PRIVACY AND SPECTRAL ANALYSIS OF SOCIAL NETWORK RANDOMIZATION

by

Xiaowei Ying

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Information Technology

Charlotte

2011

Approved by:

_____
Dr. Xintao Wu

_____
Dr. Zbigniew W. Ras

_____
Dr. Aidong Lu

_____
Dr. Jiancheng Jiang

_____
Dr. Xinde Dai

ABSTRACT

XIAOWEI YING.Privacy and spectral analysis of social network randomization.
(Under the direction of DR. XINTAO WU)

Social networks are of significant importance in various application domains. Understanding the general properties of real social networks has gained much attention due to the proliferation of networked data. Many applications of networks such as anonymous web browsing and data publishing require relationship anonymity due to the sensitive, stigmatizing, or confidential nature of the relationship. One general approach for this problem is to randomize the edges in true networks, and only release the randomized networks for data analysis. Our research focuses on the development of randomization techniques such that the released networks can preserve data utility while preserving data privacy.

Data privacy refers to the sensitive information in the network data. The released network data after a simple randomization could incur various disclosures including identity disclosure, link disclosure and attribute disclosure. Data utility refers to the information, features, and patterns contained in the network data. Many important features may not be preserved in the released network data after a simple randomization. In this dissertation, we develop advanced randomization techniques to better preserve data utility of the network data while still preserving data privacy. Specifically we develop two advanced randomization strategies that can preserve the spectral properties of the network or can preserve the real features (e.g., modularity) of the network. We quantify to what extent various randomization techniques can protect data privacy when attackers use different attacks or have different background knowledge. To measure the data utility, we also develop a consistent spectral framework to measure the non-randomness (importance) of the edges, nodes, and the overall graph. Exploiting the spectral space of network topology, we further develop fraud detection

techniques for various collaborative attacks in social networks. Extensive theoretical analysis and empirical evaluations are conducted to demonstrate the efficacy of our developed techniques.

ACKNOWLEDGMENTS

In my way of pursuing this Ph.D. degree in the past years, many people contributed a lot, in many different ways, to make me successfully finish the research work. I would like to express my sincere gratitude to all of them.

First of all, I should thank my dissertation committee members, Dr. Xintao Wu, Dr. Zbigniew W. Ras, Dr. Aidong Lu, Dr. Jiancheng Jiang, and Dr. Xinde Dai. I greatly appreciate their time, effort and advices on my research and dissertation.

I also would like to express my thanks to all the co-authors for collaborating with me to develop some research ideas presented in this dissertation, especially Leting Wu, Kai Pan, Dr. Ling Guo, Dr. Aidong Lu, Dr. Weichao Wang, Lane T. Harrison, and Xianlin Hu from University of North Carolina at Charlotte, Dr. Daniel Barbará from George Mason University, Dr. Lei Chen from Hong Kong University of Science and Technology, Dr. Kun Liu from Yahoo! Lab. and Dr. Zhi-Hua Zhou from Nanjing University. I am fortunate to have the opportunity to work with my colleagues at Data Privacy Lab., especially Dr. Yong Ye, Dr. Songtao Guo, and Jun Zhu. I appreciate all their friendships and encouragement to finish this dissertation.

I would like to greatly thank my advisor, Dr. Xintao Wu, who always guides me, supports me, encourages me, and even spoils me along my way to the Ph.D. degree. He has made the years of Ph.D. study one of the most cherishable periods in my life. This dissertation was part of the research carried out through his vision throughout last five years, and was supported by U.S. National Science Foundation IIS-0546027 and CNS-0831204.

Finally, it is impossible for me to finish this dissertation without the love and support from my parents, Yongmao Ying, and Wuping Dong. This dissertation is dedicated to them.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Social networks are of significant importance in various application domains such as marketing, psychology, epidemiology and homeland security. The management and analysis of these networks have attracted increasing interests in the sociology, database, data mining and theory communities. Most previous studies are focused on revealing interesting properties of networks and discovering efficient and effective analysis methods [7, 9, 10, 33, 36, 54, 56, 58, 60, 84, 87, 89, 91, 98].

Social networks often contain some private attribute information about individuals as well as their sensitive relationships. Many applications of social networks such as anonymous Web browsing require identity and/or relationship anonymity due to the sensitive, stigmatizing, or confidential nature of user identities and their behaviors. The privacy concerns associated with data analysis over social networks have incurred the recent research. In particular, privacy disclosure risks arise when the data owner wants to publish or share the social network data with the third party for research or business-related applications. Privacy-preserving techniques [3] for social network publishing aim to protect privacy through masking, modifying and/or generalizing the original data while without sacrificing much data utility.

A network $G(V, E)$ is a set of n nodes connected by a set of m links, where V denotes the set of nodes and $E \subseteq V \times V$ is the set of links. In our work, we mainly focus on undirected, and un-weighted graphs without self-loops. Let $A = (a_{ij})_{n \times n}$ denote the adjacency matrix of G: $a_{ij} = 1$ if node i and j are connected and $a_{ij} = 0$ otherwise. The degree of node i, d_i, is the number of the nodes connected to node i, i.e., $d_i = \sum_j a_{ij}$, and $\boldsymbol{d} = \{d_1, \ldots, d_n\}$ denotes the degree sequence. The released

graph after perturbation is denoted by $\widetilde{G}(\widetilde{V}, \widetilde{E})$. $\widetilde{A} = (\tilde{a}_{ij})_{n\times n}$ is the adjacency matrix of $\tilde{G}$, and $\tilde{d}_i$ and $\tilde{\boldsymbol{d}}$ are the degree and degree sequence of $\widetilde{G}$ respectively. Note that, for ease of presentation, we use the following pairs of terms interchangeably: "graph" and "network", "node" and "vertex", "edge" and "link", "entity" and "individual", "attacker" and "adversary".

1.1 Privacy in Publishing Social Networks

In a social network, nodes usually correspond to individuals or other social entities, and an edge corresponds to the relationship between two entities. Each entity can have a number of attributes, such as age, gender, income, and a unique identifier. One common practice to protect privacy is to publish a naive node-anonymized version of the network, e.g., by replacing the identifying information of the nodes with random IDs. While the naive node-anonymized network permits useful analysis, as first pointed out in [8, 46], this simple technique does not guarantee privacy since adversaries may re-identify a target individual from the anonymized graph by exploiting some known structural information of his neighborhood.

The privacy breaches in social networks can be grouped into three categories: *identity disclosure*, *link disclosure*, and *attribute disclosure*. The identity disclosure corresponds to the scenario where the identity of an individual who is associated with a node is revealed. The link disclosure corresponds to the scenario where the sensitive relationship between two individuals is disclosed. The attribute disclosure denotes the sensitive data associated with each node is compromised. Compared with existing anonymization and perturbation techniques of tabular data [40, 41, 51], it is more challenging to design effective anonymization techniques for social network data because of difficulties in modeling background knowledge and quantifying information loss.

Adversaries usually rely on background knowledge to de-anonymize nodes and learn the link relations between de-anonymized individuals from the released anonymized

graph. The assumptions of the adversary's background knowledge play a critical role in modeling privacy attacks and developing methods to protect privacy in social network data. In the following, we briefly introduce some background knowledge and how the adversary can utilize them to breach the privacy of the network. However, we should point out that it is very challenging to model all types of background knowledge of adversaries and quantify their impacts on privacy breaches in the scenario of publishing social networks with privacy preservation [64, 114].

One type of background knowledge is fraudulent members. The adversary can himself join the network or bribe some individual in the network. The created fraudulent nodes in the network can be utilized to compromise the privacy information. For example, in the *active attack* [8], an adversary creates a small number of new user accounts with links to the targeted individuals and establishes a highly distinguishable pattern of links among the new accounts. Once the anonymized graph is released, the adversary can then efficiently find these new accounts together with the target individuals in the released anonymized network.

Another type of background knowledge is the neighborhood of the targeted individual. The adversaries are assumed to possess some knowledge on the neighborhood of the target, such as degree, the topological structure of the neighbor, and some statistics about the neighborhood (e.g., the number of triangles, centrality value etc.) [46, 47, 63, 113]. Different from *active attack* that can target arbitrary individuals in the network, the background knowledge of neighborhood is more detrimental to those nodes playing central role in the social network. This is because the central nodes usually have higher degrees than most of the other nodes, and hence their neighborhood structures are very likely to be unique in the network. Once the adversary knows such knowledge, he has a high probability to identify the neighborhood in the anonymized graph and identify the targeted individual.

1.2 Randomization as a Privacy Preservation Approach

One of the reasons that naive anonymization can not prevent privacy breaches is that adversary may have various background knowledge about the targets and the network. With the background knowledge, the adversaries may be able to uniquely link a node, an edge, or a subgraph to the targeted individuals, and the privacy is then jeopardized. Even when the background knowledge does not lead to the unique identification of the targeted individual, it can still significantly reduce the number of candidates and hence increase the adversary's confidence. Therefore, many privacy preservation approaches aim to modify the released graph so that there should be multiple nodes, link, or subgraphs that match the background knowledge of the targeted individual. Currently there are mainly three categories of privacy preservation approaches:

- K-anonymity: this approach modifies graph structure via a sequence of edge deletions and additions such that each node in the modified graph is indistinguishable with at least $K-1$ other nodes in terms of some types of structural patterns such as degree and neighborhood subgraph.

- Edge randomization: This approach modifies graph structure by randomly adding and(or) deleting edges or switching edges. It protects against re-identification in a probabilistic manner.

- Clustering-based generalization: This approach clusters nodes and edges into groups and anonymizes a subgraph into a super-node. The details about individuals are hidden.

Our work mainly focus on the randomization approach. Randomization approaches have been well investigated in privacy-preserving data mining for numerical data (e.g., [3, 4, 41, 51]) and categorical data (e.g., [39, 40, 93]). For social networks, two edge-based randomization strategies have been commonly adopted:

- *Rand Add/Del*: randomly add k false edges followed by deleting k true edges. This strategy preserves the total number of edges in the original graph.

- *Rand Switch*: randomly switch a pair of existing edges (t, w) and (u, v) (satisfying edge (t, v) and edge (u, w) do not exist in G) to (t, v) and (u, w), and repeat this process for k times. This strategy preserves the degree of each vertex.

After randomization, the randomized graph is expected to be different from the original one. The adversary is generally not able to re-identify the correct individual if he simply finds the nodes, links, or subgraphs that match the background knowledge. As a result, the node identities as well as the true sensitive relationship between two nodes are protected. We will discuss why randomized graphs are resilient to structural attacks in Chapter 3.

Link Privacy Disclosure Risks. The randomization approaches protect against re-identification in a probabilistic manner. Therefore, we quantify the privacy disclosure risk by the probability that the adversary estimates the true data correctly. The process of randomization and the randomization parameter k are assumed to be published along with the released graph.

We first quantify to what extent the randomization approaches can protect sensitive links in Chapter 3. There exist some scenarios that node identities (and even entity attributes) are not confidential but sensitive links between target individuals are confidential and should be protected. For example, in a transaction network, an edge denoting a financial transaction between two individuals is considered confidential while nodes corresponding to individual accounts is non-confidential. In such cases, data owners can release the edge randomized graph without removing node annotations. To extent to which releasing a randomized graph $\tilde{G}$ jeopardizes the link privacy is measured by the adversary's posterior belief about the existence of edge (i, j) given randomized graph $\tilde{G}$. We will investigate two types of posterior belief in Chapter 3: the posterior probability in which the adversary only utilizes the existence

(or non-existence) of a link in $\tilde{G}$, and the enhanced posterior probability derived by exploiting the proximity between two nodes.

With the released graph $\tilde{G}$, the adversary can first rely on the observation of existing (or non-existing) link (i, j) in $\tilde{G}$ to breach the true sensitive link between node i and j. Take *Rand Add/Del* as an example. The adversary's prior belief about the existence of edge (i, j) (without exploiting the released graph) can be calculated as $\Pr(a_{ij} = 1) = \frac{2m}{n(n-1)}$. With the released graph and perturbation parameter k, the posterior belief when observing $\tilde{a}_{ij} = 1$ is $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1) = \frac{m-k}{m}$. If $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1)$ is significantly higher than $\Pr(a_{ij} = 1)$, releasing the randomized graph can lead to high privacy disclosure risk.

However, the above effort to measure link disclosure risks of randomization mainly relies on the randomization magnitude. We further propose an attacking model that exploits the relationship between the existence of a link and the similarity of the node pairs in the released randomized graph. Proximity measures have been shown to be effective in the classic link prediction problem [62] (i.e., predicting the future existence of links among nodes given a snapshot of a current graph). We investigate four proximity measures (common neighbors, Katz measure, Adamic/Adar measure, and commute time) and quantify how much the posterior belief on the existence of a link can be enhanced by exploiting those similarity values derived from the released graph which is randomized by the *Rand Add/Del* strategy. One difference between the proximity based link prediction and the proximity based privacy breach is that, the data miner derives the similarity of nodes based the true (probably incomplete) graph, whereas the adversary derives the measure from the randomized graph.

Identity Privacy Disclosure Risks. In Chapter 4, we study the risk of identity disclosure associated with randomization procedures. Since the goal of an adversary is to map the nodes/edges in the released graph to real world entities/relationships, we investigate the relationship between the amount of randomization and the ad-

versarys ability to correctly infer the node identity, and quantify both identity and link disclosure risks when adversaries know the degrees of target individuals. When the graph is published using naive anonymization, once the identities of the targeted nodes are compromised, the link privacy is also breached. This is because all the links in the released graph are true links. However, this is not true if we release the randomized graph, as the observed link can actually be a fake link. When both the nodes' identities and link existence are uncertain to the adversaries, releasing the randomized graph can greatly reduce the privacy disclosure risks.

1.3 Data Utility and Feature Preserving Randomization

Data Utility. An important goal of publishing social network data is to permit useful analysis tasks. In general, it is very challenging to quantify the information loss of anonymizing social networks, because different analysis tasks may expect different utility properties to be preserved. In our work, we consider two types of measures to evaluate the data utility of social network data.

