$

$$\beta_{ij(new)}^{dir} = w_{dir} \times \beta_{ij(old)}^{dir} + q_{ij} \tag{5.2}$$

In practice, α_{ij}^{dir} and β_{ij}^{dir} may not exactly reflect the real forwarding behaviors. For example, it is possible that node j indeed assists with forwarding, but the corresponding PFM is delivered to its destination (i.e., the previous data sender) before its expiration, which most likely happens in MSNs due to long delay and frequent network partition. To deal with such uncertainty in evaluating the reputation for data forwarding, we adopt an approach proposed in [39] which uses the Dempster-Shafer Belief Theory [57] to quantify the uncertainty of some random variables. We refer the readers to [39] for details. Specifically in our context, the two parameters (α_{ij}^{dir} and β_{ij}^{dir}) are mapped to a tuple ($b_{ij}^{dir}, d_{ij}^{dir}$ and u_{ij}^{dir}), where b_{ij}^{dir} defines node i's belief in node j's good forwarding behavior in terms of the direct observation, d_{ij}^{dir} is the disbelief of a good forwarding, and u_{ij}^{dir} represents the uncertainty of a good forwarding, subject to the constraint $b_{ij}^{dir} + d_{ij}^{dir} + u_{ij}^{dir} = 1$. The mapping is specified in Equations (5.3), (5.4) and (5.5), where the meanings of variables are listed in Table 5.1.

$$b_{ij}^{dir} = \frac{\alpha_{ij}^{dir}}{\alpha_{ij}^{dir} + \beta_{ij}^{dir}} \times (1 - u_{ij}^{dir}) \tag{5.3}$$

$$d_{ij}^{dir} = \frac{\beta_{ij}^{dir}}{\alpha_{ij}^{dir} + \beta_{ij}^{dir}} \times (1 - u_{ij}^{dir}) \tag{5.4}$$

$$u_{ij}^{dir} = \frac{12 \times \alpha_{ij}^{dir} \times \beta_{ij}^{dir}}{(\alpha_{ij}^{dir} + \beta_{ij}^{dir})^2 \times (1 + \alpha_{ij}^{dir} + \beta_{ij}^{dir})} \tag{5.5}$$

With the help of the Dempster-Shafer Belief Theory, although we could not completely avoid the inaccuracy of estimating forwarding reputation, at least the uncertainty is reduced reasonably.

• Indirect Observation: For the indirect observation, a node will collect it not only from the currently connecting nodes but also from all nodes it has met before. Since MSNs are characterized by frequent partition and long delay, fully taking advantage of the indirect observation will improve the accuracy of the prediction to a node's future forwarding behavior. Technically, when two nodes meet, they exchange their own direct observation on the forwarding behaviors of other nodes they have encountered previously as indirect observation information, as formalized in the Equations (5.9), (5.10) and (5.11).

$$b_{ij-k}^{ind} = b_{kj}^{dir} \tag{5.6}$$

$$d_{ij-k}^{ind} = d_{kj}^{dir} \tag{5.7}$$

$$u_{ij-k}^{ind} = u_{kj}^{dir} \tag{5.8}$$

5.4.2 Trust Evaluation Module

When node i meets node j opportunistically, they will comprehensively evaluate each other's reputation of forwarding data in terms of the collected direct and indirect information. They first combine the indirect observation from all nodes encountered before, and then integrate it with the direct observation to derive the trust value.

1) *The combination of indirect observation.* Referring to the approaches proposed in [39], node i calculates the indirect reputation of node j by combining the indirect information collected from each previous encountering node, say node k, based on Equations (5.9), (5.10) and (5.11).

$$b_{ij}^{ind} = \sum_{k \in S_{ij}} \frac{b_{ik}^{dir} \times b_{ij-k}^{ind}}{|S_{ij}|} \tag{5.9}$$

$$d_{ij}^{ind} = \sum_{k \in S_{ij}} \frac{b_{ik}^{dir} \times d_{ij-k}^{ind}}{|S_{ij}|} \tag{5.10}$$

$$u_{ij}^{ind} = \sum_{k \in S_{ij}} \frac{b_{ik}^{dir} \times u_{ij-k}^{ind} + d_{ik}^{dir} + u_{ik}^{dir}}{|S_{ij}|} \tag{5.11}$$

2) *The integration of direct and indirect observation.* To integrate the direct observation with the indirect information properly, the comprehensive belief, disbelief and uncertainty can be derived based on Equations (5.12), (5.13) and (5.14), as proposed in [39].

$$b_{ij}^{com} = \phi_1 \times b_{ij}^{dir} + \phi_2 \times b_{ij}^{ind} \tag{5.12}$$

$$d_{ij}^{com} = \phi_1 \times d_{ij}^{dir} + \phi_2 \times d_{ij}^{ind} \tag{5.13}$$

$$u_{ij}^{com} = 1 - b_{ij}^{com} - d_{ij}^{com} \tag{5.14}$$

$$\phi_1 = \frac{\gamma \times u_{ij}^{ind}}{(1-\gamma) \times u_{ij}^{dir} + \gamma \times u_{ij}^{ind} - 0.5 \times u_{ij}^{dir} \times u_{ij}^{ind}} \tag{5.15}$$

$$\phi_2 = \frac{(1-\gamma) \times u_{ij}^{dir}}{(1-\gamma) \times u_{ij}^{dir} + \gamma \times u_{ij}^{ind} - 0.5 \times u_{ij}^{dir} \times u_{ij}^{ind}} \tag{5.16}$$

3) *The quantification of the trust value.* A trust to node j's future forwarding behavior at node i is quantified by Equation (5.17), following the literature [31], which takes uncertainty into consideration.

$$T_{ij} = b_{ij}^{com} + \sigma \times u_{ij}^{com} \tag{5.17}$$

where σ is defined as the relative atomicity, which is based on the principle of insufficient reasoning: the uncertainty of n atomic states is split equally among these n states. Without any biased view on the two states, good forwarding or bad forwarding, we set σ=0.5 by default. This parameter varies with different scenarios, so it can be tuned more accurately, given some specific knowledge regarding the network.

5.4.3 Forwarding Decision Module

The forwarding decision module takes two values as an input, the trust to the forwarding behavior of a node encountered, M_t, and the ability of delivering data (i.e., the possibility of meeting the destination), M_f, claimed by the encountering node itself. The former is the output of the trust evaluation module, and the latter comes from the existing data forwarding protocols in MSNs, which is popularly estimated based on the encountering history or the similarity between the encountering node and the destination node. In general, our proposed trust-based framework is applicable to most of the existing data forwarding protocols in MSNs by integrating the trust value with any pure metrics used to measure a node's ability of delivering data. However, we only exemplify the application of our framework to one of the classic protocols in DTNs, PROPHET [45]. In the following, we will briefly introduce PROPHET first and then discuss the integration of the trust with the ability of delivering data, where we will propose the "**Self-Trusting**" principle and discuss its effect on data forwarding decision in MSNs.

5.4.3.1 Overview of PROPHET

PROPHET [45] is a data forwarding protocol designed for DTNs, which uses the encountering history and transitivity to predict the possibility of two node encountering in the future. PROPHET is based on three important equations, as formulated in Equations (5.18) (5.19) and (5.20), to update the delivery probability values, which are introduced in detail as follows.

Node A will update its probability of delivering data to node B by Equation (5.18), whenever it meets node B.

$$P(A,B) = P(A,B)_{old} + (1 - P(A,B)_{old}) \times P_{init} \tag{5.18}$$

where P_{init} is an initialization constant, and set to 0.75 in our experiments by following the experiment configuration in [45].

Similar to the reputation aging applied in our proposed framework, if a pair of nodes, A and B, do not meet each other for a long time, node A (node B) would age its probability of delivering data to node B (node A) by Equation (5.19) with a constant time interval.

$$P(A,B) = P(A,B)_{old} \times \xi^k \tag{5.19}$$

where ξ is the aging constant.

In addition to the probability updated by directly encountering, the delivery probability also has a transitive property. Specifically, when node A encounters node B after node B has met node C, node A will update its delivery probability to node C based on P(A,C) and P(B,C) using Equation (5.20).

$$P(A,C) = P(A,C)_{old} + (1 - P(A,C)_{old}) \times P(A,B) \times P(B,C) \times \zeta \tag{5.20}$$

where ζ is a scaling constant which controls how large the impact the transitivity value has on the delivery predictability.

Apparently, in PROPHET, a malicious node who arbitrarily claims delivery probability will be able to intercept data from other nodes. If the malicious node either drops or arbitrarily forwards the data, it will detrimentally degrade the network performance.

5.4.3.2 Comprehensive Measurement For Forwarding Data

Essentially, the trust to the encounter's forwarding behavior, M_t, and the pure forwarding metric, M_f, are completely different metrics. Because while the trust evaluation is related to the message transmission over the network, the latter metric usually represents the chance of meeting the destination, which is determined by the node mobility pattern. Both of the two metrics are indispensable, so we need to integrate them properly to evaluate a node's real forwarding competence.