- Graph topological properties: one of the most important applications of social network data is for analyzing graph properties. To understand and utilize the information in a network, researches have developed various measures to indicate the structure and characteristics of the network from different perspectives. [23]. Properties including degree sequences, shortest connecting paths, and clustering coefficients are addressed in many works [46, 47, 63, 104, 106, 113].

- Graph spectral properties: the spectrum of a graph is defined as the set of eigenvalues of the graph's adjacency matrix or other derived matrices. The eigenvalues and eigenvectors decode various properties of the graph as well as the edges and nodes in the graph. The graph spectrum has close relations with many graph characteristics and can provide global measures for some network properties [84].

- Aggregate network queries: an aggregate network query calculates the aggregate on some paths or subgraphs satisfying some query conditions. One example is that the average distance from a medical doctor vertex to a teacher vertex in a network. Some researchers considered the accuracy of answering aggregate network queries as the measure of utility preservation [11, 22, 113, 115].

Edge randomization may significantly affect the utility of the released randomized graph. To preserve utility, certain aggregate characteristics (a.k.a., feature) of the original graph should remain basically unchanged or at least some properties can be reconstructed from the randomized graph. However, as what we will show later, many topological features are lost due to randomization. In Chapter 5, we propose two randomization procedures that can preserve structural properties.

Spectrum Preserving Randomization. Since the spectra of graph matrices have close relations with many important topological properties such as diameter, presence of cohesive clusters, long paths and bottlenecks, and randomness of the graph [84], we aim to preserve the data utility by preserving two important eigenvalues during the randomization: the largest eigenvalue of the adjacency matrix and the second smallest eigenvalue of the Laplacian matrix. Pure randomization tends to move the eigenvalues toward one direction, and the eigenvalues of the randomized graph can be significantly different from the original values. The two proposed algorithms, *Spctr Add/Del* and *Spctr Switch*, selectively pick up those edges that can increase (or decrease) the target eigenvalue by examining the eigenvector values of the nodes involved in the randomization. By doing so, they guarantee that the eigenvalues after randomization do not move far from the original value. Our empirical evaluations showed that the proposed algorithms can keep the spectral features as well as many topological features close to the original ones even when the magnitude of randomization is large.

Markov Chain Based Feature Preserving Randomization. The degree sequence and topological features are of great importance to the graph structure. One natural idea is that it can better preserve the data utility if the released graph $\widetilde{G}$ preserves the original degree sequence and a certain topological feature, such as transitivity or average shortest distance [44, 104]. To preserve data utility, data owners may want to preserve some particular feature $\boldsymbol{S}$ within a precise range in the released graph. All the graphs that satisfy the degree sequence $\boldsymbol{d}$ and the feature constraint $\boldsymbol{S}$ form a graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ (or $\mathcal{G}_{\boldsymbol{d}}$ if no feature constraint). Starting with the original graph, series of switches form a Markov chain that can explore the graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$. We propose an algorithm that can generate any graph in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ with equal probability. The constraint $\boldsymbol{S}$ guarantees the utility of the randomized graph, and the equal probability for all the graphs in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ aims to reduce the privacy disclosure risk.

Privacy Analysis for Feature Preserving Randomization. We also study the link disclosure risks for feature preserving randomization procedures in Chapter 5. Note that the adversaries can exploit the released graph as well as feature constraints to breach link privacy. The feature constraint may reduce the graph space and increase the risk of privacy disclosure. We study the attacking model in which the adversary is able to calculate the posterior probability of existence of a certain link by exploiting the graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$. If many graphs in the graph space have link (i, j), the original graph is also very likely to have link (i, j), and hence the adversary's posterior belief about link (i, j) is given by

$$\Pr[G(i,j) = 1 | \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}] = \frac{1}{|\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}|} \sum_{G_t \in \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}} G_t(i,j).$$

Knowing the degree sequence $\boldsymbol{d}$ and the feature constraint $\boldsymbol{S}$, the adversary can generate and exploit the graph space via the Markov chain that starts with the released graph $\widetilde{G}$. The adversary can take the node pairs with highest posterior beliefs as candidate links. This attacking model works because the convergence of

the Markov chain does not depend on the initial point. Our evaluations showed that some feature constraints can significantly enhance the adversary's attacking accuracy and the extent to which a feature constraint jeopardizes link privacy varies for different graphs.

1.4 Spectral Analysis on Social Network Randomization

Social networks tend to contain some amount of randomness and some amount of non-randomness. Consider an online social network where each node denotes an individual and an edge between two nodes denotes a social interaction between the two individuals. An individual's social network tends to consist of members of the same ethnic group, race, or social class. Intuitively, two friends of a given individual are more likely to be friends with each other than they are with other randomly chosen members. The edge connecting one individual's two friends contains less randomness. However, an individual also tends to have some number of random friends from other groups and those edges between this individual and his random friends contain more randomness. The amount of randomness versus non-randomness at node/edge levels can clearly affect various properties of a social network.

As we discussed earlier, there are numerous features, measure and statistics that characterize the graph from various perspectives. It is tedious, if not impossible, to consider all the features in analyzing the social network. Among the three types data utility measures mentioned in Section 1.3, we mainly focus on the graph spectral properties in our work. It has been show that many topological features have an close relationship with the graph spectrum [84]. This is also corroborated by our spectrum preserving randomization procedures. The spectrum preserving randomization procedures aim to preserve data utility via preserving some certain graph eigenvalues. Our empirical studies show that when the spectral features are preserved during the edge randomization, many topological features are preserved as well. In spectral analysis of social network randomization, we mainly focus on two closely related questions: how

the graph spectra reflect the difference between a real-world graph (or a randomized one) and the random one, and how to quantify the difference.

A Framework of Non-randomness Measures. In Chapter 6, we present a framework which provides a series of non-randomness measures at all granularity levels, from edge, node, subgraph to the whole graph. Non-randomness specified at the edge level can help users quantify how different a given interaction is from random ones. Similarly, non-randomness at the node level can help users quantify how different a given individual is from random nodes (those individuals actually not belonging to this social network). In our framework, we first examine how much non-randomness a given edge (social interaction) has, then measure a node's non-randomness by examining the non-randomness values of edges connecting to this node. Finally, we derive the non-randomness measure of the entire graph (subgraph) by incorporating the non-randomness values of all edges within the graph (subgraph).

Our framework of non-randomness measure is based on our finding that the real-world graphs exhibit the clear line orthogonality patterns in the adjacency spectral space, and the patterns are closely related to their topological structure. We show that graph with k clear communities displays k quasi-orthogonal lines in the space spanned by the leading k eigenvectors of the adjacency matrix, and nodes from the same community all lie on or around one line starting from the the origin with central nodes far away from the origin and noisy nodes close to the origin. We further explain why community structure results to such pattern in the adjacency spectral space, and why it is different from the pattern in the normal or Laplacian spectral space. We show that graph non-randomness can be obtained mathematically from the spectra of the adjacency matrix of the network. Both theoretical and empirical studies in spectral geometries of social networks show that our proposed non-randomness measures well characterize and capture graph randomness.

Applications. One application of the graph non-randomness framework is the com-

munity partition. Utilizing the spectral patterns corresponding to the community structure in the adjacency spectral space, we develop a graph partition algorithm *AdjCut* in Chapter 6. Our *AdjCut* algorithm is different from the normal or Laplacian spectrum based community partition algorithms in two aspects. First, our *AdjCut* algorithm partitions the graph by fitting the k-orthogonal lines in the adjacency spectral space, while the normal or Laplacian based algorithms find the cluster in their spectral spaces. Second, our *AdjCut* algorithm tends to reduce the edges non-randomness among the graph communities, whereas the normal or Laplacian based algorithms simply minimize the number of cuts among the communities.

In Chapter 7, we investigate another application of the graph non-randomness measures: the fraud detection in social network settings. One merit of the graph non-randomness framework is that it not only captures the randomness added by the randomization process, but also more general randomness due to various types of noise. For example, in the *active attack*, the adversaries join the network and introduce fraudulent nodes and links to the original graph. The subgraph created by the adversaries is expected to have patterns different from the original graph, thus introducing randomness to the original graph to some extent. Therefore, the non-randomness framework can be used to detect frauds in the social network. Based on the non-randomness framework, we propose a general framework for detecting attacks in social networks. Particularly, we focus on the *random link attack* (RLA) in which the adversaries join the network, form some subgraph among themselves, and send links to a randomly selected legitimate users. Many attacks in social networks can be modeled (or partly modeled) as RLA, such as the bipartite core attack [19, 77] and the distributed denial of service attack. We show that node non-randomness values of the fraudulent nodes are significantly lower than normal nodes regardless how the adversaries create links among themselves. Therefore, by identifying the nodes with significantly low non-randomness measure, we can capture a large proportion of

fraudulent nodes with few false positives.

1.5 Graph Features and Data Sets

In this section, we summarize some graph features and data sets that are commonly used all over the work.

Features. The *harmonic mean of the shortest distance* h is defined in [61] as:

$$h = \left\{ \frac{1}{n(n-1)} \sum_{i \neq j} \frac{1}{d_{ij}} \right\}^{-1} \tag{1.1}$$

The inverse of the harmonic mean of the shortest distance, also known as the global efficiency, varies between 0 and 1, with $h^{-1} = 0$ when all vertices are isolated and $h^{-1} = 1$ when the graph is complete.

The *modularity* Q indicates the goodness of the community structure [23]. It is defined as the fraction of all edges that lie within communities minus the expected value of the same quantity in a graph in which the vertices have the same degrees but edges are placed at random without regard for the communities. A value $Q = 0$ indicates that the community structure is no stronger than would be expected by random chance and values other than zero represent deviations from randomness. Formally, for a graph with g groups, the modularity is defined to be $Q = \sum_i e_{ij} - \sum_{ijk} e_{ij} e_{ki} = Tr(e) - \|e^2\|$ where $\|e\|$ indicates the sum of all elements of e and $Tr(e)$ denotes the trace of a matrix (i.e., the sum of the entries on the diagonal). The element e_{ij} is the fraction of edges in the original graph that connect vertices in group i to those in group j. The modularity measure Q indicates the goodness of the community structure, and the real-world unweighted networks with high community structure generally have Q values within a range from 0.3 to 0.7 [73].

$$Q = \sum_i [e_{ii} - (\sum_j e_{ij})^2], \tag{1.2}$$

where e_{ij} is the proportion of edges between community i and j. Since the modularity

measure is defined on a given partition, its change indicates how the quality of the original partition changes along the perturbation.

The *transitivity* C is one type of clustering coefficient measure and characterizes the presence of local loops near a vertex. It is formally defined as

$$C = \frac{3N_\Delta}{N_3} \tag{1.3}$$

where N_Δ is the number of triangles and N_3 is the number of connected triples.

The *subgraph centrality* SC is used to quantify the centrality of vertex i based the subgraphs [31].

$$SC = \frac{1}{n}\sum_{i=1}^{n} SC_i = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{\infty}\frac{P_i^k}{k!} \tag{1.4}$$

where P_i^k is the number of paths that start with i and end in i with length of k.

Data Sets. We give some basic information of the data sets that are commonly used in our work, and some basic statistics are summarized in Table 1.1.

Table 1.1: Statistics summary of datasets

Network	n	m	Partition Labels
karate	34	78	None
dolphins	62	159	None
polbooks	105	441	2 partitions
Enron	151	869	None
E-mail	1133	5451	None
polblogs	1222	16714	2 partitions
netsci	1589	2742	None
Facebook	63731	817090	None

polbooks: US politics book data [59] contains 105 vertices and 441 edges. In this graph, nodes represent books about US politics sold by the online bookseller Amazon.com while edges represent frequent co-purchasing of books by the same buyers on Amazon. Nodes are separated into groups according to their political views: "liberal", "neutral", or "conservative". These alignments were assigned separately by Mark Newman based on a reading of the descriptions and reviews of the books posted on Amazon.

polblogs: political blogosphere data set [1] compiles the data on the links among US political blogs, containing over 1,000 vertices and 15,000 edges. The blogs were labeled as either liberal or conservative, based on incoming and outgoing links and posts around the time of the 2004 presidential election. The original data is a directed graph. Here we simply consider $a_{ij} = 1$ if the two blogs have a link between them.

Enron: the Enron email network was built from email corpus of a real organization over the course covering a 3 years period. We used a pre-processed version of the dataset provided by [85]. This dataset contains 252,759 emails from 151 Enron employees, mainly senior managers. We regard there is an edge between node i and j if there is at least 5 emails between them if not otherwise noted.

dolphins: the dolphins data set contains an undirected social network of 159 frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand [67].

karate: the karate date set contains the network of 78 pairs of friendships between the 34 members of a karate club at a US university, as described by Wayne Zachary in 1977 [111].

E-mail: The *E-mail* graph is the network of e-mail interchanges between members of the Univeristy Rovira i Virgili (Tarragona) (`http://deim.urv.cat/~aarenas/data/welcome.htm`).

netsci: the net science data set contains a coauthorship network of scientists working on network theory and experiment, as compiled by M. Newman in May 2006 [74].

Facebook: the date set contains a subset of all of the user-to-user links from the Facebook New Orleans networks [95]. The crawler started from a single user and visited all friends of the user and their friends in a breadth-first-search fashion. The crawler could only view users who made their profiles visible to the network. This is an undirected graph though the crawler treated it like a directed graph. Since friend

link on facebook is created with the confirmation on both side, we convert the result of crawler to an undirected graph. It contains 63731 users and 817090 links. A link between two users means they appear on each other's friend list.

CHAPTER 2: RELATED WORKS

In this chapter, we briefly review some recently proposed privacy-preserving techniques for publishing social network data. It is difficult to compare and/or categorize current techniques systematically. However, we would like to point out that privacy-preserving techniques are different from each other mainly in the following aspects: (1) assumptions of attacking models and background knowledge of the adversaries; (2) network data settings (simple or rich); (3) disclosure risks (identity disclosure, link disclosure, attribute disclosure.); (4) privacy-preserving approaches (K-anonymity, generalization, or randomization); and (5) utility preservation targets (graph features, accuracy of structural queries, etc.). Before presenting privacy-preserving approaches, we first review some attacks that can incur privacy breaches in social networks.

2.1 Privacy Attacks on Naive Anonymized Networks

The practice of naive anonymization replaces the personally identifying information associated with each node with a random ID. However, an adversary can potentially combine external knowledge with the observed graph structure to compromise privacy, de-anonymize nodes, and learn the existence of sensitive relationships between explicitly de-anonymized individuals.

Active Attacks and Passive Attacks. Backstrom et al. presented two different types of attacks on anonymized social networks [8].

- Active attacks: an adversary chooses an arbitrary set of target individuals, creates a small number of new user accounts with edges to these target individuals, and establishes a highly distinguishable pattern of links among the new accounts. The adversary can then efficiently find these new accounts together

with the target individuals in the released anonymized network.

- Passive attacks: an adversary does not create any new nodes or edges. Instead, he simply constructs a coalition, tries to identify the subgraph of this coalition in the released network, and compromises the privacy of neighboring nodes as well as edges among them.

The *active attack* is based on the uniqueness of small subgraphs embedded in the network. The constructed subgraph H by the adversary needs to satisfy the following three properties in order to make the *active attack* succeed:

- There is no other subgraph S in G such that S and H are isomorphic.
- H is uniquely and efficiently identifiable regardless of G.
- The subgraph H has no non-trivial automorphisms.

Backstrom et al. showed that a randomly generated subgraph H formed by $O(\sqrt{\log n})$ nodes can compromise the privacy of arbitrarily target nodes with high probability for any network. The *passive attack* is based on the observation that most nodes in real social network data already belong to a small uniquely identifiable subgraph. A coalition X of size k is initiated by one adversary who recruits $k-1$ of his neighbors to join the coalition. It assumes that the users in the coalition know both the edges amongst themselves (i.e., the internal structure of H) and the names of their neighbors outside X. Since the structure of H is not randomly generated, there is no guarantee that it can be uniquely identified. The primary disadvantage of the *passive attack* in practice, compared to the *active attack*, is that it does not allow one to compromise the privacy of arbitrary users. The adversaries can adopt a hybrid *semi-passive* attack: they create no new accounts, but simply create a few additional out-links to target users before the anonymized network is released. We refer readers

to [56] for more details on theoretical results and empirical evaluations on a real social network with 4.4 million nodes and 77 million edges extracted from LiveJoural.com.

Structural Queries. Hay et al. studied three types of background knowledge to be used by adversaries to attack naively-anonymized networks [47]. They modeled adversaries' external information as the access to a source that provides answers to a *restricted knowledge query* Q about a single target node in the original graph. Specifically, background knowledge of adversaries is modeled using the following three types of queries.

- Vertex refinement queries: these queries describe the local structure of the graph around a node in an iterative refinement way. The weakest knowledge query, $\mathcal{H}_0(x)$, simply returns the label of the node x; $\mathcal{H}_1(x)$ returns the degree of x; $\mathcal{H}_2(x)$ returns the multiset of each neighbors' degree, and $\mathcal{H}_i(x)$ can be recursively defined as:

$$\mathcal{H}_i(x) = \{\mathcal{H}_{i-1}(z_1), \mathcal{H}_{i-1}(z_2), \cdots, \mathcal{H}_{i-1}(z_{d_x})\}$$

  where $z_1, \cdots, z_{d_x}$ are the nodes adjacent to x.

- Subgraph queries: these queries can assert the existence of a subgraph around the target node. The descriptive power of a query is measured by counting the number of edges in the described subgraph. The adversary is capable of gathering some fixed number of edges focused around the target x. By exploring the neighborhood of x, the adversary learns the existence of a subgraph around x representing partial information about the structure around x.

- Hub fingerprint queries: a hub is a node in a network with high degree and high betweenness centrality. A hub fingerprint for a target node x, $\mathcal{F}_i(x)$, is a description of the node's connections to a set of designated hubs in the network where the subscript i places a limit on the maximum distance of observable hub

connections.

The above queries represent a range of structural information that may be available to adversaries, including complete and partial descriptions of node's local neighborhoods, and node's connections to hubs in the network.

Vertex refinement queries provide complete information about node degree while a subgraph query can never express $\mathcal{H}_i$ knowledge because subgraph queries are existential and cannot assert exact degree constraints or the absence of edges in a graph. The semantics of subgraph queries seem to model realistic adversary capabilities more accurately. It is usually difficult for an adversary to acquire the complete detailed structural description of higher-order vertex refinement queries.

Other Attacks. Narayanan and Shmatikov assumed that the adversary has two types of background knowledge: aggregate auxiliary information and individual auxiliary information [71]. The aggregate auxiliary information includes an auxiliary graph $G_{\mathsf{aux}}(V_{\mathsf{aux}}, E_{\mathsf{aux}})$ whose members overlap with the anonymized target graph and a set of probability distributions defined on attributes of nodes and edges. These distributions represent the adversary's (imperfect) knowledge of the corresponding attribute values. The individual auxiliary information is the detailed information about a very small number of individuals (called *seeds*) in both the auxiliary graph and the target graph.

After re-identifying the seeds in target graph, the adversaries immediately get a set of de-anonymized nodes. Then, by comparing the neighborhoods of the de-anonymized nodes in the target graph with the auxiliary graph, the adversary can gradually enlarge the set of de-anonymized nodes. During this *propagation* process, known information such as probability distributions and mappings are updated repeatedly to reduce the error. The authors showed that even some edge addition and deletion are applied independently to the released graph and the auxiliary graph, their de-anonymizing algorithm can correctly re-identify a large number of nodes in

the released graph.

2.2 K-anonymity Privacy Preservation via Edge Modification

The adversary aims to locate the vertex in the network that corresponds to the target individual by analyzing topological features of the vertex based on his background knowledge about the individual. Whether individuals can be re-identified depends on the descriptive power of the adversary's background knowledge and the structural similarity of nodes. To quantify the privacy breach, Hey et al. [47] proposed a general model for social networks as follows:

Definition 2.1: K-candidate anonymity. A node i is K-candidate anonymous with respect to a structure query Q if $|cand_Q(i)| \geq K$ where $cand_Q(i) = \{j \in V | Q(j) = Q(i)\}$. A graph satisfies K-candidate anonymity with respect to Q if all the nodes are K-candidate anonymous with respect to Q.

In other words, there exist at least $K-1$ other nodes in the graph that match query $Q(i)$, and the nodes in $cand_Q(i)$ are indistinguishable with respect to query Q. Then, releasing a K-candidate anonymous graph has less risk to breach the privacy.

Three types of queries (vertex refinement queries, subgraph queries, and hub fingerprint queries) were presented and evaluated on the naive anonymized graphs. Hay et al. presented a generalization technique that groups nodes into super-nodes and edges into super-edges to satisfy the K-anonymity [47].

Several methods have been investigated to prevent node re-identification based on the K-anonymity concept. These methods differ in the types of the structural background knowledge that an adversary may use.

K-degree Generalization . Liu and Terzi pointed out that the degree sequences of real-world graphs are highly skewed, and it is usually easy for adversaries to collect the degree information of a target individual [63]. They investigated how to modify a graph via a set of edge addition (and/or deletion) operations in order to construct

a new K-degree anonymous graph, in which every node has the same degree with at least $K-1$ other nodes. The K-degree anonymity property prevents the re-identification of individuals by the adversaries with prior knowledge on the number of social relationships of certain people (i.e., vertex background knowledge). The authors imposed a requirement that the minimum number of edge-modifications is made in order to preserve the utility.

Problem 2.1: Given a graph $G(V, E)$, construct a new graph $\widetilde{G}(\widetilde{V}, \widetilde{E})$ via a set of edge-addition operations such that 1) $\widetilde{G}$ is K-degree anonymous; 2)$V = \widetilde{V}$; and 3) $\widetilde{E} \cap E = E$.

The proposed algorithm is outlined below.

1. Starting from the degree sequence $\boldsymbol{d}$ of the original graph $G(V, E)$, construct a new degree sequence $\tilde{\boldsymbol{d}}$ that is K-anonymous and the L_1 distance, $\|\tilde{\boldsymbol{d}} - \boldsymbol{d}\|_1$ is minimized.

2. Construct a new graph $\widetilde{G}(\widetilde{V}, \widetilde{E})$ such that $\boldsymbol{d}_{\widetilde{G}} = \tilde{\boldsymbol{d}}$, $\widetilde{V} = V$, and $\widetilde{E} = E$ (or $\widetilde{E} \cap E \approx E$ in the relaxed version).

The first step is solved by a linear-time dynamic programming algorithm while the second step is based on a set of graph-construction algorithms given a degree sequence. The authors also extended their algorithms to allow for simultaneous edge additions and deletions. Their empirical evaluations showed that the proposed algorithms can effectively preserve the graph utility (in terms of topological features) while satisfying the K-degree anonymity.

K-neighborhood Anonymity. Zhou and Pei assumed that the adversary knows subgraph constructed by the immediate neighbors of a target node [113]. A node u is **K-neighborhood anonymous** if there exist at least $K-1$ other nodes $v_1, \ldots, v_{K-1} \in V$ such that the subgraph constructed by the immediate neighbors of each node $v_1, \cdots, v_{K-1}$ is isomorphic to the subgraph constructed by the immediate neighbors of

u. A graph satisfies K-neighborhood anonymity if all the nodes are K-neighborhood anonymous. The definition can be extended from the immediate neighbor to the d-neighbors ($d > 1$) of the target vertex, i.e., the vertices within distance d to the target vertex in the network.

Problem 2.2: Given a graph $G(V, E)$, construct a new graph $\widetilde{G}(\widetilde{V}, \widetilde{E})$ satisfying the following conditions: 1) $\widetilde{G}$ is K-neighborhood anonymous; 2)$V = \widetilde{V}$; 3) $\widetilde{E} \cap E = E$; and 4) $\widetilde{G}$ can be used to answer aggregate network queries as accurately as possible.

The simple case of constructing a K-neighborhood anonymous graph satisfying condition 1-3) was shown as NP-hard. The proposed algorithm is outlined below.

1. Extract the neighborhoods of all vertices in the network. A *neighborhood component coding* technique, which can represent the neighborhoods in a concise way, is used to facilitate the comparisons among neighborhoods of different vertices including the isomorphism tests.

2. Organize vertices into groups and anonymize the neighborhoods of vertices in the same group until the graph satisfies K-anonymity. A heuristic of starting with vertices with high degrees is adopted since these vertices are more likely to be vulnerable to structural attacks.

Zhou and Pei studied social networks with vertex attributes information in addition to the unlabeled network topology [113]. The vertex attributes form a hierarchy. Hence, there are two ways to anonymize the neighborhoods of vertices: generalizing vertex labels and adding edges. In terms of utility, it focuses on using anonymized social networks to answer aggregate network queries.

K-automorphism Anonymity. Zou et al. adopted a more general assumption: the adversary can know any subgraph around a certain individual [115]. If such a subgraph can be identified in the anonymized graph with high probability, user α has

a high identity disclosure risk. The authors aimed to construct a graph $\widetilde{G}$ so that for any subgraph $X \subset G$, $\widetilde{G}$ contains at least K subgraphs isomorphic to X.

Definition 2.2: Graph isomorphism and automorphism. Given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, G_1 is isomorphic to G_2 if there exists a bijective function $f : V_1 \rightarrow V_2$ such that for any two nodes $u, v \in V_1$, $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$. If G_1 is isomorphic to itself under function f, G_1 is an automorphic graph, and f is called an automorphic function of G_1.

Definition 2.3: K-automorphic graph. Graph G is a K-automorphic graph if 1) there exist $K-1$ non-trivial automorphic functions of G, $f_1, \ldots, f_{K-1}$; and 2) for any node u, $f_i(u) \neq f_j(u)$ $(i \neq j)$.

If the released graph $\widetilde{G}$ is a K-automorphic graph, when the adversary tries to re-identify node u through a subgraph, he will always get at least K different subgraphs in $\widetilde{G}$ that match his subgraph query. The authors then considered the following problem:

Problem 2.3: Given the original graph G, construct graph $\widetilde{G}$ such that $E \subseteq \widetilde{E}$ and $\widetilde{G}$ is a K-automorphic graph.

The following steps briefly show the framework of their algorithm:

1. Partition graph G into several groups of subgraphs $\{U_i\}$, and each group U_i contains $K_i \geq K$ subgraphs $\{P_{i1}, P_{i2}, \ldots, P_{iK_i}\}$ where any two subgraphs do not share a node or edge.

2. For each U_i, make $P_{ij} \in U_i$ isomorphic to each other by adding edges. Then, there exists function $f^{(i)}_{s,t}(\cdot)$ under which P_{is} is isomorphic to P_{it}.

3. For each edge (u, v) across two subgraphs, i.e. $u \in P_{ij}$ and $v \in P_{st}$ $(P_{ij} \neq P_{st})$, add edge $\left(f^{(i)}_{j,\pi_j(r)}(u), f^{(s)}_{t,\pi_t(r)}(v)\right)$, where $\pi_j(r) = (j + r) \mod K$, $r = 1, 2, \ldots, K-1$.

After the modification, for any node u, suppose $u \in P_{ij}$, define $f_r(\cdot)$ as $f_r(u) = f^{(i)}_{j,\pi_j(r)}(u)$, $r = 1, \ldots, K-1$. Then, $f_r(u)$, $r = 1, \ldots, K-1$, are $K-1$ non-trivial automorphic functions of $\widetilde{G}$, and for any $s \neq t$, $f_s(u) \neq f_t(u)$, which guarantees the K-automorphism.

To better preserve the utility, the authors expected that the above algorithm introduces the minimal number of fake edges, which implies that subgraphs within one group U_i should be very similar to each other (so that Step 2 only introduces a small number of edges), and there are few edges across different subgraphs (so that Step 3 will not add many edges). This depends on how the graph is partitioned. If G is partitioned into fewer subgraphs, there are fewer crossing edges to be added. However, fewer subgraphs imply that the size of each subgraph is large, and more edges within each subgraph need to be added in Step 2. The authors proved that to find the optimal solution is *NP*-complete, and they proposed a greedy algorithm to achieve the goal.