Here are two extreme cases in design of the integration of these two metrics. One is to completely depend on the pure forwarding metric, M_f, however, black hole attacks can be launched successfully in this case; the other is to solely rely on the trust value, M_t, but the original intention of data forwarding design in MSNs will be damaged as data forwarding will not consider any possibility of data delivery. For the purpose of avoiding these extreme cases, we leverage Equation (5.21) to evaluate the delivery competency which balances the roles of M_f and M_t.

$$Comprehensive\ Competency = M_t \times M_f \tag{5.21}$$

We notice one property of the trust to data forwarding in MSNs, namely "**Self-Trusting**"principle, depicted in Principle 5.4.3.2. Particularly, in design of our integration function, because a node always completely trusts itself, its M_t is set to one all the time, such that its comprehensive competency is equal to its pure data forwarding metric. The result is a node prefers to directly deliver data itself to the destination, unless it encounters the other node with a much stronger competency of meeting the destination node. The effect of "Self-Trusting" principle is reflected into the increased delay of delivering data, which will be particularly discussed in Section 5.5.

SELF-TRUSTING: Nodes trust themselves more than others in the forwarding behavior.

After evaluating the encounter's comprehensive competency, the node or its encountering node whoever has greater competence will take the responsibility of carrying data and is expected to successfully deliver the data with a higher probability in the near future.

Table 5.2: Parameters used for simulation

Parameters	**Values**
Network Area	1000×1000, 2000×2000, 3000×3000 (m^2)
Number of benign nodes	20
Number of malicious nodes	2,6,10,14
Communication Range	25 m
Mobility Pattern	RandomWayPoint
Node Speed	0-5 m per second
Message Life Time	30000 s
Data Generation Rate	1 message per 100 s
Simulation Time	860000 s
Seeds	1,2,3
w_{dir}	0.8
Δt	100 s
γ	0.5
P_{init}	0.75
ξ	0.98
ζ	0.25

5.5 Experimental study

In this section, we will first introduce our experiment configuration and then analyze our framework performance against black hole attack, where data are forwarded based on the PROPHET protocol [45] with(without) integrating with our framework. The PROPHET protocol with our framework is named as T-PROPHET in our experimental study. In addition, we will show some experiment results related to our framework properties, such as its stabilization and the effect of "Self-Trusting" principle.

5.5.1 Simulation Setup

We implemented our trust-based framework on the Opportunistic Network Environment simulator (ONE) [34], and some parameter values in our experiments are listed in Table 5.2. In our experiments we created benign and malicious nodes which move around in the network with the RandomWayPoint mobility model. The pairs of data source/destination are selected uniformly at random only from benign nodes for each message with a fixed generation rate. Given the same simulation time and a fixed message generation rate, the amount of data totally created is the same for all of our experiments. In our scenario, malicious nodes launch black hole attack by intercepting data from other nodes and then dropping them. To simulate the black hole attack, we assume that each malicious node is able to boost its delivery probability at the beginning of the network running. Thus, we set a high value of 0.9 to the probability. Moreover, we replicated each experiment configuration 3 times with independent seeds. Unless otherwise stated, each result presented hereafter is the average over the 3 replicas with 0.95 confidence interval.

We conducted two groups of experiments to evaluate our framework: 1) different attack strength–changing the number of malicious nodes in the network of the size 1000×1000 (m^2); 2) varying network density–configuring networks of different size while keeping a fixed number of nodes (20 benign nodes and 6 malicious nodes). All result analysis in the following subsections comes from the two groups of experiments.

5.5.2 The Ratios of Data Attracted and Delivered

To evaluate the effectiveness of our trust-based framework against black hole attack, we compare the network performance under PROPHET and T-PROPHET with respect to the ratios of data attracted and data delivered.

In Fig. 5.4 which presents the results from the first group of experiments by varying attack strength, we can see under the attack with a particular strength, the ratio of data attracted in PROPHET is greater than that in T-PROPHET. In addition, although the ratio of data attracted goes up with the increase of the attack strength in both of the protocols, the increasing in PROPHET is more tangible and much faster as compared to T-PROPHET. Similarly, as shown in Fig. 5.5 relevant to the second group of experiments with different network sizes, the attraction ratio of T-PROPHET is smaller than that of PROPHET.

In addition, since the basic functionality of MSNs is to deliver data by opportunistically pairwise node encounterings, the ratio of data finally delivered is also a critical metric in evaluating any data forwarding mechanisms in MSNs. In Fig. 5.6, we can see given a fixed number of malicious nodes, the ratio of data delivered in T-PROPHET is more than that in PROPHET in general. Note that although the ranges within which the delivery ratio is estimated to lie in PROPHET and T-PROPHET overlap a bit when there are two malicious nodes, we confirmed that for each seed, the delivery ratio of T-PROPHET is still much higher than that of PROPHET. Moreover, in Fig. 5.7, in the networks with different sizes, the delivery ratio of T-PROPHET is greater than that of PROPHET.

In summary, based on the experimental study on the ratios of data attracted and delivered, we could see the T-PROPHET outperforms PROPHET, which demonstrates the effectiveness of our trust-based framework against black hole attacks in MSNs with malicious wireless environment.

5.5.3 Framework Stabilization

In this subsection, still using the results from the two groups of experiments mentioned in the previous subsection, we will discuss the stabilization of our framework, which will indicate whether the damage resulting from black attack is under control. In our experimental study, we intend to show the change of the ratios of data attracted and data delivered over time in the experiments. We noticed that the trend of results under the same network configuration while with different seeds are similar, therefore, we just selected the results, where seed = 1, for illustration.

As shown in Fig. 5.8 regarding the first group of experiments, we notice that the attraction ratio in T-PROPHET keeps going down over time, which indicates that the number of data attracted is under control; however, the attraction ratio in PROPHET becomes stable at some point, which means that with more data being generated, still more data are attracted. The similar trend is illustrated in Fig. 5.9 when we vary the network size in the second group of experiments.

5.5.4 Self-Trusting Effect

As explained in Principle 5.4.3.2, since each node always assigns its trust value to one, they more trust themselves in the data forwarding behavior, which may lead to the delay of data delivery. Although in general, MSN is sort of Delay Tolerant Network (DTN), in which the delay is not a critical concern, we would like to demonstrate the effect of "Self-Trusting" in our experimental study. Therefore, we plot the delay of data delivered. In our scenario, the delay of a data message denotes the time period from message data generation until the message is delivered. Based on the two groups of experiments, we could see that the average delay of data delivered in T-PROPHET is larger than that in PROPHET, as shown in both Fig. 5.10 and Fig. 5.11.

5.6 Summary

In this chapter, we focused on data forwarding in MSNs in a malign wireless environment, where malicious nodes attempt to launch black hole attacks by dropping or arbitrarily forwarding data. In order to defend against such attacks, we proposed a trust-based framework, which can be readily integrated with a large family of existing data forwarding protocols in MSNs, such as PROPHET. We proposed Positive Feedback Message (PFM) as the evidence of the forwarding behavior of a node, which is fed into the Watchdog component in our framework. Additionally, we systematically introduced our trust-based framework for data forwarding in MSNs, which comprehensively evaluates an encounter's forwarding ability by integrating its forwarding reputation and the possibility of meeting the destination node. Through comprehensive experiments, we demonstrated the effectiveness of our proposed framework on improving the network performance under black hole attacks, and also demonstrated some features of our framework.