Recently, W. Wu et al. [101] proposed a concept called K-symmetry model similar to K-automorphism. Both of the methods aim to modify the graph so that for any node in graph G, there are at least K-1 other nodes automorphic to the node. The difference is that the method in [101] achieve this goal by the *orbit copying operation*, which adds both edges and nodes. In term of the data utility, different from some K-anonymity approaches such as [63, 113, 115] and feature preserving randomization approaches such as [104, 106], the graph modification procedure in [101] does not aim to preserve one or some particular features. To retrieve meaningful information about the original graph, the authors develop a sampling method that reconstructs an approximate version of the original graph, and the analyst calculates graph features from the approximate version instead of from the anonymized one directly.

Comparison of K-anonymity and randomization approaches. Bonchi et al. [12] compared the K-anonymity, especially the K-degree anonymity approach,

with the randomization procedures on both the privacy and utility aspects. The authors proposed a new information-theoretic perspective on quantifying the level of anonymity that is obtained by random perturbation. Their work quantifies the anonymity level provided by the randomization by means of entropy. Based on thorough experimentation on large datasets and various features, the authors showed that randomization techniques achieve meaningful levels of obfuscation while preserving most of the features of the original graph. They also claimed that sparsification, which removes true edges randomly, outperforms perturbation, as it maintains better the characteristics of the graph at the same anonymity levels.

In [107], the authors also compare randomization approaches with the K-anonymity. Their work adopts a different metric to measure the level of anonymity. Their metric aims to guarantee that the probability for the adversary to successfully re-identity an arbitrary individual in the randomized graph is less than a given threshold. When the entropy based measure satisfies the threshold, it is still possible for a small number of individuals not to meet the privacy anonymity level.

2.3 Privacy Preservation via Generalization

To preserve privacy, both K-anonymity and randomization approaches modify the graph structure by adding and (or) deleting edges and then release the detailed graph. Different from the above two approaches, generalization approaches can be essentially regarded as grouping nodes and edges into partitions called *super-nodes* and *super-edges*. The idea of generalization has been well adopted in anonymizing tabular data. For social network data, the generalized graph, which contains the link structures among partitions as well as the aggregate description of each partition, can still be used to study macro-properties of the original graph.

Hay et al. applied structural generalization approaches that groups nodes into clusters, by which privacy details about individuals can be hidden properly [47]. To ensure node anonymity, they proposed to use the size of a partition as a basic guarantee

against re-identification attacks. Their method obtains a vertex K-anonymous super-graph by aggregating nodes into super-nodes and edges into super-edges, such that, each super-node represents at least K nodes and each super-edge represents all the edges between nodes in two super-nodes. Because only the edge density is published for each partition, it is impossible for the adversary to distinguish between individuals in partition. Note that more than one partition may be consistent with a knowledge query about target individual x. Hence, the size of a partition is used to provide a conservative guarantee against re-identification and there exists an improved bound on the size of candidate sets.

To retain utility, the partitions should fit the original network as closely as possible given the anonymity condition. The proposed method estimates fitness via a maximum likelihood approach. The likelihood is defined as one over the size of possible worlds implied by the partition. For any generalization $\mathcal{G}$, the number of edges in the super-node X is denoted as $c(X, X)$, the number of edges between X and Y is denoted as $c(X, Y)$, the set of possible worlds that are consistent with $\mathcal{G}$ is denoted by $\mathcal{W}(\mathcal{G})$ whose size is given by:

$$|\mathcal{W}(\mathcal{G})| = \prod_{X \in \mathcal{V}} \binom{\frac{1}{2}|X|(|X|-1)}{c(X,X)} \prod_{X,Y \in \mathcal{V}} \binom{|X||Y|}{c(X,Y)}$$

The likelihood for a graph $g \in \mathcal{W}(\mathcal{G})$ is then $1/|\mathcal{W}(\mathcal{G})|$. The partitioning of nodes is chosen so that the generalized graph satisfies privacy constraints and maximizes the utility $(1/|\mathcal{W}(\mathcal{G})|)$.

Their algorithm searches the approximate optimal partitioning, using simulated annealing [82]. Starting with a single partition containing all nodes, the algorithm proposes a change of state by splitting a partition, merging two partitions, or moving a node to a different partition. The movement from one partition to next valid partition is always accepted if it increases the likelihood and accepted with some probability if it decreases the likelihood. Search terminates when it reaches a local maximum.

One problem of this generalization approach is that since the released network only contains a summary of structural information about the original network (e.g., degree distribution, path lengths, and transitivity), users have to generate some random sample instances of the released network. As a result, uncertainty may arise in the later analysis since the samples come from a large number of possible worlds.

2.4 Feature Reconstruction from the Randomized Graph

Wu et al. focused on whether we can reconstruct a graph $\hat{G}$ from the randomized one $\widetilde{G}$ such that $\hat{G}$ is closer to the original graph G than $\widetilde{G}$ in terms of some feature f, i.e., $|f(\hat{G}) - f(G)| \leq |f(\widetilde{G}) - f(G)|$ [99]. If a good reconstructed graph can be found, the data analyst or the attacker can calculate graph features or breach privacy based on the reconstructed graph $\hat{G}$ instead of the randomized one $\widetilde{G}$. In particular, Wu et al. studied the use of low rank approximation approach to reconstruct structural features from the graph randomized via *Rand Add/Del*.

Recall that the edge randomization process can be written in the matrix form $\widetilde{A} = A + E$, where A ($\widetilde{A}$) is the adjacency matrix of the original (randomized) graph and E is the perturbation matrix. In the setting of randomizing numerical data, a data set U with m records of n attributes is perturbed to $\widetilde{U}$ by an additive noise data set V with the same dimensions as U. In other words, $\widetilde{U} = U + V$. Distributions of U can be approximately reconstructed from the perturbed data $\widetilde{U}$ using distribution reconstruction approaches (e.g., [3, 4]) when some a-priori knowledge (e.g., distribution, statistics etc.) about the noise V is available. Specifically, Agrawal and Aggawal [3] provided an expectation-maximization (EM) algorithm for reconstructing the distribution of the original data from perturbed observations. However, it is unclear whether similar distribution reconstruction methods can be derived for network data. This is because 1) it is hard to define distribution for network data; and 2) the randomization mechanism for network data is based on the positions of randomly chosen edges rather than the independent random additive values for all entries for

numerical data.

The low rank approximation has been well investigated as a point-wise reconstruction method in the numerical setting. A spectral filtering based reconstruction method was first proposed in [52] to reconstruct original data values from the perturbed data. Similar methods (e.g., PCA based reconstruction method [51], SVD based reconstruction method [41]) were also investigated. All methods exploited spectral properties of the correlated data to remove the noise from the perturbed one.

Let λ_i ($\tilde{\lambda}_i$) be A's ($\widetilde{A}$'s) i-th largest eigenvalue in magnitude whose eigenvector is $\boldsymbol{x}_i$ ($\tilde{\boldsymbol{x}}_i$). Then, the rank l approximations of A and $\widetilde{A}$ are respectively given by:

$$A_l = \sum_{i=1}^{l} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^T \quad \text{and} \quad \widetilde{A}_l = \sum_{i=1}^{l} \tilde{\lambda}_i \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^T.$$

By choosing a proper l, Wu et al. showed that $\widetilde{A}_l$ can preserve the major information of the original graph and filter out noises added in the rest dimensions. This is because real-world data is usually highly correlated in a low dimensional space while the randomly added noise is distributed (approximately) equally over all dimensions. In $\widetilde{A}_l$, those entries close to 1 are more likely to have true edges while those entries close to 0 are less likely to have edges. Therefore the reconstructed graph $\hat{A}$ can be simply derived by setting the $2m$ largest off-diagonal entries in $\widetilde{A}_l$ as 1, and 0 otherwise. Empirical evaluations showed that many accurate features can be reconstructed via the low rank approximation even when the magnitude of additive noise k equals to $0.8m$.

Wu et al. also empirically studied the link disclosure risk by comparing the number of different edges between the reconstructed graph and the original one, i.e. $\|\hat{A} - A\|_F^2$. The preliminary results [99] showed that the reconstructed graph has more false links compared with the randomized graph $\widetilde{A}$, indicating that the link disclosure risk of the reconstructed graph may not be higher than the randomized graph.

2.5 Anonymizing Rich Graphs

Real social network sources usually contain much richer information in addition to the simple graph structure. For example, in an online social network, the main entities in the data are individuals whose profiles can list lots of demographic information, such as age, gender and location, as well as other sensitive personal data, such as political and religious preferences, relationship status, etc. Between users, there are many different kinds of interactions such as friendship and email communication. Interactions can also involve more than two participants, e.g., many users can play a game together. Bhagat et al. [11] referred to the connections formed in the social networks as *rich interaction graphs*. Various queries on the network data are not simply about properties of the entities in the data, or simply about the pattern of the link structure in the graph, but rather on their combination. Thus it is important for the anonymization to mask the associations between entities and their interactions.

Notice that for rich social networks, a K-anonymous social network may still leak privacy. For example, if all nodes in a K-anonymous group are associated with some sensitive information, the adversary can derive that sensitive attribute of target individuals. Mechanism analogous to l-diversity [68] can be applied here. Several rich graph data models, which may contain labeled vertices/edges in addition to the structural information associated with the network, have been investigated in the privacy-preserving network analysis.

Link Protection in Rich Graphs. Zheleva et al. considered a graph model, in which there are multiple types of edges but only one type of nodes [112]. Edges are classified as either sensitive or non-sensitive. The problem of link re-identification is defined as inferring sensitive relationships from non-sensitive ones. The goal is to attain privacy preservation of the sensitive relationships, while still producing useful anonymized graph data. They proposed to use the number of removed non-sensitive edges to measure the utility loss. Several graph anonymization strategies were pro-

posed, including the removal of all sensitive edges and/or some non-sensitive edges, and the cluster-edge anonymization. In the cluster-edge anonymization approach, all the anonymized nodes in an equivalence class are collapsed into a single super-node and a decision is made on which edges to be included the collapsed graph. One feasible way is to separately publish the number of edges of each type between two equivalence classes. The difference between the cluster-edge anonymization approach and the generalization approach in [47] is that the former aggregates edges by type to protect link privacy while the latter clusters vertices to protect node identities.

Campan and Truta considered an undirected graph model, in which edges are not labeled but vertices are associated with some attributes including identifier, quasi-identifier, and sensitive attributes [15]. Those identifier attributes such as name and SSN are removed while the quasi-identifier and the sensitive attributes as well as the graph structure are released. To protect privacy in network data, they adopted the K-anonymity model for both the quasi-identifier attributes and the quasi-identifier relationship homogeneity. The goal is that any two nodes from any cluster are indistinguishable based on either their relationships or their attributes. They also proposed an edge generalization based method for structural anonymization. They perform social network data clustering followed by anonymization through cluster collapsing. Specifically, the method first partitions vertices into clusters and attaches the structural description (i.e., the number of nodes and the number of edges) to each cluster. From the privacy standpoint, an original node within such a cluster is indistinguishable from other nodes. Then all vertices in the same cluster are made uniform with respect to the quasi-identifier attributes and the quasi-identifier relationship. This homogenization is achieved by using generalization, for both the quasi-identifier attributes and the quasi-identifier relationship. All vertices in the same cluster are collapsed into one single vertex (labeled by the number of vertices and edges in the cluster) and edges between two clusters are collapsed into a single edge (labeled with

the number of edges between them). The method takes into account the information loss due to both the attribute generalization and the changes of structural properties. Users can tune the process to balance the tradeoff between preserving more structural information and preserving more vertex attribute information.

Anonymization Techniques on Rich Graphs. Cormode et al. [22] studied a particular type of network data that can be modeled as bipartite graphs – there are two types of entities, and an association only exists between two entities of different types. One example is the pharmacy (customers buy products). The association between two nodes (e.g., who bought what products) is considered to be private and needs to be protected while properties of some entities (e.g., product information or customer information) are public. Their anonymization method can preserve the graph structure exactly by masking the mapping from entities to nodes rather than masking or altering the graph structure. As a result, analysis principally based on the graph structure is correct. Privacy is ensured in this approach because given a group of nodes, there is a secret mapping from these nodes to the corresponding group of entities. There is no information published that would allow an adversary to learn, within a group, which node corresponds to which entity.

Bhagat et al. adopted a flexible representation of rich interaction graphs which is capable of encoding multiple types of interactions between entities [11]. Interactions involving large number of participants are represented by a hypergraph, denoted by $G(V, I, E)$. V is the node set. Each entity $v \in V$ has a hidden identifier u and a set of properties. Each entity in I is an interaction between/among a subset of entities in V. E is the set of hyperedges: for $v \in V$ and $i \in I$, an edge $(v, i) \in E$ represents node v participates in interaction i. The authors assumed that adversaries know part of the links and nodes in the graph. They presented two types of anonymization techniques based on the idea of grouping nodes in V into several classes. The authors pointed out that merely grouping nodes into several classes can not guarantee the privacy.

For example, consider the case where the nodes within one class form a complete graph via a certain interaction. Then, once the adversary knows the target is in the class, he can be sure that the target must participate in the interaction. The authors provided a safety condition, called *class safety* to ensure that the pattern of links between classes does not leak information: each node cannot have interactions with two (or more) nodes from the same group. Note that the released graph contains the full topological structure of the original graph, some structural attacks such as the *active attack* and *passive attack* [8] can be applied here to de-anonymize the nodes in V. However, the adversary cannot further obtain the attributes of the target, for the attributes of those nodes within the same class are mixed together, which is similar to the anatomy approach [102] for the tabular database.

Beyond the ongoing privacy-preserving social network analysis which mainly focuses on un-weighted social networks, in [26, 65], the authors studied the situations in which the network edges as well as the corresponding weights are considered to be private.

Das et al. considered the problem of anonymizing the weights of edges in the social network [26]. The authors proposed a framework to re-assign weights to edges so that a certain *linear property* of the original graph can be preserved in the anonymized graph. A *linear property* is the property that can be expressed by a specific set of linear inequalities of edge weights. If the newly assigned edge weights also satisfy the set of linear inequalities, the corresponding *linear property* is also preserved. Then, finding new weight for each edge is a linear programming problem. The authors discussed two linear properties in details, single source shortest paths and all pairs shortest paths, and proposed the algorithms that can efficiently construct the corresponding linear inequality sets. Their empirical evaluations showed that the proposed algorithms can considerably improve the edge k-anonymity of the modified graph, which prevents the adversary to identify an edge by its weight.

Liu et al. also proposed two randomization strategies aiming to preserve the shortest paths in the weighted social network [65]. The first one, which is easier to implement, is the Gaussian randomization multiplication strategy. The algorithm multiplies the original weight of each edge by an i.i.d. Gaussian random variable with mean 1 and variance σ^2. In the original graph, if the total weight of the shortest path between two nodes is much smaller than that of the second shortest path, the strategy can preserve the original shortest path with high probability. The authors further proposed the second strategy which can preserve a set of the target shortest paths or even all the shortest paths in the graph. The authors pointed out that all edges can be divided into three categories: the *all-visited edge* which belongs to all shortest paths, the *non-visited edge* which belongs to no shortest path, and the *partially-visited edge* which belongs to some but not all shortest paths. In order to preserve the target shortest paths, one can then reduce the weight of *all-visited edges*, increase the weight of *non-visited edges*, and perturb the weight of *partially-visited edges* within a certain range. The weight sum of a target shortest path is changed and is probably not the same as the original one, but the difference is minimized by the proposed greedy perturbation algorithm.

2.6 Differential Privacy for Querying Social Network Data

Differential privacy [28, 29] is a paradigm of post-processing the output of queries such that the inclusion or exclusion of a single individual from the data set makes no statistical difference to the results found. Differential privacy is agnostic to auxiliary information an adversary may possess, and provides guarantees against arbitrary attacks. Differential privacy is achieved by introducing randomness into query answers.

Definition 2.4: (ϵ-differential privacy) A mechanism $\mathcal{K}$ is ϵ-differentially private if for all databases x and x' differing on at most one element, and any subsets of outputs $S \subseteq Range(\mathcal{K})$,

$$\Pr[\mathcal{K}(x) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(x') \in S].$$

Differential privacy provides formal privacy guarantees that do not depend on an adversary's background knowledge (including access to other databases) or computational power. It focuses on comparing the risk to an individual when included in, versus when not included in the database, which is different from prior work on comparing an adversary's prior and posterior views of an individual. In other words, it achieves the *ad omina* privacy goal: anything that can be learned about a participant from the database should be learnable without access to the database.

Theorem 2.1: [29] For $f : D \rightarrow \mathbf{R}^d$, the mechanism $\mathcal{K}_f$ that adds independently generated noise with distribution $Lap(\Delta f/\epsilon)$ to each of the d output terms satisfies ϵ-differential privacy, where the sensitivity, Δf, is $\Delta f = \max_{x,x'} \|f(x) - f(x')\|_1$ for all x, x' differing in at most one element.

The mechanism for achieving differential privacy computes the sum of the true answer and random noise generated from a Laplace distribution. The magnitude of the noise distribution is determined by the sensitivity of the computation and the privacy parameter specified by the data owner. The sensitivity of a computation bounds the possible change in the computation output over any two neighboring databases (differing at most one record). The privacy parameter controls the amount by which the distributions induced by two neighboring databases may differ (smaller values enforce a stronger privacy guarantee).

Enabling accurate analysis of social network data while preserving differential privacy has been little studied except a few recent results: techniques for computing properties such as degree distributions [48] and clustering coefficient [80]. In social network analysis, the robustness of various graph features such as modularity often has a high sensitivity, which is different from traditional aggregate functions (often with low sensitivity values) on tabular data.

CHAPTER 3: LINK DISCLOSURE ANALYSIS

In this chapter, we investigate the link privacy disclosure risk of a randomized graph. We focus on the situation when the existence of a certain link is considered confidential. When releasing or outsourcing network data, there exist some scenarios that node identities (and even entity attributes) are not confidential but sensitive links between target individuals are confidential and should be protected. For example, in a transaction network, an edge denoting a financial transaction between two individuals is considered confidential while nodes corresponding to individual accounts is non-confidential. To make the discussion concise, we simply assume that the identity of each node is released with the data. In Chapter 4, we will investigate the node privacy disclosure risk and the combination of both the link and node privacy disclosure.

To prevent the entry-wise privacy disclosure in publishing data, randomization is a widely adopted strategy for privacy-preserving data analysis. For numerical data, additive noise based randomization approaches have been well investigated in privacy-preserving data mining (e.g., [3, 4]). For categorical data, randomized response model has also been applied to prevent the privacy disclosure [39, 40, 93]. Randomization can also be used to prevent the disclosure of link privacy.

The process of randomization and the randomization parameter k are assumed to be published along with the released graph. By using adjacency matrix, the edge randomization process can be expressed in the matrix form $\tilde{A} = A + E$, where $E = (e_{ij})_{n\times n}$ is the perturbation matrix: $e_{ij} = e_{ji} = 1$ if edge (i, j) is added, $e_{st} = e_{ts} = -1$ if edge (s, t) is deleted, and 0 otherwise. Naturally, edge randomization can be considered as an additive-noise perturbation. After the randomization, the

randomized graph is expected to be different from the original one. As a result, the node identities as well as the true sensitive or confidential relationship between two nodes are protected. However, we should note that the randomization approaches protect against re-identification in a probabilistic manner.

In Section 3.1, we derive the attacker's prior belief on the existence of a link as well as his posterior belief on the link when observing an existing (or non-existing) link in the released graph. Based on the prior and posterior beliefs, we also derive the minimal randomization magnitude needed to preserve privacy to a given level. In Section 3.2, we derive the enhanced the posterior belief on a link that exploits the similarity between the two nodes. Some results in this chapter are also reported in [103, 106].

3.1 Randomization and Link Privacy

The link disclosure problem of edge-randomized graphs focuses on networks where node identities (and even entity attributes) are not confidential but sensitive links between target individuals are confidential. The problem can be regarded as, compared to not releasing the graph, to what extent releasing a randomized graph $\tilde{G}$ jeopardizes the link privacy.

When it comes to link privacy, it is usually $a_{ij} = 1$ that people want to hide, not $a_{ij} = 0$ and attackers are capable of calculating posterior probabilities. Formally, we use $\Pr(a_{ij} = 1)$ to denote the users' prior belief about the event of $a_{ij} = 1$ and use $\Pr(a_{ij} = 1|\tilde{G})$ to denote its posterior belief about $a_{ij} = 1$. The released graph $\tilde{G}$ is regarded as jeopardizing the privacy if $\Pr(a_{ij} = 1|\tilde{G}) > \Pr(a_{ij} = 1)$. If only the basic statistics of a graph such as m and n are given, the adversary's prior belief about the existence of edge (i, j) (without exploiting the released graph) can be calculated as

$$\Pr(a_{ij} = 1) = \frac{2m}{n(n-1)}. \tag{3.1}$$

For *Rand Add/Del*, with the released graph and perturbation parameter k, the pos-

terior belief when observing $\tilde{a}_{ij}$ is

$$\Pr(a_{ij}=1|\tilde{a}_{ij}=1)=\frac{m-k}{m},\quad \text{and}\quad \Pr(a_{ij}=1|\tilde{a}_{ij}=0)=\frac{k}{N-m}, \tag{3.2}$$

where $N=\binom{n}{2}$. Next, we give the posterior probability of $\Pr(a_{ij}=1|\tilde{a}_{ij})$ under *Rand Switch* randomization strategy.

We here assume that the attacker has no other information except each vertex's degree which is kept unchanged in the perturbed data for the *Rand Switch* strategy. Intuitively, $S_i=\frac{d_i}{n-1}$ is the probability that a randomly selected vertex turns out an neighbor of vertex i's. Therefore, the prior probability can be shown as

$$\Pr(a_{ij}=1)=S_i+S_j-S_iS_j. \tag{3.3}$$

The posterior probability $\Pr(a_{ij}=1|\tilde{a}_{ij}=1)$ is the probability that an edge (i,j) in $\tilde{G}$ is a true edge in G. let c_i denote the number of false edges associated to vertex i in graph $\tilde{G}$, i.e. $c_i=\frac{1}{2}\sum_{j=1}^{n}|\tilde{a}_{ij}-a_{ij}|$, and $\mathbf{E}(c_i)$ is its expectation. Then, $P_i=1-\frac{\mathbf{E}(c_i)}{d_i}$ is vertex i's proportion of true edges. Hence,

$$\Pr(a_{ij}=1|\tilde{a}_{ij}=1)=P_i+P_j-P_iP_j \tag{3.4}$$

Similarly, $Q_i=\frac{\mathbf{E}(c_i)}{n-1-d_i}$ is vertex i's proportion of false edges,

$$\Pr(a_{ij}=1|\tilde{a}_{ij}=0)=Q_i+Q_j-Q_iQ_j \tag{3.5}$$

The key of calculating (3.4) and (3.5) is to calculate $\mathbf{E}(c_i)$. Here, we give the result on its calculation.

Result 3.1: For *Rand Switch*, denote $c_i=\frac{1}{2}\sum_{j\neq i}|\tilde{a}_{ij}-a_{ij}|$, $0\leq c_i\leq C_i:=\min\{d_i,n-1-d_i\}$. Denote q_i as the probability that a switching occurs to vertex i. It can be approximated as $q_i\approx\frac{d_i}{m}+\sum_{k\neq i}\frac{d_k}{m}\cdot\frac{d_i-a_{ik}}{m-d_k}$. The expectation of c_i is shown as

$$\mathbf{E}(c_i)=(0,1,2,\ldots,C_i)\left((1-q_i)I+q_iP_i\right)^k\boldsymbol{e}_1.$$

where $\boldsymbol{e}_1 = (1, 0, 0, \ldots, 0)^T$, $P_i = (p_{st}^{(i)})_{(C_i+1)\times(C_i+1)}$ and

$$p_{st}^{(i)} = \begin{cases} \frac{t^2}{d_i(n-1-d_i)}, & (s = t-1) \\ \frac{t(n-1-2t)}{d_i(n-1-d_i)}, & (s = t) \\ \frac{(d_i-t)(n-1-d_i-t)}{d_i(n-1-d_i)}, & (s = t+1) \\ 0, & (otherwise). \end{cases} \tag{3.6}$$

Proof. The probability that a switching occurs to vertex is a constant. By saying a switch occurs to vertex i, we mean that one of the two switched edges connects to vertex i. Suppose one switch occurs to vertex i. In the ith row of the adjacency matrix $\boldsymbol{a}_i = (a_{i1}, a_{i2}, \ldots, a_{i,n})$, one component, say a_{ip}, changes from 1 to 0 and another component a_{iq} change from 0 to 1. Equivalently, we replace a 1 in $\boldsymbol{a}_i$. Since we select the edges uniformly, every 1 (0) has same possibility to become 0 (1). Given r of the k times of switch to vertex i, we first calculate $E(c_i|r)$. The change of c_i follows the Markov chain with the stationary probabilities, and c_i has finite states: $0, 1, \ldots, C_i$. Then, it is easy to establish the transition matrix P_i whose elements $p_{st}^{(i)} = \Pr(c_i^{(n+1)} = s | c_i^{(n)} = t)$ is shown in (3.6). The initial probability distribution vector is $\boldsymbol{e}_1$. Hence,

$$\begin{aligned} \mathbf{E}(c_i|r) &= \sum_{x=0}^{C_i} x \Pr(c_i = x) = (0, 1, 2, \ldots, C_i) P_i^r \boldsymbol{e}_1. \\ \mathbf{E}(c_i) &= \sum_{x=0}^{k} \mathbf{E}(c_i|r = x) \Pr(r = x) = \sum_{x=0}^{k} \binom{k}{x} q_i^x (1-q_i)^{k-x} \mathbf{E}(c_i|r = x) \\ &= (0, 1, 2, \ldots, C_i) \left((1-q_i)I + q_i P_i\right)^k \boldsymbol{e}_1. \end{aligned}$$

□

3.1.1 Link Privacy Protection vs. Perturbation k

As the magnitude of the perturbation increases, there are more false edges in the network, and the released graph approaches to a pure random graph. Therefore, large

magnitude of perturbation can enhance the privacy protection, but, on the other hand, may decrease the utility, resulting a dataset useless for mining any meaningful information. In this section, we develop our formal privacy protection measures.

We define the absolute measure of protection as

$$\tau_a(i,j) = 1 - \max\{\Pr(a_{ij} = 1 \mid \tilde{a}_{ij} = 0), \Pr(a_{ij} = 1 \mid \tilde{a}_{ij} = 1)\} \tag{3.7}$$

Note that the second term in (3.7) can be considered as the maximal suspicion of existing $a_{ij} = 1$. The relative measure of protection is defined as

$$\tau_r(i,j) = \frac{\tau_a(i,j)}{1 - \Pr(a_{ij} = 1)} \tag{3.8}$$

Our following result shows how to calculate the privacy measure.

Result 3.2: For *Rand Add/Del*, assume $k \leq (1-r)m$ where $r = \frac{2m}{n(n-1)}$ is the sparse ratio, we have

$$\tau_a(i,j) = \frac{k}{m}, \quad \tau_r(i,j) = \frac{kN}{m(N-m)}$$

where $N = \binom{n}{2}$.

For *Rand Switch*, after k switches, for vertex i, let c_i denote the number of false edges associated to vertex i in graph $\tilde{G}$, i.e. $c_i = \frac{1}{2}\sum_{j=1}^{n} |\tilde{a}_{ij} - a_{ij}|$, and $\mathbf{E}(c_i)$ is its expectation. Then,

$$\tau_a(i,j) = (1 - P_i)(1 - P_j), \tag{3.9}$$

$$\tau_r(i,j) = \frac{1 - P_i}{1 - S_i} \cdot \frac{1 - P_j}{1 - S_j}, \tag{3.10}$$

where $P_i = 1 - \frac{\mathbf{E}(c_i)}{d_i}$, and $S_i = \frac{d_i}{n-1}$.