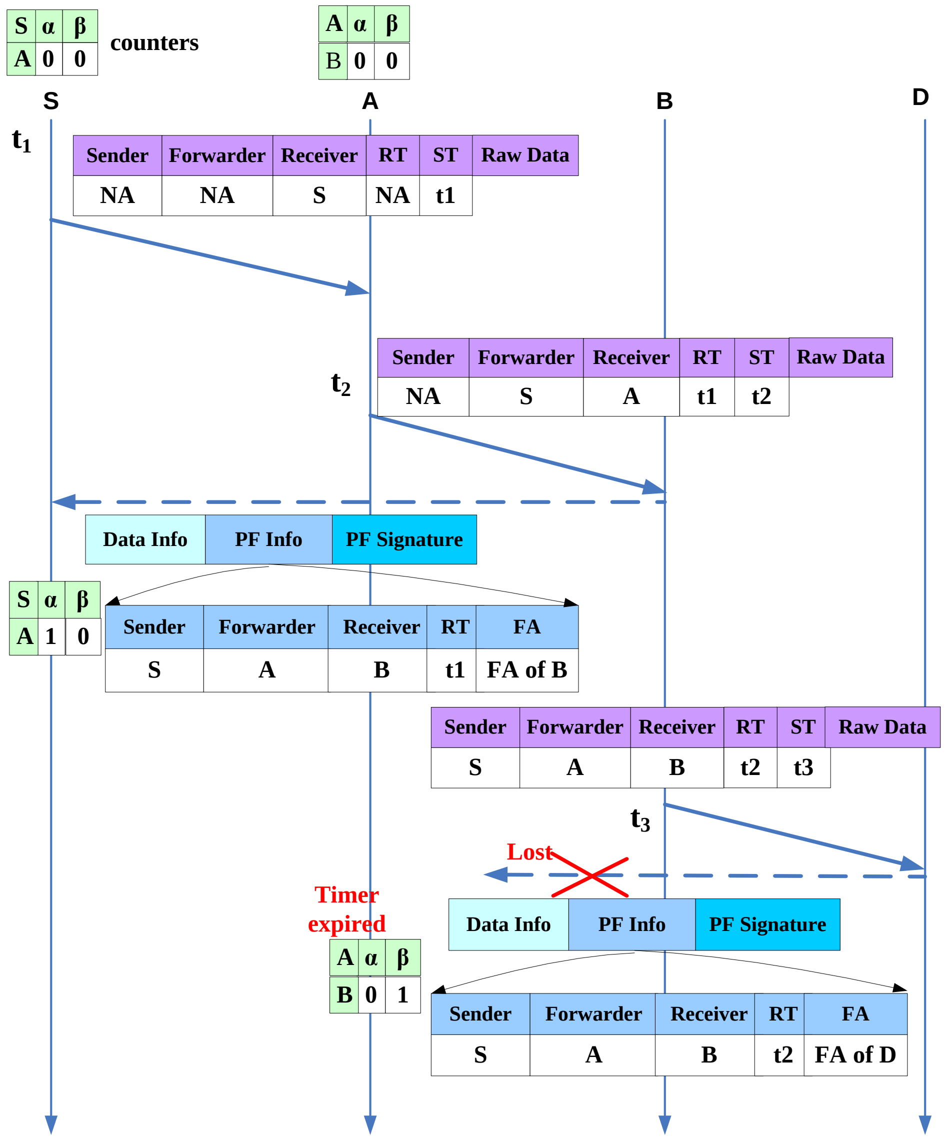


Figure 5.2: An example of the generation and transmission of PFM

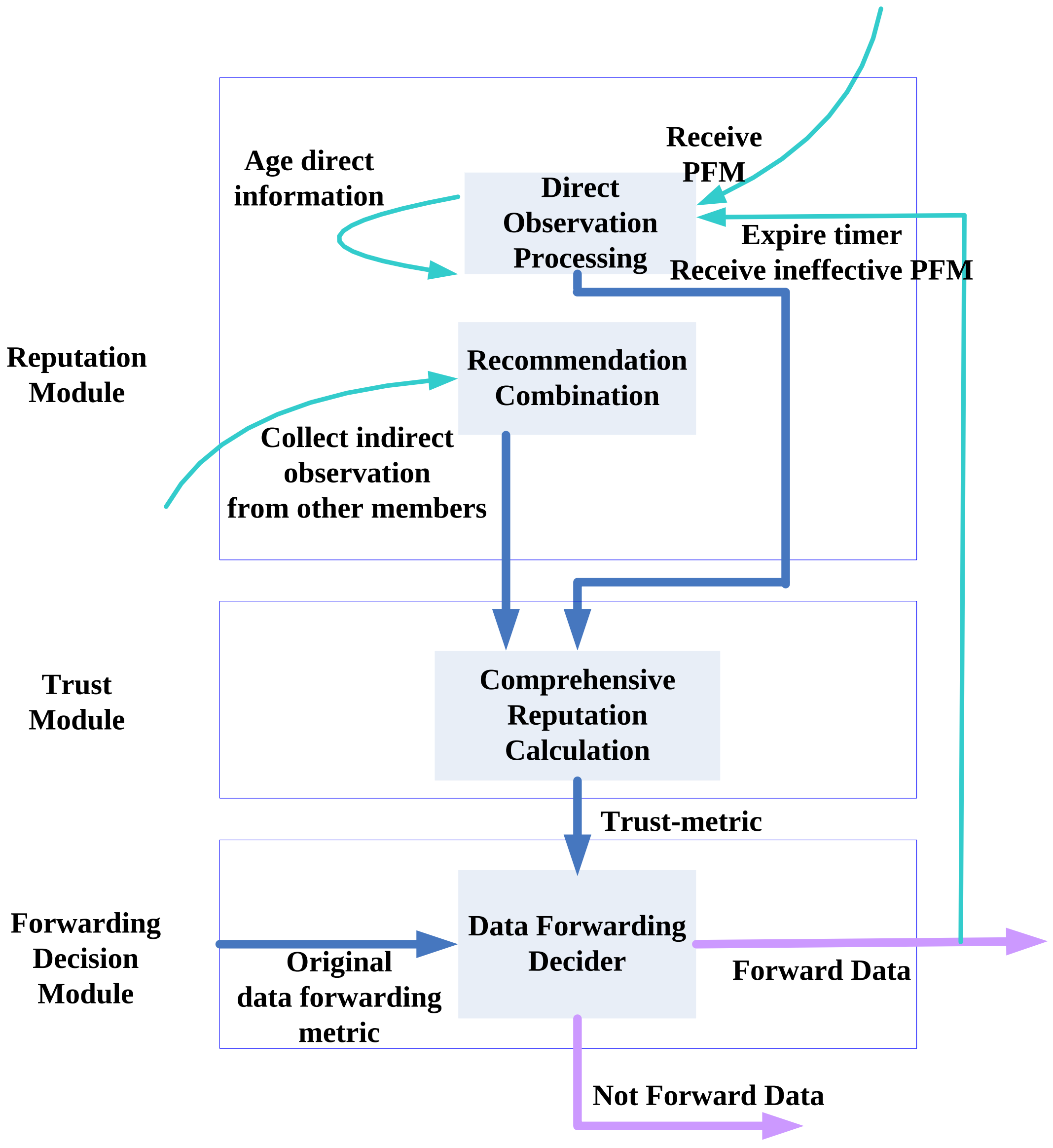

Receive PFM
Age direct information
Direct Observation Processing
Expire timer
Receive ineffective PFM
Reputation Module
Recommendation Combination
Collect indirect observation from other members
Trust Module
Comprehensive Reputation Calculation
Trust-metric
Forwarding Decision Module
Data Forwarding Decider
Original data forwarding metric
Forward Data
Not Forward Data