Proof. The result for *Rand Add/Del* is easy to derive. We only give the proof for *Rand Switch*. Notice that P_i is a decreasing function of k and Q_i is an increasing with

k, and

$$\lim_{k\to\infty} P_i = \lim_{k\to\infty} Q_i = \frac{d_i}{n-1}.$$

We thus have, $P_i \geq Q_i$. As a result

$$P_i + P_j - P_iP_j \geq Q_i + Q_j - Q_iQ_j.$$

(3.9) and (3.10) is then derived by incorporating (3.4) and (3.3) in (3.8). □

The measures of protection (τ_a and τ_r) are defined in terms of one individual edge. In the privacy preserving data mining, one natural question is how many perturbations we need such that we can guarantee the protection for all individual edges are above some threshold. Formally, we expect

- For *Rand Add/Del* strategy,

$$J_1(k) = \min_{i,j} \tau_r(i,j) = \frac{kN}{m(m-N)} > 1-\varepsilon.$$

- For *Rand Switch* strategy,

$$J_2(k) = \min_{i,j} \tau_r(i,j) = \min_{i,j} \left\{ \frac{1-P_i}{1-S_i} \cdot \frac{1-P_j}{1-S_j} \right\} > 1-\varepsilon.$$

It is easy to check that the protection for all individual edges remains the same with *Rand Add/Del* strategy. The relative measure in *Rand Switch* is a function of k, d_i, and d_j. Our next result shows we only need to consider the protection of the edges that connect the two vertices with the smallest degrees.

Result 3.3: We re-numerate the vertices by their degree in ascending order: $d_1 \leq d_2 \leq \cdots \leq d_n$,

$$J_2(k) = \frac{1-P_1}{1-S_1} \cdot \frac{1-P_2}{1-S_2}, \tag{3.11}$$

Proof. We first prove that given a fixed k, if two vertices i and j, $d_i \leq d_j$, then

$$\frac{1-P_i}{1-S_i} \leq \frac{1-P_j}{1-S_j}. \tag{3.12}$$

Table 3.1: τ_r vs. k for two strategies on Political Book data

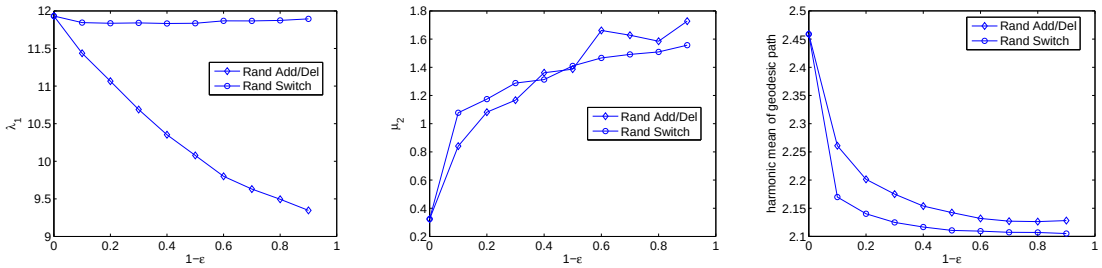
$1-\varepsilon$	Rand Add/Del	Rand Switch
0.1	48	54
0.2	96	84
0.3	150	114
0.4	210	141
0.5	282	174
0.6	372	210
0.7	492	258
0.8	654	318
0.9	936	420

To a single vertex i, *Rand Switch* strategy actually rearranges the position of 1 and 0 on the ith row of the adjacency matrix. A false edge of vertex i corresponds to a 1 reallocated elsewhere in the ith row of the adjacency matrix. Hence, to produce the same proportion of false edges, the number of 0's in j-th row of adjacency matrix should at least increase to $\frac{d_j}{d_i}(n-1-d_i)$:

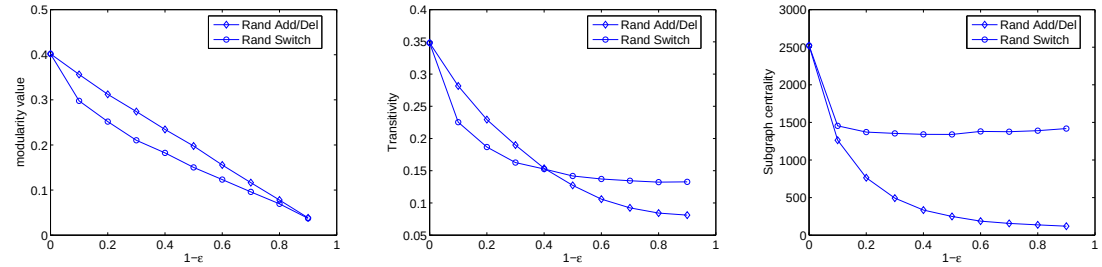
$$\frac{\mathbf{E}(c_i)}{n-1-d_i} \leq \frac{\mathbf{E}(c_j)}{\frac{d_j}{d_i}(n-1-d_i)} \leq \frac{\mathbf{E}(c_j)}{\frac{d_j}{d_i}(n-1-d_j)},$$

and with some simple deduction (3.12) follows. Since $d_1 \leq d_2 \leq \cdots \leq d_n$, then by the above property, (3.11) stands. □

Table 3.1 shows the number of perturbations we need for *Rand Add/Del* strategy and *Rand Switch* when we aim to achieve different levels of privacy protection $(1-\varepsilon)$. Similarly Figure 3.1 shows how graph characteristics vary with different privacy protection thresholds for both *Rand Add/Del* and *Rand Switch* strategies. We can see the higher the privacy protection we aim, the more perturbation we need, and the less the utility of the graph we can achieve.



(a) λ_1 (b) μ_2 (c) harmonic mean of geodesic path



(d) modularity (e) transitivity (f) subgraph centrality

Figure 3.1: Graph characteristic vs. varying privacy protection on Political Book data

3.2 Enhanced Posterior Link Beliefs with Proximity Measures

In this section, we investigate an attacking model that exploits the relationship between the probability of existence of a link and the similarity measure values of node pairs in the released randomized graph.

3.2.1 Existence of Links vs. Similarity Measure

Let m_{ij} be a similarity measure on node pair (i, j) in graph G (a larger value of m_{ij} indicates that nodes i and j are more similar). We apply four similarity measures in our work. The first one is the number of common neighbors: $CN_{ij} = \sum_{k=1}^{n} a_{ik}a_{kj}$. The second one is the Adamic/Adar measure [2] , which is the weighted number of common neighbors. The weights are assigned based on the information theory: $Ad_{ij} = \sum_{k=1}^{n} \frac{1}{\log d_k} a_{ik}a_{kj}$, where d_k is the degree of node k. The third one is the Katz measure, which is a weighted sum of the number of paths in the graph that connect two nodes. Shorter paths are given the larger weight with parameter β

[53]: $K_{ij} = \sum_{k=1}^{\infty} \beta^k P_{ij}^{(k)}$, where $P_{ij}^{(k)}$ denotes the number of paths from i to j with length equal to k while β is a damping factor. In this work, we take $\beta = 0.1$. The fourth one is the commute time CT_{ij}, which is the expected steps of random walks from i to j and back to i. The commute time is a distance measure: more similar nodes have smaller CT values. The commute time can be calculated through the eigenvalues and eigenvectors of the graph's normal matrix [66]. Let $N = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ where $D = \text{diag}\{d_1, d_2, \ldots, d_n\}$. N has n real eigenvalues: $\nu_1 \geq \nu_2 \geq \nu_3 \cdots \nu_n$ with corresponding eigenvectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n$, and let z_{ki} denote the the k'th entry of $\boldsymbol{z}_i$. Then

$$CT_{ij} = 2m \sum_{k=2}^{n} \frac{1}{1-\nu_k} \left(\frac{z_{ki}}{\sqrt{d_i}} - \frac{z_{kj}}{\sqrt{d_j}} \right)^2.$$

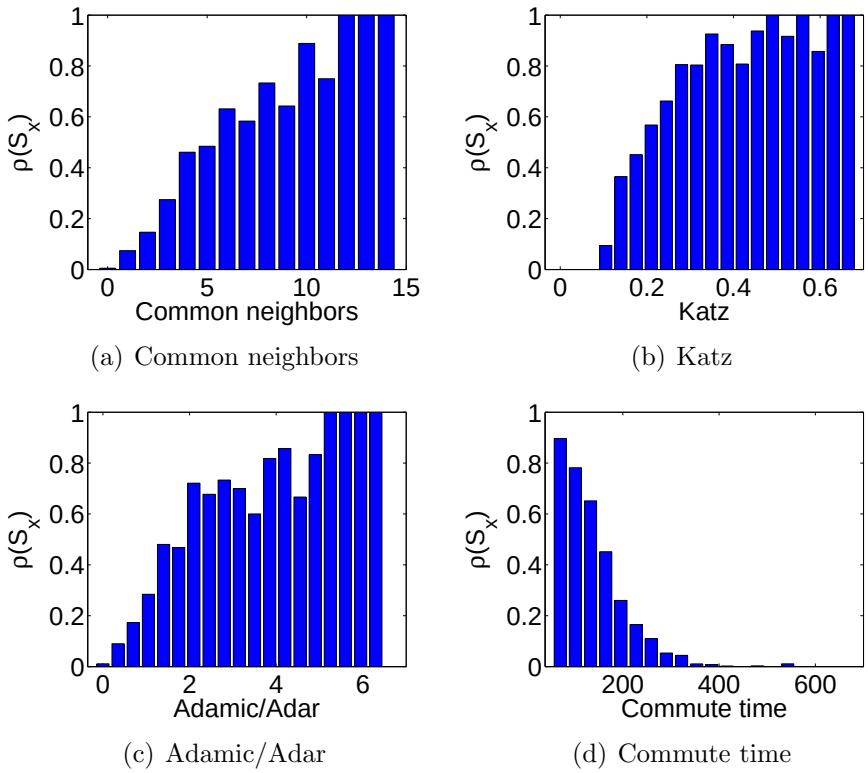
Let $\rho(\Omega)$ denote the proportion of true edges in the set of node pairs Ω:

$$\rho(\Omega) = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} a_{ij},$$

where $|\Omega|$ denotes the number of elements in set Ω. Let $S_x = \{(i,j) : m_{ij} = x\}$ denote the set of all node pairs with the similarity measure $m_{ij} = x$. Hence $\rho(S_x)$ denotes the proportion of true edges in the S_x, which can be considered as the probability of existence of a link between node pair (i,j) in S_x. Next, we empirically show how $\rho(S_x)$ varies with x in real social networks.

Figure 3.2 shows how the proportions of true edges in S_x are varied with similarity measure values x in terms of four measures (Common neighbors, Katz, Adamic/Adar, and Commute time) in the US political books network (*polbooks*). We can observe that $\rho(S_x)$ increases with x. In other words, the probability that $a_{ij} = 1$ is highly correlated with similarity measure m_{ij}: the larger m_{ij} is, the more likely a_{ij} is equal to 1.

We then perturb the *polbooks* network by adding 200 false edges and deleting 200 true edges. From the perturbed graph $\tilde{G}$, we define $\tilde{S}_x = \{(i,j) : \tilde{m}_{ij} = x\}$ as the set of node pairs with similarity measure $\tilde{m}_{ij} = x$. Figure 3.3 shows how the proportions



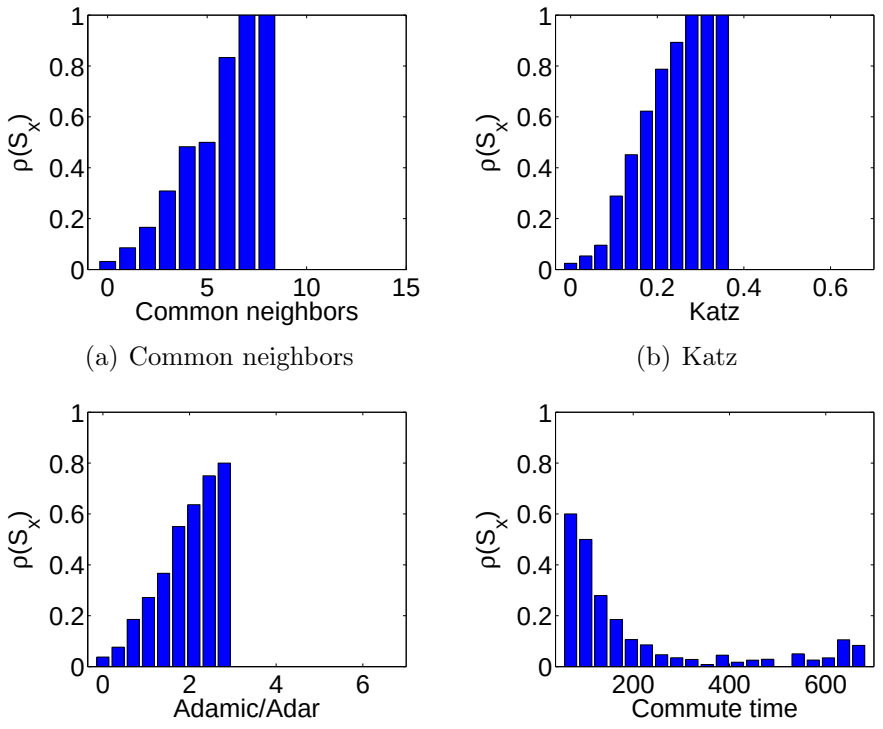
(a) Common neighbors (b) Katz

(c) Adamic/Adar (d) Commute time

Figure 3.2: Similarity measure vs. the prob. of true edges in the original graph ($\rho(S_x)$) for *polbooks*

of true edges in $\tilde{S}_x$ (i.e., the probability of existence of a link) are varied with similarity measure values x in terms of four measures in the randomized *polbooks* network. We can observe that the same pattern still holds even if the randomized graph itself is quite different from the original one (200 false edges out of 441 edges). In the next section, we will show how attackers exploit $\tilde{m}_{ij}$ in the perturbed graph $\tilde{G}$ to improve their posterior belief on existence of a true link between nodes (i, j) in the original graph.

Proximity measures have been shown to be effective in the classic link prediction problem [62] (i.e., predicting the future existence of links among nodes given a snapshot of a current graph). In [62], the authors compute the similarity measures of all the node pairs, and regard the node pairs with high similarity has greater probability to be connected in the future. The strategy is consistent with our observation.



(a) Common neighbors (b) Katz

(c) Adamic/Adar (d) Commute time

Figure 3.3: Similarity measure vs. the prob. of true edges in the randomized graph ($\rho(\tilde{S}_x)$) for *polbooks*

3.2.2 Link Prediction by Exploiting Similarity Measure

In this section, we quantify how much the posterior belief can be enhanced by exploiting similarity measure between two nodes (i, j) in the randomized graph. Recall that the randomization strategy is to randomly add k false edges followed by deleting k true edges. In other words, every true link is to be deleted independently with probability p_1 and every non-existing link is to be added independently with probability p_2. We can easily derive $p_1 = k/m$ and $p_2 = k/[\binom{n}{2} - m]$.

Let $\tilde{m}_{ij}$ denote the similarity measure of nodes i and j in $\tilde{G}$. We define $\tilde{S}_x = \{(i, j) : \tilde{m}_{ij} = x\}$ as the set of node pairs with $\tilde{m}_{ij} = x$ in the perturbed graph. Then we have $\Pr(a_{ij} = 1|\tilde{m}_{ij} = x) = \rho(\tilde{S}_x)$, and $\Pr(a_{ij} = 0|\tilde{m}_{ij} = x) = 1 - \rho(\tilde{S}_x)$. Recall that $\rho(\tilde{S}_x)$ denotes the proportion of true edges in the set $\tilde{S}_x$ derived from the perturbed graph. Also notice that $\Pr(\tilde{a}_{ij} = 1|a_{ij} = 1) = 1 - p_1$ and $\Pr(\tilde{a}_{ij} = 1|a_{ij} = 0) = p_2$.

With the Bayes' theorem, the posterior belief is then given by

$$\Pr(a_{ij}=1|\tilde{a}_{ij}=1,\tilde{m}_{ij}=x)=\frac{(1-p_1)\rho(\tilde{S}_x)}{(1-p_1)\rho(\tilde{S}_x)+p_2[1-\rho(\tilde{S}_x)]}, \tag{3.13}$$

$$\Pr(a_{ij}=1|\tilde{a}_{ij}=0,\tilde{m}_{ij}=x)=\frac{p_1\rho(\tilde{S}_x)}{p_1\rho(\tilde{S}_x)+(1-p_2)[1-\rho(\tilde{S}_x)]}. \tag{3.14}$$

(3.13) ((3.14)) shows the enhanced posterior belief that an observed (missing) edge (i,j) in the $\tilde{G}$ is a true edge in G. The following property shows that the event of an observed link $\tilde{a}_{ij}=1$ usually has more indications to be a true link than that of $\tilde{a}_{ij}=0$.

Property 3.1: Let r denote the sparse ratio of the graph, $r=m/\binom{n}{2}$. If $k\leq(1-r)m$, given a fixed x, we have the following inequality stands:

$$\Pr(a_{ij}=1|\tilde{a}_{ij}=1,\tilde{m}_{ij}=x)\geq\Pr(a_{ij}=1|\tilde{a}_{ij}=0,\tilde{m}_{ij}=x). \tag{3.15}$$

Proof. It is easy to verify that when $1-p_1-p_2\geq 0$, Inequality (3.15) stands if and only if $(1-p_1-p_2)[1-\rho(\widetilde{S}_x)]\geq 0$. We need only guarantee $1-p_1-p_2\geq 0$. Notice that $p_1=\frac{k}{m}$, and $p_2=\frac{k}{\binom{n}{2}-m}$, then we have

$$\begin{aligned}1-p_1-p_2\geq 0\Leftrightarrow 1-\frac{k}{m}-\frac{k}{\binom{n}{2}-m}\geq 0\Leftrightarrow\binom{n}{2}k\leq m\left[\binom{n}{2}-m\right]\\ \Leftrightarrow k\leq\left[1-\frac{m}{\binom{n}{2}}\right]m=(1-r)m.\end{aligned}$$

□

Many real-world social networks are very sparse ($r\approx 0$). Hence $k\leq(1-r)m$ is usually satisfied. We thus focus on the risk of the released links, $\Pr(a_{ij}=1|\tilde{a}_{ij}=1,\tilde{m}_{ij}=x)$.

One issue here is that attackers cannot know the proportion of true edges in $\tilde{S}_x$ from the perturbed graph. What they can know actually is the proportion of observed edges in $\tilde{S}_x$. Our next result shows the maximum likelihood estimate of $\rho(\tilde{S}_x)$ can be

derived from the proportion of observed edges in $\tilde{S}_x$.

Result 3.4: Given the perturbed graph and a fixed x, define $\tilde{S}_x^1 = \tilde{S}_x \cap \tilde{E} = \{(i,j) : \tilde{a}_{ij} = 1, \tilde{m}_{ij} = x\}$. Assume $p_1 + p_2 \neq 1$, then the maximum likelihood estimator (MLE) of $\rho(\tilde{S}_x)$ is given by

$$\hat{\rho}\left(\tilde{S}_x\right) = \frac{|\tilde{S}_x^1|/|\tilde{S}_x| - p_2}{1 - p_1 - p_2}, \tag{3.16}$$

and the MLE is unbiased.

Proof. Let $N = |\widetilde{S}_x|$, $N_1 = |\widetilde{S}_x^1|$ and $\rho = \rho(\widetilde{S}_x)$. Then, for a randomly selected node pair (i,j), $\tilde{a}_{ij}$ is a Bernoulli random variable:

$$\Pr(\tilde{a}_{ij} = 1 | \tilde{m}_{ij} = x) = (1 - p_1)\rho + p_2(1 - \rho)$$
$$\Pr(\tilde{a}_{ij} = 0 | \tilde{m}_{ij} = x) = p_1\rho + (1 - p_2)(1 - \rho)$$

Then the likelihood function of $\widetilde{S}_x$ is

$$L = [(1 - p_1)\rho + p_2(1 - \rho)]^{N_1} [p_1\rho + (1 - p_2)(1 - \rho)]^{N - N_1}.$$

Take derivative to $\ln L$ with respect of ρ, we have

$$\frac{d \ln L}{d\rho} = \frac{N_1(1 - p_1 - p_2)}{(1 - p_1)\rho + p_2(1 - \rho)} - \frac{(N - N_1)(1 - p_1 - p_2)}{p_1\rho + (1 - p_2)(1 - \rho)}.$$

Set $\frac{d \ln L}{d\rho} = 0$, we have $\hat{\rho} = \frac{N_1/N - p_2}{1 - p_1 - p_2}$, and the unbiasedness is then obvious. □

By replacing $\rho(\tilde{S}_x)$ in (3.13) with $\hat{\rho}(\tilde{S}_x)$ (shown in (3.16)), we have derived our enhanced posterior belief $\Pr(a_{ij} = 1 | \tilde{a}_{ij} = 1, \tilde{m}_{ij} = x)$. Attackers may simply calculate the posterior belief of all node pairs in the perturbed graph and choose top-t node pairs as predicted candidate links.

For those similarity measures with continuous ranges (e.g., commute time), the number of node pairs with similarity measure equal exactly to x is usually small. In practice, we can apply histogram approximation by partitioning the value of the

similarity measure: $x_0 \leq x_1 \leq \cdots \leq x_i \leq \cdots$, and for $x \in [x_{i-1}, x_i)$

$$\frac{|\tilde{S}_x^1|}{|\tilde{S}_x|} = \frac{|\{(i,j) : \tilde{a}_{ij} = 1, \tilde{m}_{ij} = x \in [x_{i-1}, x_i)\}|}{|\{(i,j) : \tilde{m}_{ij} = x \in [x_{i-1}, x_i)\}|}.$$

A probably more statistically preferred method is to use the kernel estimator:

$$\frac{|\tilde{S}_x^1|}{|\tilde{S}_x|} = \frac{\sum_{i<j} \tilde{a}_{ij} K[(x - m_{ij})/h]}{\sum_{i<j} K[(x - m_{ij})/h]},$$

where $K(x)$ is the p.d.f. of the standard normal distribution and h is the parameter controlling the smoothness.

We would emphasize that our enhanced posterior belief $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij} = x)$ more accurately reflects the existence of a true link than the posterior belief $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1)$ without exploiting the similarity measure derived in previous work [106]. We can see that $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1)$ (shown in (3.2)) is the same for all observed links. On the contrary, our enhanced posterior belief $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij} = x)$ tends to be larger for those observed links with higher similarity values, and tends to be smaller for links with lower similarity values. Hence, it can more accurately reflect the existence of true links. We show our theoretical explanations in Results 3.5 and 3.6 and will compare the precisions of top-t predicted links derived from these two posterior beliefs in our empirical evaluations.

Result 3.5: $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij} = x)$ is an increasing function of $\rho(\tilde{S}_x)$, and when $\rho(\tilde{S}_x) \geq \frac{p_2}{p_1+p_2}$, we have the following inequality stands:

$$\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij} = x) \geq \Pr(a_{ij} = 1|\tilde{a}_{ij} = 1). \tag{3.17}$$

Proof. Notice that $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1) = \frac{m-k}{m} = 1 - p_1$, and with (3.13), it is easy to verify this result. □

Our next result shows more clearly the relationship between a-priori belief (3.1), posterior belief without exploiting similarity measures (3.2), and our enhanced posterior belief with exploiting similarity measures in (3.13) and (3.14).

Result 3.6: Both the sum of a-priori belief over all node pairs and the sum of posterior belief (without exploiting similarity measures) overall all node pairs are equal to the number of edges:

$$\sum_{i<j} \Pr(a_{ij} = 1) = \sum_{i<j} \Pr(a_{ij} = 1|\tilde{a}_{ij}) = m.$$

The sum of our enhanced posterior belief (with exploiting similarity measures) also approaches to the number of edges:

$$\sum_{i<j} \Pr(a_{ij} = 1|\tilde{a}_{ij}, \tilde{m}_{ij}) \to m \quad \text{as} \quad n \to \infty.$$

Proof. $\sum_{i<j} \Pr(a_{ij} = 1) = m$ is obvious. Notice that the number of edges does not change along the perturbation, then we have

$$\begin{aligned}\sum_{i<j} \Pr(a_{ij} = 1|\tilde{a}_{ij}) &= \sum_{(i,j)\in\widetilde{E}} \Pr(a_{ij} = 1|\tilde{a}_{ij} = 1) + \sum_{(i,j)\notin\widetilde{E}} \Pr(a_{ij} = 1|\tilde{a}_{ij} = 0) \\ &= m \cdot \frac{m-k}{m} + \left[\binom{n}{2} - m\right] \cdot \frac{k}{\binom{n}{2} - m} = m. \end{aligned} \tag{3.18}$$

Given a randomized graph $\widetilde{G}$, $\tilde{a}_{ij}$ and $\tilde{m}_{ij}$ are fixed for all i and j. Let Φ denote the set of $\tilde{m}_{ij}$ values in $\widetilde{G}$, we have

$$\begin{aligned}\frac{1}{m}\sum_{i<j} \Pr(a_{ij} = 1|\tilde{a}_{ij}, \tilde{m}_{ij}) = \sum_{x\in\Phi} \Bigg\{ &\frac{1}{m} \sum_{(i,j)\in\widetilde{S}_x^1} \Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij} = x) \\ &+ \frac{1}{m} \sum_{(i,j)\in\widetilde{S}_x - \widetilde{S}_x^1} \Pr(a_{ij} = 1|\tilde{a}_{ij} = 0, \tilde{m}_{ij} = x) \Bigg\}. \end{aligned} \tag{3.19}$$

Consider the first term of the right hand side of (3.19). To make the notation simple, we write $\rho = \rho(\widetilde{S}_x)$.

$$\frac{1}{m} \sum_{(i,j)\in\widetilde{S}_x^1} \Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij} = x) = \frac{(1-p_1)\rho}{(1-p_1)\rho + p_2(1-\rho)} \cdot \frac{\sum_{i<j} \tilde{a}_{ij}}{|\widetilde{S}_x|} \cdot \frac{|\widetilde{S}_x|}{m}. \tag{3.20}$$

Since $|\widetilde{S}_x| \to \infty$ as $m \to \infty$. Given x, for all $(i,j) \in \widetilde{S}_x$, $\tilde{a}_{ij}$ are i.i.d. bernoulli random

variables, and with the law of large numbers, we have

$$\frac{\sum_{i<j}\tilde{a}_{ij}}{|\widetilde{S}_x|} \to \Pr(\tilde{a}_{ij}=1) = (1-p_1)\rho + p_2(1-\rho), \quad \text{as } |\widetilde{S}_x| \to \infty \tag{3.21}$$

Substituting (3.21) into (3.20), we have,

$$\lim_{m\to\infty}\left[\frac{1}{m}\sum_{(i,j)\in\widetilde{S}_x^1}\Pr(a_{ij}=1|\tilde{a}_{ij}=1,\tilde{m}_{ij}=x)\right] = (1-p_1)\rho\frac{|\widetilde{S}_x|}{m}. \tag{3.22}$$

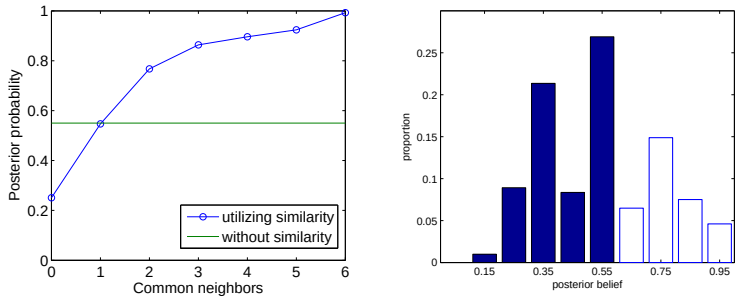
Note that $\frac{|\widetilde{S}_x|}{m} \leq 1$, the above equation is well defined. Similarly, we also have

$$\lim_{m\to\infty}\left[\frac{1}{m}\sum_{(i,j)\in\widetilde{S}_x-\widetilde{S}_x^1}\Pr(a_{ij}=1|\tilde{a}_{ij}=0,\tilde{m}_{ij}=x)\right] = p_1\rho\frac{|\widetilde{S}_x^1|}{m}. \tag{3.23}$$

Combining (3.19), (3.22) and (3.23) together, we have

$$\frac{1}{m}\sum_{i<j}\Pr(a_{ij}=1|\tilde{a}_{ij},\tilde{m}_{ij}) \to \sum_{x\in\Phi}\left([(1-p_1)\rho+p_1\rho]\frac{|\widetilde{S}_x^1|}{m}\right) \to 1 \quad \text{as } m\to\infty.$$

Then, due to the law of large number, we can conclude that we prove the result. □



(a) Posterior belief vs. Common neighbors (b) Posterior belief distribution

Figure 3.4: Posterior belief for *polbooks* network

Figure 3.4 shows the relationship between the two posterior beliefs and the common neighbors for the *polbooks* data. We set $k = 200$. We can observe that the posterior belief without exploiting the similarity measure, $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1)$, is 0.55 for all observed links. However, our enhanced posterior belief $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1, \tilde{m}_{ij})$

are greater than 0.55 for those links with more than 2 common neighbors as shown in Figure 3.4(a). Figure 3.4(b) shows the distribution of the calculated posterior belief values. We can observe that 33.5% of released links have their posterior beliefs enhanced with similarity measures.