Figure 5.3: Our trust-based framework for data forwarding in MSNs

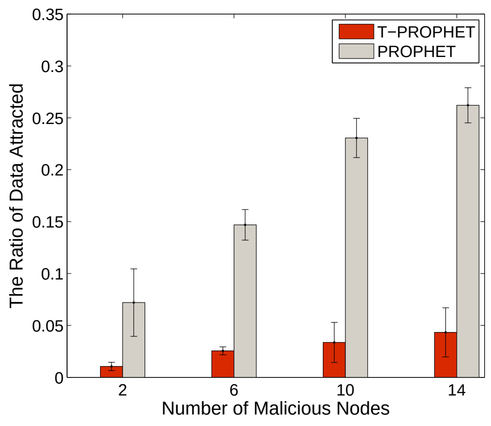


Figure 5.4: Attraction ratio with different attack strengths

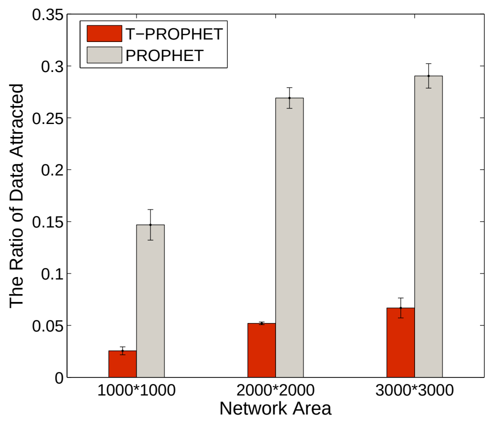


Figure 5.5: Attraction ratio with different network sizes

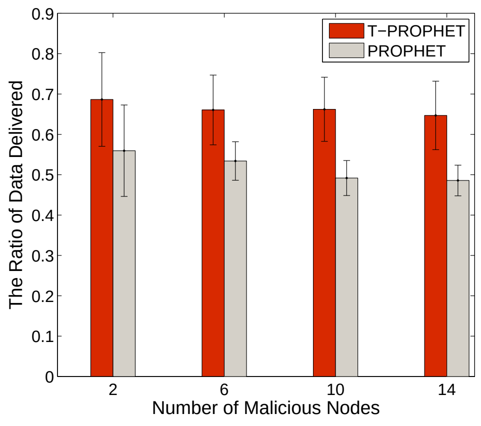


Figure 5.6: Delivery ratio with different attack strengths

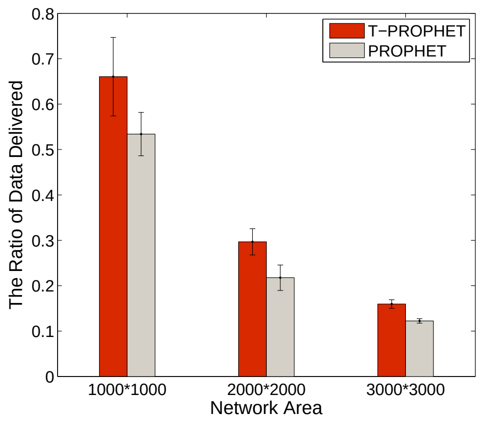


Figure 5.7: Delivery ratio with different network sizes

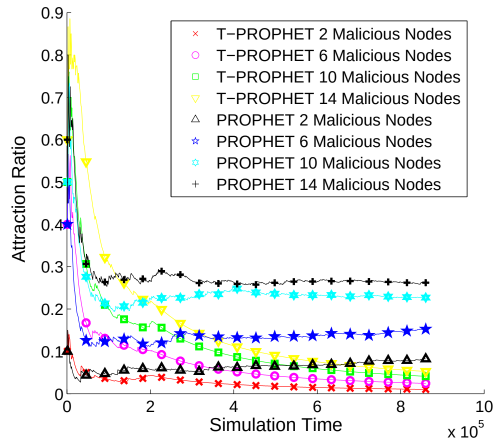


Figure 5.8: Attraction ratio under different attack strengths

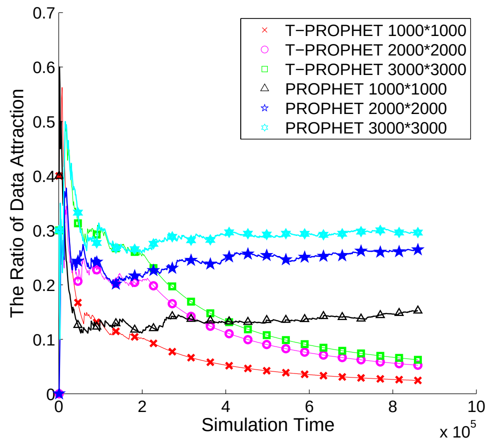


Figure 5.9: Attraction ratio in networks of different sizes

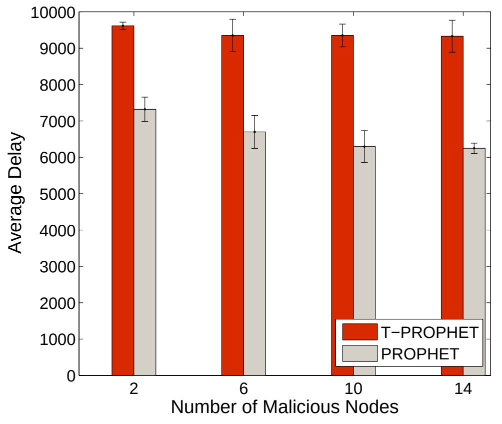


Figure 5.10: Average delay with different attack strengths

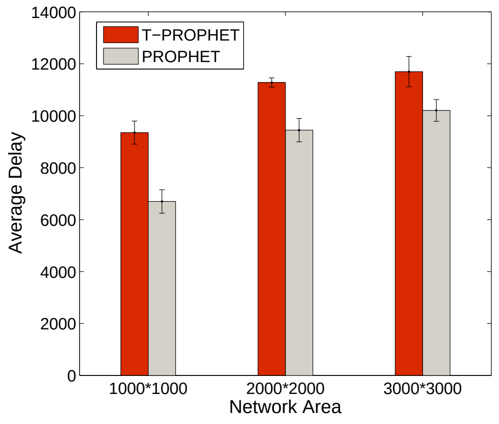


Figure 5.11: Average delay with different network sizes

CHAPTER 6

SOCIAL-AWARE CONGESTION CONTROL IN MSNs

In recent years, great research effort has been made on controlling traffic congestion in MSNs which are derived from Delay Tolerant Networks (DTNs). The occurrence of traffic congestion in MSNs is caused by several reasons. In addition to the ones which have already been researched on traditional networks, such as the Internet, a unique reason in MSNs is applying the social-aware data forwarding protocols to facilitate data delivery. In design of such protocols, the nodes which more frequently encounter with other nodes in the network, namely high-centrality nodes, will be selected as data forwarders with a higher probability. The intuition behind the protocol design is since high-centrality nodes in our social community have a higher chance to meet others, they should also more likely encounter the destination nodes, thus becoming good candidates for data forwarding. However, if all data traffic is forwarded towards the high-centrality nodes which usually has a limited number in a social community, they will get congested soon and become the bottleneck of network communication.

To date, although a multitude of work has addressed congestion control in MSNs from different angles, ranging from migrating data to avoiding data transmission along congested paths, we believe that slowing down the data generation rate at data sources should be the most robust approach to controlling congestion. This is because even for applying migration-based or avoidance-based approaches, there always exists a condition where almost all of the nodes are seriously congested, while data sources still keep high data generation rates without any awareness of the network congestion. A couple of techniques have been proposed to control congestion on the Internet by adjusting traffic generation rate at data sources. The common design principle of these techniques is to notify a data source of the traffic congestion at a particular node by routing a control message back along the data transition path from the congested node. However, directly applying these techniques to controlling the network congestion in MSNs is impractical due to the lack of the contemporary path between any pair of nodes which hurdles the end-to-end transmission of a congestion control message. The other aspect of the congestion problem in MSNs, which is different from the traditional networks is if high-centrality nodes get congested, the entire network may not function for data delivery. Therefore, instead of notifying a particular data source of the congestion situation, all data sources in the network should be aware of it. In this chapter, we propose a novel congestion control framework which leverages social influence to propagate congestion signals to notify all data sources of the congestion condition at high-centrality nodes in an MSN.

This chapter is organized as follows. Section 6.1 provides an overview of the basic function and the design principle of our congestion control framework. Sections 6.2, 6.3 and 6.4 correspondingly introduce the three modules which are composed of our framework, namely self-determination module, signal-collection module and signal-update module. The experimental study is addressed in Section 6.5, followed by a chapter summary in Section 6.6.

6.1 The Overview of Our Congestion Control Framework

As high-centrality nodes in MSNs are the bottleneck of network communication in social-aware data forwarding protocols, we care more about the congestion condition at these nodes. This is different from the congestion concern in the conventional networks where congestion control mechanisms focus on each individual node to ensure them not to get congested. The essence in design of our congestion control framework is to notify all data sources of the occurrence of traffic congestion at high-centrality nodes so that they can adjust their data generation rate to alleviate the network congestion. The challenge in our design is how to make data sources aware of the network congestion in a more real-time manner in MSNs characterized by long delay and frequent partition. Moreover, once the congestion is relieved at high-centrality nodes, data sources should also be notified quickly so as to raise their data generation rate back aimed to guarantee the network throughput.

A straightforward approach is to let high-centrality nodes create and flood control messages over the entire network to notify data sources of the dynamics of the congestion situation. Apparently, this naive mechanism requires to introduce a large number of control messages into the network. Furthermore, if the information carried by a control message is static, it cannot synchronize with the real network congestion condition, such that it is still possible to notify a data source with an out of date control message.

In order to cope with the aforementioned challenge, we propose a congestion control framework which leverages social influence to spread congestion condition at high-centrality nodes to explicitly warn data sources. Specifically, we define a *congestion signal* of binary value for each node and make the nodes carry their signals and propagate them over the network. A signal is intended to indicate the congestion condition at high-centrality nodes and its value is updated periodically at individual nodes based not only on a node's local determination on traffic congestion but also on the signals collected from other nodes it encountered. Basically, our proposed framework is composed of three modules, *self-determination module*, *signal-collection module* and *signal-update module*, as shown in Figure 6.1. Our framework is designed to control traffic in a distributed manner to fit the features of MSNs of long delay

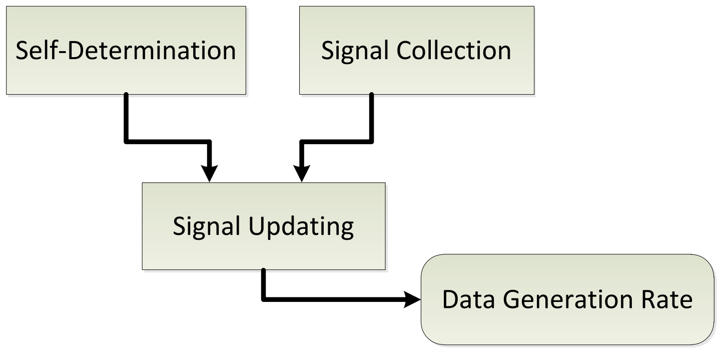


Figure 6.1: Our congestion control framework

and frequent partition. Our framework can be built upon any existing social-aware forwarding protocols in MSNs to balance network throughput and traffic congestion. In the following sections, we will detail the function of each of the three modules in our framework. Note we divide the network lifetime into epoches with the same time interval in our framework design and experimental study.

- In the *self-determination module*, each node determines its own congestion condition in each epoch based on its buffer usage and the number of data forwarded to it by its encountering nodes.
- In the *signal-collection module*, each node collects the congestion signals from other nodes when it encountered.
- In the *signal-update module*, each node takes its self-determination on its local congestion situation and the signals collected from its encounters as input to comprehensively estimate the network congestion at hight-centrality nodes and update the signal value at the end of each epoch. The data generation rate at data sources will be adjusted in accordance with their updated signals.

6.2 Self-determination Module

In the self-determination module, each node determines its congestion condition according to its local data traffic. The output of this module is a binary value which will be fed into the signal-update module to derive a new congestion signal value which can more accurately capture the network congestion condition at centrality nodes. The self-determination module produces the output at the end of each epoch before the signal updating proceeds.

A straightforward idea of determining a node's congestion situation is based on its buffer usage, calculating the ratio of the time period when the node's buffer is full to the time interval of an epoch. However, this approach may not effectively demonstrate a node's congestion condition. An extreme example is suppose a node has a full buffer over the entire epoch in MSNs. However, since none of its encountering nodes intend to forward data to the node, the node is not congested. Therefore, solely leveraging the buffer usage at a node does not suffice to accurately estimate the traffic congestion at the node.

Before we start to introduce our approach to determining a node's congestion condition, let us first clarify the model for buffering data at individual nodes. We assume when data is forwarded to a node with a full buffer, the message with the oldest time stamp in the buffer will be dropped. Similarly, if new data is created at a data source node which no longer has buffer room, the oldest data in the buffer will be removed for storing the new data. We assume all data have the same size through our discussion in this chapter.

$$s_{self_i} = \frac{num_{f_i}}{num_{si}} \tag{6.1}$$

In the module, node i maintains two counters, num_{si} and num_{f_i}. num_{si} records the total number of data forwarded to node i from other nodes as well as the data created at node i itself. num_{f_i} records the number of data dropped at node i due to its full buffer. We define s_{self_i} to represent node i's self-evaluation on congestion condition based on Equation 6.1. The value of s_{self_i} directly derived from Equation 6.1 is in the range [0,1], but we convert it to a binary value to prepare for the procedure of information aggregation in the signal-update module. Specifically, we preset a threshold value α: if $s_{self_i} > \alpha$, we set s_{self_i} to 1, indicating the node is congested; otherwise, it is assigned 0, showing the normal traffic condition at node i.

6.3 Signal Collection Module

The main function of the signal-collection module is for each node to collect information from its encountering nodes, including their congestion signals and their centralities, which will be used for signal aggregation in the signal-update module. The procedure of collecting signals happens whenever two nodes move into each other's communication range. They exchange their signals as well as their centralities and then store them locally. One can see the overhead introduced by the information exchange is trivial as only one piece of such information is exchanged between any two encountering nodes. Furthermore, the amount of the exchanged information is small, which can be represented by $1 + \lg(n-1)$ bits. Suppose there are n nodes in the network, then the maximum centrality of a node is $n-1$. Therefore, $1+\lg(n-1)$ bits are enough for presenting the exchanged information, where 1 bit for the congestion signal and the $\lg(n-1)$ bits for node centrality. Furthermore, as the nodes will exchange Hello messages when they meet to build their communication connection, the light information of $1 + \lg(n-1)$ bits can piggy-back on the Hello message.

The node centrality is calculated at individual nodes in a distributed manner to fit the features of MSNs of long delay and frequent partition, and it is updated periodically. We employ the CWindowCentrality approach [29] to calculate the centrality for each node, in which the number of unique nodes seen by a node within a time unit is used to approximate the node centrality. We derive the final centrality for a node by averaging its centralities within a couple of consecutive time units. Note the period of updating node centrality is not necessarily the same as the epoch. When two nodes encounter, they simply exchange their centralities which are recalculated at the end of the last period.

6.4 Signal Update Module

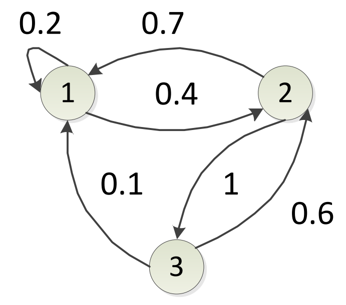
The signal-update module is the core of our congestion control framework as it controls the dynamics of congestion signals which is intended to represent the network congestion condition at high-centrality nodes. This module functions at the end of every epoch, helping each node update its signal based not only on the node's local determination on traffic congestion but also on the congestion signals collected from its encountering nodes in the current epoch. The signal-update procedure essentially aggregates all collected information together to derive a more accurate signal to capture the network congestion condition. In this section, we will first detail the design of our signal updating module and then address how the updated signal will impact data generation rates at data sources.

6.4.1 Procedure of Updating Signals

We employ *Influence Theory* [6] to model the signal-update procedure. The influence model is a tractable mathematical representation of random, dynamical interactions on networks. Specifically, an influence model consists of a network of nodes, each with a status that evolves over the time. The evolution of the status at a node is based on an internal Markov chain, but with transition probabilities depending not only on the current status of that node but also on the statuses of the neighboring nodes. The influence among the nodes occur probabilistically, starting once a change of status at one node alters the transition probabilities of its neighbors, which then alter those of their neighbors, and so on. The design of our signal updating procedure is based on the intuition that the congestion signal carried by an encountering node with a higher centrality should influence more on a node's estimation for the network congestion in MSNs. In the following, we will first briefly introduce the basics of Influence Theory, particularly focusing on *Binary Influence Model* and then discuss our model of updating congestion signal.

6.4.2 Binary Influence Model

In the binary influence model, each node has a binary status. Mathematically, the status of node i at time point k is $s_i[k]$, either 0 or 1. At each time point k, the statuses of all nodes form a binary vector $\mathbf{s[k]}$. In addition, an $n \times n$ stochastic matrix M is built in the model, called *the network influence matrix*, where n is the number of nodes in the network. The matrix presents the probability of a node's status being impacted by other nodes and is constructed according to the graph $\Gamma(M)$, called *the network influence graph*, which shows the influence among nodes in the network. In $\Gamma(M)$, if node i has an influence on node j, there is a directed edge pointing to node j from node i, labeled a weight which indicates the amount of influence node i exerts on node j. The total amount of the influence received by any node is equal to the sum of incoming edge weights, which is required to be one in the model to make M a stochastic matrix. See the example in Figure 6.2, which shows the mapping between the influence network graph and the influence matrix. Given the influence matrix



$$M = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0.2 & 0.4 & 0 \\ 2 & 0.7 & 0 & 1 \\ 3 & 0.1 & 0.6 & 0 \end{array}$$

Figure 6.2: An example of the binary influence model

and the status vector for all nodes in the network at a time point k, we can derive at time point $k + 1$ the possibility of the status vector for all nodes in the network by using Equation 6.2. $s[k] \times M$ is a vector of probabilities of any node updating its status to 1, on which the classic probability function *Bernoulli* is applied for generating a status for each node.

$$s[k+1] = Bernoulli(s[k] \times M) \tag{6.2}$$

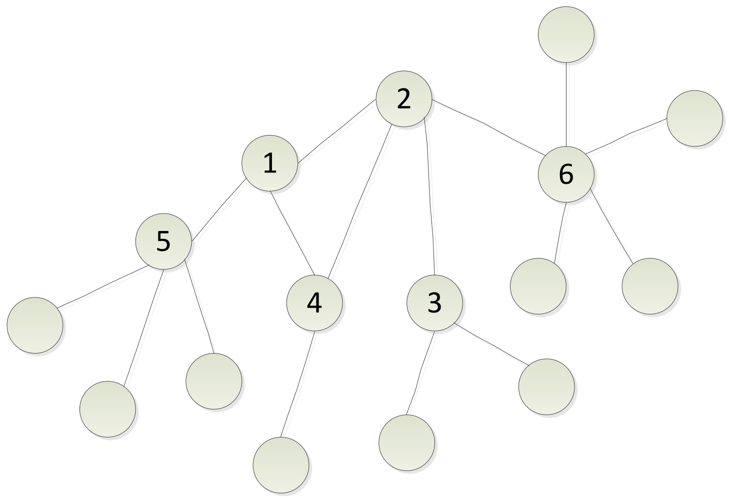
6.4.3 Signal Update Model

Our sinal-updating model is based on the binary influence model. In our model, each node is in one of the two possible statuses at each discrete-time instant, *congested* or *normal*, which is represented by 1 and 0, respectively. Here we emphasize again that the status of a node does not indicate the congestion situation at that node, instead, it presents the congestion condition at hight-centrality nodes in the entire network.

In MSNs, one node can influence the other node only when they encounter each other, therefore, we build the network influence graph based on the records of nodes' encountering history. Specifically, at the end of each epoch, the signal-update module pictures the influence graph for each node according to their encountering records which occurred in that epoch. Each record is associated to two edges with the opposite direction in the influence graph, showing the mutual influence between two nodes. In addition, we create a self-loop edge for each node in the graph, indicating how the current status of the node impacts its status in the next epoch. There are two unique features of our modeled influence graph, *dynamic* and *local*. Due to the mobility of nodes in MSNs, a node's encountering nodes vary in different epochs, which leads to the dynamic change of the influence graph. Moreover, because of the features of long delay and frequent partition in MSNs, each node can picture only part of the influence graph which is formed by the nodes that directly impact it. An example is illustrated in Figures 6.3 and 6.4. Suppose the graph in Figure 6.3 is the global influence graph, then the two graphs shown in Figure 6.4 are the local influence graphs associated with node 1 and node 2, respectively.

After constructing the topology, the next step to draw an influence graph is to weigh the edges based on the amount of influence that one node exerts on the other node. The essence of our design is since the bottleneck in the social-aware data forwarding protocols in MSNs is the nodes with high-centrality, the congestion signals carried by these nodes should have a greater influence on other nodes' inferring the network congestion condition. Therefore, we associate edge weights with the node centralities. In this chapter, we equate the node centrality with the node degree, and from now on we use them interchangeably.

The detailed procedure of deriving the weight for an edge is given as follows. First of all, we define a node set Q_i for each node i, including all of node i's encountering nodes in the epoch as well as node i itself. Second, by Equation 6.3, we normalize the node degree for any node q in Q_i, which was collected along with node



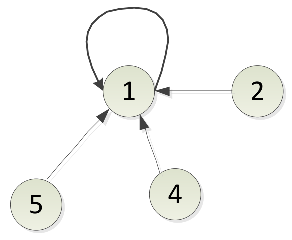
Node ID	1	2	3	4	5	6	…	…
Congestion Signal	0	1	1	0	0	1		
Self-Evaluation	1	1	0	0	0	0		
Centrality (i.e., Degree)	3	4	3	3	4	5		

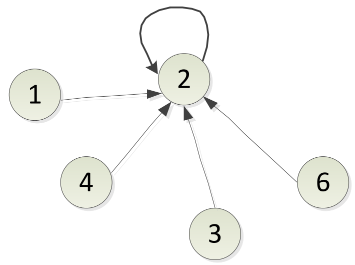
Figure 6.3: An example of a full influence graph

q's congestion signal in the signal-collection procedure. Evidently, the higher the centrality of a node, the larger its $f_c(\cdot, i)$ value. Additionally, the sum of $f_c(\cdot, i)$ for all nodes in Q_i is equivalent to one. Therefore, $f_c(\cdot, i)$ can be used to reflect the amount influence any node in Q_i can exert on node i. We define a vector with all $f_c(\cdot, i)$, which is called node i's *influence vector*.

$$f_c(q, i) = \frac{d_q}{\sum_{j \in Q_i} d_j} \tag{6.3}$$

where q is any node in Q_i and d_q is node q's degree.





ID	s[k]	d	Influence to node 1
1	1	3	3/(3+4+3+4)=3/14
2	1	4	4/14
4	0	3	3/14
5	0	4	4/14

ID	s[k]	d	Influence to node 2
1	0	2	2/(2+4+3+2+5)=2/16
2	1	4	4/16
3	1	3	3/16
4	0	2	2/16
6	0	5	5/16

Figure 6.4: The analysis of two nodes

Let's see the example again which is illustrated in Figures 6.3 and 6.4. Suppose in the network node 2 and node 4 met first, and then node 1 encountered node 2, followed by node 1 seeing node 4. Note the sequence of node encounterings does not make any difference in the topology of the influence graph, while it impacts the information collected at individual nodes, such as the encountering node's centrality. For example, node 1's degree was 2 when it met node 2; however, it became 3 when it saw node 4. Therefore, the centralities which node 1 provided to node 2 and node 4 are different. If we change the sequence, the centralities node 1 provides to node 2 and node 4 may also be different, which leads to the difference in calculating the edge weights in the influence graph. In the two tables displayed in Figure 6.4, the two columns of d and the influence to node 1(node 2) respectively list the degree of the encountering nodes observed locally and their influence to node 1(node 2) after the degree normalization.

In additional to the local influence graph, each node has a local status vector $\mathbf{s[k]}$. The value of $\mathbf{s[k]}$ at node i includes not only the signals collected from its encountering nodes but also $s_i[k]$ which is equal to s_{self_i} - the binary value output from node i's self-evaluation module as we discussed in Section 6.3. Note that $\mathbf{s[k]}$ at different nodes may have different sizes as it is determined by the number of nodes a node encountered. Suppose a node encountered 4 other nodes in one epoch, the size of its $\mathbf{s[k]}$ is 5 after considering the node itself. Given $\mathbf{s[k]}$ and the influence graph, the possibility of updating a node's signal to "congested" can be derived by the multiplication of the two vectors, $\mathbf{s[k]}$ and the influence vector, rather than $s[k] \times M$ in Equation 6.2. In our example, node 1 has a probability of 7/14 to generate a signal in congestion status while node 2 has a chance of 7/16.

6.4.4 Impact of Signal Updating

A data source will adjust its data generation rate in terms of its updated congestion signal. We set an initial rate for all data sources. If the congestion signal at a data source is 0, the data generation rate is the same as the initial value; otherwise, the rate is cut in half of the previous rate. Formula 6.4 shows the data generation rate of any data source in the t^{th} epoch. We can see if a data source has signal of 1 for a number of consecutive epochs, the data generation rate will decrease exponentially. But whenever a data source gets its congestion signal back to 0, its generation rate will become the initial assignment.

$$r_t = \begin{cases} r_0, signal = 0 \\ \frac{r_{t-1}}{2}, signal = 1 \end{cases} \tag{6.4}$$

Table 6.1: Parameters for experimental study

Parameters	**Values**
data generation interval	400s
message size	20K
buffer size	2M
Message TTL	infinity
time unit for centrality	7200s
number of time units to average centrality	3
interval for updating centrality	600s
epoch interval	5000s
threshold for determining self-congestion	0.1

6.5 Experimental Study

To verify our social-influence based congestion control framework, we implemented it on ONE [34](a DTN simulator). Additionally, we coded a simple social-aware data forwarding protocol, in which the criteria of choosing a next forwarder is to first check whether the encountering node has a higher probability of meeting the destination, and if not, then check its centrality which is calculated based on the CWindowCentrality [29]. We conducted experiments on two real trace data sets:

- In Infocom 05, the devices were distributed to approximately fifty students attending the Infocom student workshop. Participants belong to different social communities (depending on their country of origin, research topic, etc.). However, they all attended the same event for 4 consecutive days and most of them stayed in the same hotel and attended the same sections.
- In Cambridge 05, the authors used Intel iMotes to collect the data. The iMotes were distributed to students of the University of Cambridge and were programmed to log contacts of all visible mobile devices. The number of devices that were used for this experiment is 12. This data set covers 5 days.

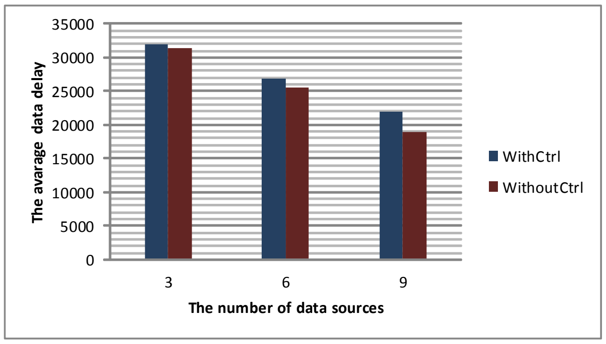


Figure 6.5: Cambridge05 - Average delay

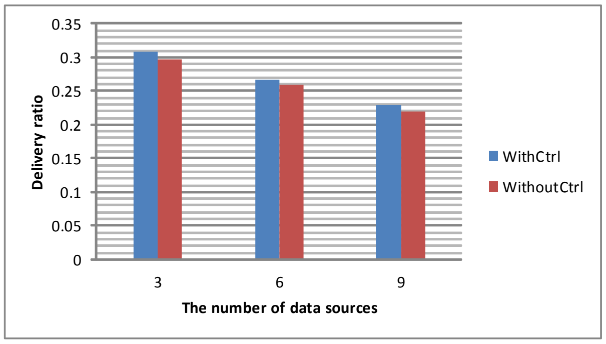


Figure 6.6: Cambridge05 - Delivery ratio

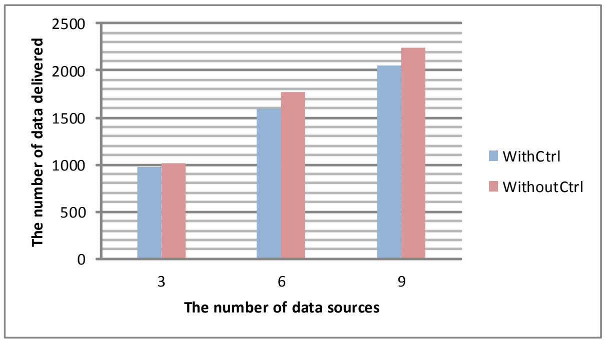


Figure 6.7: Cambridge05 - The number of data delivered

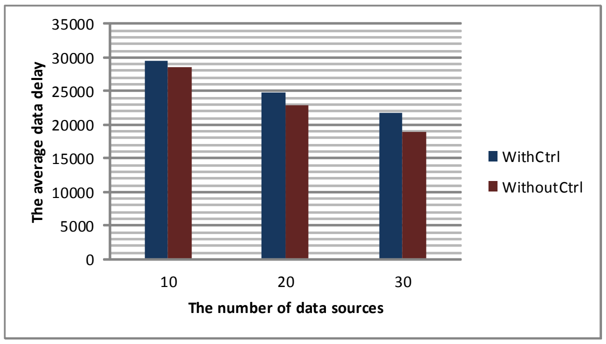


Figure 6.8: Infocom05 - Average delay

The parameters of the network configuration and their corresponding values are listed in Table 6.1.

We conducted many groups of experiments by randomly selecting 10, 20, 30 data sources for Infocom05 data set and 3, 6, 9 data sources for Cambridge05 data set. Each data generated at a data source is assigned a destination node which is randomly selected. Under the same configuration setting, we ran the simulation three times with three different seeds which control the randomness in the experiments, for example, the selection of data sources. The data plots in the following show the average of the results collected from the three corresponding experiments. We compare the two case scenarios with/without our congestion control framework, which are named WithCtrl and WithoutCtrl, respectively.

We focus on two metrics, delivery ratio and the number of data delivered, to evaluate our congestion control framework. The delivery ratio is referred to the number of data delivered divided by the number of data created, which demonstrates the performance of a protocol with our congestion control framework with respect to data forwarding. However, the number of data delivered can show the network throughput which is a traditional measurement to evaluate a congestion control mechanism. In

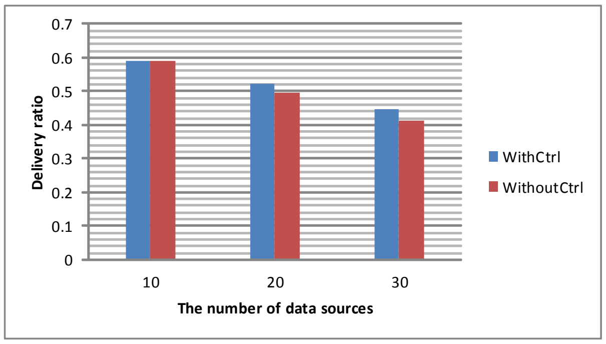


Figure 6.9: Infocom05 - Delivery ratio

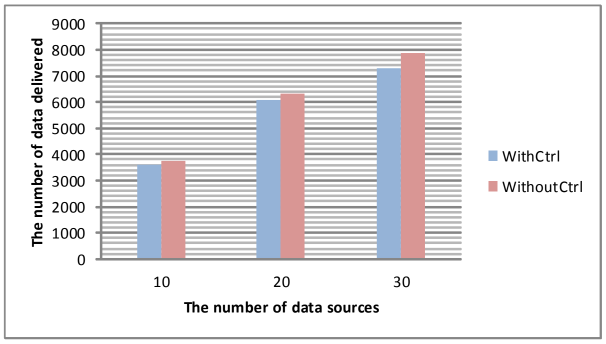


Figure 6.10: Infocom05 - The number of data delivered

addition to the two important metrics, we also show the average delay of delivered data.

From Figures 6.6 and 6.9, we can see the delivery ratio by the protocol with our congestion control framework is higher than the ratio by the protocol without it, although the difference isn't tangible. We believe a real trace data set which can better capture the topological features of social networks can be used for our experiments to better show the advantage of applying our congestion control framework in MSNs. Figures 6.7 and 6.10 show our evaluation of our framework with respect to network throughput. The protocol with our framework has a smaller throughput than the protocol without our framework. For the WithoutCtrl protocol, although it drops a lot of data, it creates data whenever possible, which increases the chance of having more data delivered. However, our congestion control framework cuts in half of the data generation rate of a data source when its congestion signal becomes 1 and will not update it until the end of the next epoch, which may miss a lot of chance of generating new data even though the node has room in its buffer. Therefore, the suboptimal scheme of adjusting data generation rate in WithCtrl is the reason for having less throughput by our congestion control framework. Moreover, as shown in the Figures 6.5 and 6.8, the average delay of delivered data by WithoutCtrl is shorter than that by WithCtrl. The reason is that as a lot of data with old stamps are dropped in WithoutCtrl, most likely, the delivered data are quite new, thereby leading to less delay.

6.6 Summary

In this chapter, we introduced how to apply social influence to control congestion in MSNs. Specifically, we defined a congestion signal for each node which captures the congestion condition at high-centrality nodes in the network. We built a congestion control framework using Binary Influence Model to spread the congestion signals. Each node aggregates the congestion signals collected from its encounter nodes to update its own signal. We implemented our congestion control framework and conducted experiments on real trace data sets. Although not all results are positive, we believe using a real trace set which can captures the topological features of social networks, like the limited number of high-centrality nodes, will help better demonstrate the advantage of applying our framework to control traffic congestion in MSNs.

CHAPTER 7

CONCLUSION AND FUTURE WORK

With the advent of Web 2.0 and advanced wireless techniques, a huge amount of data have been being collected from human beings' daily activities, like surfing the Internet or chatting through mobile phones. Using these data, we can infer what people did before, what they are doing, and even predict what they are going to do in the near future. This dissertation mainly focused on handling the data which can be used to infer the relationships among people, such as friendships or trust relationships. In particular, we studied the human relation networks we derived from the relevant data collected from our digital world, such as online social networks and mobile social networks, and researched on how to securely and effectively use the relationship networks by studying four research problems. In the following, we briefly draw a conclusion for each of the four problems and correspondingly shed a light on future work.

(1) We discovered a more realistic attack against users' relation privacy in publishing OSN data, in the worst case of which some users' identities may be disclosed. We noticed the weakness of k-degree anonymity technique in preserving users' relationship privacy and proposed a new privacy preservation model, called ℓ-diversity. In order to defend against the attack we modeled, we proposed three graph-manipulation based anonymization techniques, MaxSubB, MaxSub and MinSuper. We proved the graphs anonymized by our techniques to be ℓ-diversity anonymous. After conducting experiments on real-world data sets, we demonstrated good utility preserved by our anonymization techniques in the published data.

In this work, we only considered the preservation of users' relationship privacy in one-time OSN publishing which is a snapshot of the dynamic OSN. If the third-party requests a sequence of OSN publishings, we need to design a more efficient anonymization technique. In particular, instead of applying our anonymization techniques individually in each publishing of OSN data, we should consider how to adjust the OSN data published previously to protecting newly constructed relationships in the OSN in the current publishing. The goal is to reduce the time spent on data anonymization, as time consumption is critical especially for the large-scale data set, like Facebook.

(2) From the viewpoint of a third-party analyst, we discussed how to leverage the topological features of social networks, such as small-world, scale-free and the good-connectivity of high-degree nodes, to efficiently discover the connectivity of the subgraph which include a group of target users on OSNs. Furthermore, we took the cost of the subgraph discovery into consideration. We delicately interpreted the original problem of local-view based subgraph detection with cost concern into a problem which sequentially handles the two points of "minimum" - discovering a subgraph of minimum size with a minimum cost. To solve this challenging problem, we proposed two searching techniques, Unbalanced Multiple-Subgraph(UMS) and Balanced Multiple-Subgraph(BMS), which can quickly discover the subgraph connectivity. We conducted experiments on the real-world data sets and showed that BMS even outperforms UMS. Furthermore, the experimental results demonstrate that the well-connectivity of OSNs is for all nodes in OSNs rather than only limited to high-degree nodes, as we can find the connectivity of any group of users by a small query cost in searching.

Although this work researched only on discovering subgraph connectivity on OSNs from the perspective of the third-parties, it demonstrates the effectiveness of using topological properties of social networks in searching on OSNs. Therefore, we can also research some other graph mining problems leveraging these properties, for example, to sample graphs for estimating the size of OSNs.

(3) After noticing the possibility of constructing fake relations to damage network performance in MSNs, we proposed a trust-based framework which can be flexibly integrated with a large family of existing single-copy data forwarding protocols in MSNs, aiming at comprehensively evaluating an encounter's ability to deliver data. With the help of our proposed framework, not only black hole attack but also arbitrarily forwarding attack can be counteracted effectively.

In the attack model in this work, we assumed malicious nodes to launch attacks independently. However, a stronger attack would allow some malicious nodes to launch a collusive attack. In this case, solely using our reputation-based frame is not sufficient to ensure the reliability of forwarding data in MSNs, therefore, some other techniques are called for to defense against this stronger attack.

(4) In order to deal with network congestion in MSNs which is caused by the heavy traffic concentration at high-centrality nodes in data forwarding, especially when the social-aware data forwarding protocols are applied, we designed a social-influence based congestion control framework. The framework effectively spreads the congestion condition at high-centrality nodes to the entire network to notify data sources of the congestion situation so that they can adjust data generation rate to relieve network congestion.

In the future, we need to look for a better scheme of adjusting data generation rate when a data source's signal becomes 1 aiming at increasing the network throughput. Furthermore, we should look into the associations between the mobility pattern of nodes, data generation model(including the model to select pairs of source and destination) and the influence model, and theoretically analyze them to validate the effectiveness of our congestion control model.

REFERENCES

[1] http://vlado.fmf.uni-lj.si/pub/networks/pajek/.

[2] Stanford large network dataset collection. http://snap.stanford.edu/data/.

[3] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.

[4] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review E*, 64(4), 2001.

[5] T. Anantvalee and J. Wu. Reputation-based system for encouraging the cooperation of nodes in mobile ad hoc networks. In *Proc. of ICC'07*, pages 3383–3388, 2007.

[6] C. Asavathiratham. The influence model: A tractable representation for the dynamics of networked markov chains. In *Dept. of EECS (Ph.D. Dissertation), MIT, Cambirdge*, 2000.

[7] S. Asur and S. Parthasarathy. A viewpoint-based approach for interaction graph analysis. In *Proceedings of ACM SIGKDD*, pages 79–88, 2009.

[8] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW'07*.

[9] V. Balakrishnan, V. Varadharajan, P. Lucs, and U. K. Tupakula. Trust enhanced secure mobile ad hoc network routing. In *Proc. of AINAW'07*, pages 27–33, Canada, 2007.

[10] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. The benefits of facebook friends: social capital and college students use of online social network sites. *J. Computer-Mediated Communication*, 12(4):1143–1168, 2007.

[11] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Class-based graph anonymization for social network data. In *Very Large Databases Endowment*, volume 2, pages 766–777, 2009.

[12] C. Boldrini, M. Conti, and A. Passarella. Exploiting users' social relations to forward data in opportunistic networks: the HiBOp solution. *Elsevier Pervasive and Mobile Computing*, 4(5):633–657, October 2008.

[13] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: routing for vehicle-based disruption-tolerant networking. In *Proc. of INFOCOM'06*, pages 1–11, 2006.

[14] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *PinKDD'08*.

[15] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SIAM DM'04*.

[16] J. Cheng, A. W.-C. Fu, and J. Liu. K-isomorphism: Privacy preservation in network publication against structural attack. In *SIGMOD'10*, 2010.

[17] J. Cheng, Y. Ke, W. Ng, and J. X. Yu. Context-aware object connection discovery in large graphs. In *Proceedings of IEEE ICDE*, pages 856–867, 2009.

[18] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. In *PVLDB'08*.

[19] M. E. Daly and M. Haahr. Social network analysis for information flow in disconnected delay-tolerant manets. In *IEEE Transactions on Mobile Computing*, volume 8, pages 606–621, May 2009.

[20] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *Proceedings of NDSS*, 2009.

[21] S. Das, O. Egecioglu, and A. Abbadi. Anonymizing edge-weighted social network graphs. In *tech. report CS-2009-03, Computer Science Dept., Univ. of California, Santa Barbara*, 2009.

[22] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *Proceedings of ACM SIGKDD*, pages 118–127, 2004.

[23] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM TOSN*, 4(3):633–657, May 2008.

[24] M. Garey and D. Johnson. *Computers and Intractability.* Freeman, 1979.

[25] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proceedings of IEEE INFOCOM*, 2010.

[26] A. Grundy and M. Radenkovic. Decongesting opportunistic social-based forwarding. In *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, pages 82 –85, Feb. 2010.

[27] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural reidentification in anonymized social networks. In *PVLDB'08*.

[28] D. Hua, X. Du, L. Cao, G. Xu, and Y. Qian. A dtn congestion avoidance strategy based on path avoidance. In *Second International Conference on Future Computer and Communication (ICFCC), 2010*, pages 21–24, May 2010.

[29] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proc. of ACM MobiHoc'08*, 2008.

[30] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. In *IEEE Transactions on Mobile Computing*, Dec. 2010.

[31] A. Josang. Trust-based decision making for electronic transactions. In *Proc. of NORDSEC'99*, Stockholm University, Sweden, 1999.

[32] G. Kasneci, S. Elbassuoni, and G. Weikum. Ming: mining informative entity relationship subgraphs. In *Proceedings of ACM conference on Information and knowledge management*, 2009.

[33] L. Katzir, E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *Proceedings of WWW*, 2011.

[34] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proc. of SIMUTools '09*, New York, NY, USA, 2009. ICST.

[35] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity graphs in networks. *ACM Transactions on Knowledge Discovery from Data*, 1(3), December 2007.

[36] C. Korte and S. Milgram. Acquaintance networks between racial groups: Application of the samll world method. *Journal of Personality and Social Psychology*, 15(2):101–108, 1970.

[37] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15(2):141–145, 1981.

[38] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[39] F. Li and J. Wu. Mobility reduces uncertainty in MANETs. In *Proc. of INFOCOM'07*, pages 1946–1954, 2007.

[40] F. Li and J. Wu. Thwarting blackhole attacks in distruption-tolerant networks using encounter tickets. In *Proc. of INFOCOM'09*, pages 2428–2436, 2009.

[41] N. Li and S. Das. A trust-based framework for data forwarding in opportunistic networks. *Ad Hoc Networks Journal*, 2011.

[42] N. Li and S. K. Das. Radon: Reputation-assisted data forwarding in opportunistic network. In *Mobiopp'10*, pages 8–14, Feb 2010.

[43] N. Li, N. Zhang, and S. Das. Relationship privacy preservation in publishing online social networks. In *SocialCom'11*, October 2011.

[44] N. Li, N. Zhang, and S. K. Das. Preserving relation privacy in online social network data. *Internet Computing, IEEE*, 15(3):35–42, May-June 2011.

[45] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computing Comm. Rev.*, 7(3):19–20, 2003.

[46] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD'08*.

[47] L. Liu, J. Wang, J. Liu, and J. Zhang. Privacy preserving in social networks against sensitive edge disclosure. In *Technical Report CMIDA-HiPSCCS 006-08, Department of Computer Science, University of Kentucky, KY*, 2008.

[48] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1), March 2007.

[49] S. Milgram. The small world problem. *Psychology Today*, 1(61):60–67, 1967.

[50] A. Mtibaa, M. May, C. Diot, and M. Ammar. Peoplerank: Social opportunistic forwarding. In *Infocom*, pages 1–5, March 2010.

[51] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Oakland'09*.

[52] J. Pujol, A. Toledo, and P. Rodriguez. Fair routing in delay tolerant networks. In *INFOCOM 2009, IEEE*, pages 837 –845, April 2009.

[53] M. Radenkovic and A. Grundy. Congestion aware forwarding in delay tolerant and social opportunistic networks. In *Eighth International Conference on Wireless On-demand Network Systems and Services (WONS), 2011*, Jan. 2011.

[54] Y. Ren, M. C. Chuah, J. Yang, and Y. Chen. Detecting blackhole attacks in disruption-tolerant networks through packet exchange recording. In *Proc. of 1st Workshop D-SPAN (colocated with WoWMoM'10)*, 2010.

[55] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of ISWC*, 2003.

[56] M. Seligman, K. Fall, and P. Mundur. Storage routing for dtn congestion control. In *Wireless Communications and Mobile Computing*, pages 1183–1196, May 2007.

[57] G. Shafer. *A mathematical theory of evidence.* Princeton University Press, Princeton, NJ, 1976.

[58] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of ACM SIGKDD*, 2010.

[59] Y. L. Sun, Z. Han, W. Yu, and K. J. R. Liu. A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. In *Proc. of INFOCOM'06*, pages 1–13, 2006.

[60] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of ACM SIGKDD*, pages 404–413, 2006.

[61] J. Travers and S. Milgram. An experimental study of the samll world problem. *Sociometry*, 32(425):425–443, 1969.

[62] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. In *Duke University Tech. Rep. CS-2000-06*, 2000.

[63] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of WOSN*, August 2009.

[64] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *Proceedings of EuroSys*, 2009.

[65] B. Wu, J. Chen, J. Wu, and M. Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security, Y. Xiao, X. Shen, and D. Du , Springer, Network Theory and Applications*, 2006.

[66] S. Ye and F. Wu. Estimating the size of online social networks. In *Proceedings of IEEE SocialCom*, 2010.

[67] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 3–17, 2008.

[68] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, 2006.

[69] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. Dsybil: Optimal sybil-resistance for recommendation systems. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 283–298, 2009.

[70] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD'07*.

[71] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE'08*.

[72] L. Zou, L. Chen, and M. T. Ozsu. K-automorphism: A general framework for privacy preserving network publication. In *VLDB'09*.

BIOGRAPHICAL STATEMENT

Na Li received her B.S. degree in Computer Science and Technology from NanKai University, Tianjin, China in 2005. Prior to joining UTA, she was a research assistant in Computer Network Information Center, Chinese Academy of Sciences, Beijing, China from 2005-2007. She started her Ph.D. program at Computer Science and Engineering Department in the University of Texas at Arlington since 2007 fall. In the meantime, she is also a research assistant in the Center for Research in Wireless Mobility and Networking (CReWMaN) at UTA. Her current research interests include privacy and security issues in challenging networks, such as Wireless Sensor Networks (WSNs), Opportunistic Networks (OppNets) and Online Social Networks(OSNs), in particular preserving location privacy of data source in WSNs, defending against selfish/malicious data forwarding in OppNets, and protecting link privacy in OSNs. Besides, she is also working on community detection on OSNs.