3.2.3 Privacy Protection vs Perturbation k

When attackers utilize the similarity measure, the absolute measure of protection for an individual link (i, j) can be defined as

$$\tau_a(i,j) = 1 - \max_x \left\{ \max_{t=0,1} \Pr(a_{ij} = 1 | \tilde{a}_{ij} = t, \tilde{m}_{ij} = x) \right\} \tag{3.24}$$

where the second term denotes the maximal suspicion of existing $a_{ij} = 1$. Compared with the protection under the attack without exploiting similarity measures, we define the relative measure of protection as

$$\tau_r(i,j) = \frac{\tau_a(i,j)}{1 - \max_{t=0,1} \Pr(a_{ij} = 1 | \tilde{a}_{ij} = t)}.$$

The measures of protection (τ_a and τ_r) are defined in terms of one individual edge. In the privacy preserving data mining, one natural question is how many perturbations we need such that we can guarantee the protection for all individual edges are above the threshold. Our next result shows the formula of the minimum number of perturbations to achieve the protection of all individual links. It is of great importance to evaluate the relationship between the required minimum number of perturbations and the utility loss of the perturbed graph.

Result 3.7: In the original graph, let $S_x = \{(i,j) : m_{ij} = x\}$, $\rho_{\max} = \max_x \rho(S_x)$, and sparse ratio $r = m/\binom{n}{2}$. When the protection threshold $\epsilon < \frac{1-\rho_{\max}}{1-r}$, there exists the minimum k such that $\tau_r(i,j) \geq \epsilon$ stands for all the node pair (i,j) is given by:

$$k_{\min} = \frac{[(1-r)\epsilon\rho_{\max} - r(1-\rho_{\max})]m}{\epsilon(\rho_{\max} - r)}. \tag{3.25}$$

Proof. When $k \leq (1-r)m$, with Result 3.1 and 3.5, we have that

$$\max_x \left\{ \max_{t=0,1} \Pr(a_{ij} = 1 | \tilde{a}_{ij} = t, \tilde{m}_{ij} = x) \right\} = \Pr(a_{ij} = 1 | \tilde{a}_{ij} = 1, \tilde{m}_{ij} = x_0),$$

where x_0 is the value such that $\rho(\widetilde{S}_x)$ is maximized: $\rho(\widetilde{S}_{x_0}) = \max_x \rho(\widetilde{S}_x)$. Let $\tilde{\rho}_{\max} = \rho(\widetilde{S}_{x_0})$. Meanwhile,we can also conclude

$$\max_{t=0,1} \Pr(a_{ij} = 1 | \tilde{a}_{ij} = t) = \Pr(a_{ij} = 1 | \tilde{a}_{ij} = 1).$$

Then we have

$$\tau_r(i,j) = \frac{p_2[1 - \tilde{\rho}_{\max}]}{p_1[(1-p_1)\tilde{\rho}_{\max} + p_2(1 - \tilde{\rho}_{\max})]}. \tag{3.26}$$

Substitute $p_1 = \frac{k}{m} = \frac{k}{rN}$ and $p_2 = \frac{k}{N-m} = \frac{k}{(1-r)N}$ into (3.26), we can verify that $\tau_r(i,j)$ is an increasing function of k, and the maximum value is $\frac{1-\tilde{\rho}_{\max}}{1-r}$ when $k = (1-r)m$.

When $k \geq (1-r)m$, we similarly have the following:

$$\max_x \left\{ \max_{t=0,1} \Pr(a_{ij} = 1 | \tilde{a}_{ij} = t, \tilde{m}_{ij} = x) \right\} = \Pr(a_{ij} = 1 | \tilde{a}_{ij} = 0, \tilde{m}_{ij} = x_0),$$

$$\max_{t=0,1} \Pr(a_{ij} = 1 | \tilde{a}_{ij} = t) = \Pr(a_{ij} = 1 | \tilde{a}_{ij} = 0).$$

In this case, $\tau_r(i,j)$ is a decreasing function of k, and the maximum is also $\frac{1-\tilde{\rho}_{\max}}{1-r}$ when $k = (1-r)m$. Therefore, $k_{\min}$ exists if and only if $\epsilon \leq \frac{1-\tilde{\rho}_{\max}}{1-r}$, and $k_{\min} < (1-r)m$. Then, $\tau_r(i,j)$ is given by (3.26). Solving the inequality $\tau_r(i,j) \geq \epsilon$, we have that

$$k \geq \frac{[(1-r)\epsilon\tilde{\rho}_{\max} - r(1 - \tilde{\rho}_{\max})]m}{\epsilon(\tilde{\rho}_{\max} - r)}.$$

However, $\tilde{\rho}_{\max} = \max_x \rho(\widetilde{S}_x)$ varies from time to time due to the perturbation, and data owner can substitute it with the true maximum value $\rho_{\max} = \max_x \rho(S_x)$, then we get the result. □

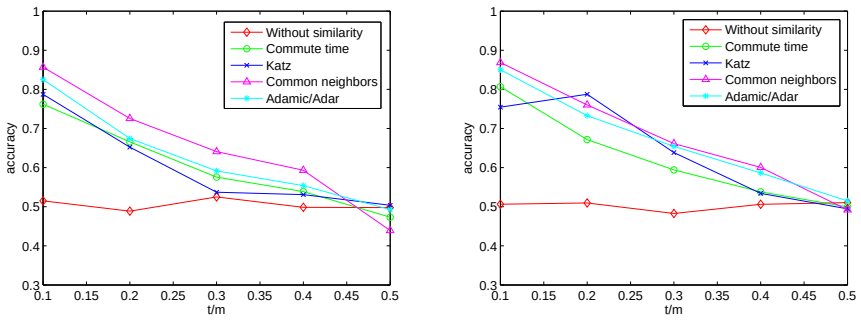
3.2.4 Empirical Evaluation

We use four network data sets (*polbooks*, *Enron*, *E-mail*, *polblogs*) in our evaluation. For each graph G, we randomly add k false edges and delete k true edges. We set

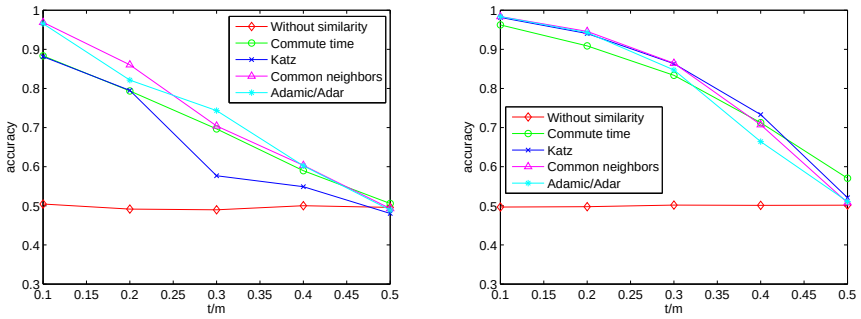
$k = 0.3m, 0.5m, 0.7m$ in our work. We apply four similarity measures (Common neighbors, Katz, Adamic/Adar, Commute time) to predict top-t candidate links. The prediction performance is evaluated by the precision of the top-t predicted links. We vary t values from $0.1m$ to $0.5m$ for all four data sets.

For each t, we calculate the precision of prediction links with different similarity measures. We also calculate the precision of prediction links using the posterior belief without exploiting the similarity measure. Figure 3.5 plots our results on four data sets. We can observe that for all four data sets we can achieve very high accuracy (greater than 0.8) by using our enhanced posterior belief for a subset (top $0.1m$) of released links, which indicates severe privacy disclosures for those sensitive links. We can also see that our enhanced posterior belief achieves higher precisions than the previous posterior belief without exploiting similarity measures for most links ($0.5m$) with high similarity measure values, indicating that the network topology does indeed contain latent information from which to infer interactions. From Figure 3.5, we can also observe that we achieve different precisions using different similarity measures: one measure which achieves the highest precision for one data set is not necessarily the one for another data set. It is of great significance to explore what similarity measures can be exploited by attackers to achieve the highest privacy disclosure for a given social network. We will investigate this in our future work.

In the next experiment, we vary the noise magnitude k from $0.3m$ to $0.7m$. Table 3.2 shows the precisions of top t predictions using different similarity measures on four networks. We can see that for every noise magnitude, predictions that utilize similarity measures achieve a higher accuracy than those without exploiting similarity measures. We can also observe that, for any t, the precision decreases as noise magnitude k increases. This is intuitively reasonable, for large noises can greatly reduce the correlation between the similarity measures and existences of links, and thus decrease the prediction precision. We would point out that $k = 0.7m$ corresponds to



(a) *polbooks* $G(105, 441)$ (b) *Enron* $G(151, 1377)$



(c) *E-mail* $G(1133, 5451)$ (d) *polblogs* $G(1222, 16714)$

Figure 3.5: Precision of top t predictions by the posterior belief w/o similarity measures for four data sets, $k = 0.5m$

a large randomization (i.e., 70% original links have been removed). The posterior belief without exploiting similarity measures, $\Pr(a_{ij} = 1|\tilde{a}_{ij} = 1)$ is only 0.3. However, the posterior belief with exploiting similarity measures is significantly improved. For example, the precision of top $0.1m$ predictions using common neighbors is 0.87 for *polblogs* data.

3.3 Summary

In this chapter, we conduct privacy analysis for *Rand Add/Del* and *Rand Switch* procedures. We derive the attacker's prior belief (without the released graph) and posterior belief (with the observation of an existing or non-existing link) on the existence of a sensitive link. We derive the minimal randomization magnitude needed for *Rand Add/Del* and *Rand Switch* procedures to preserve privacy to a given level.

We also conduct theoretical analysis on the attacking model in which the attacker

exploits node proximity measures to enhance his posterior belief on sensitive links. Our empirical evaluations show that, by exploiting nodes' similarity measures, the attacker can significantly increase his confidence on the existence of a sensitive link between two nodes with high similarity value.

Table 3.2: Precision of top t predictions by the posterior belief w/o similarity measures for four data sets, $k = 0.3m, 0.5m, 0.7m$

(a) Without similarity measures

	polbooks			*Enron*			*E-mail*			*polblogs*		
k:	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$
t: $0.1m$	0.69	0.52	0.28	0.70	0.51	0.30	0.71	0.50	0.30	0.69	0.49	0.29
$0.2m$	0.70	0.49	0.33	0.70	0.51	0.30	0.69	0.49	0.30	0.70	0.49	0.29
$0.3m$	0.69	0.53	0.30	0.71	0.48	0.30	0.70	0.49	0.31	0.69	0.50	0.30
$0.4m$	0.71	0.50	0.30	0.70	0.51	0.28	0.70	0.50	0.30	0.71	0.50	0.29
$0.5m$	0.72	0.50	0.28	0.69	0.51	0.31	0.70	0.50	0.29	0.70	0.51	0.30

(b) Commute time

	polbooks			*Enron*			*E-mail*			*polblogs*		
k:	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$
t: $0.1m$	0.93	0.76	0.39	0.93	0.81	0.42	0.94	0.88	0.68	0.98	0.96	0.87
$0.2m$	0.85	0.67	0.36	0.86	0.67	0.41	0.90	0.79	0.48	0.96	0.91	0.69
$0.3m$	0.82	0.58	0.39	0.81	0.59	0.39	0.88	0.70	0.36	0.95	0.83	0.48
$0.4m$	0.74	0.54	0.36	0.78	0.54	0.32	0.83	0.59	0.33	0.90	0.71	0.33
$0.5m$	0.70	0.47	0.30	0.72	0.50	0.28	0.76	0.51	0.29	0.84	0.57	0.23

(c) Katz

	polbooks			*Enron*			*E-mail*			*polblogs*		
k:	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$
t: $0.1m$	0.94	0.79	0.59	0.95	0.75	0.39	0.97	0.88	0.69	1.00	0.98	0.90
$0.2m$	0.81	0.65	0.42	0.91	0.79	0.36	0.98	0.79	0.53	0.98	0.94	0.73
$0.3m$	0.75	0.54	0.30	0.87	0.64	0.32	0.94	0.58	0.40	0.97	0.86	0.49
$0.4m$	0.76	0.53	0.23	0.80	0.53	0.32	0.88	0.55	0.30	0.94	0.73	0.32
$0.5m$	0.70	0.50	0.27	0.75	0.49	0.30	0.79	0.48	0.24	0.88	0.52	0.20

(d) Common neighbors

	polbooks			*Enron*			*E-mail*			*polblogs*		
k:	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$
t: $0.1m$	0.97	0.85	0.45	0.97	0.86	0.41	0.99	0.96	0.70	0.99	0.98	0.87
$0.2m$	0.94	0.72	0.35	0.96	0.76	0.34	0.98	0.86	0.49	0.98	0.94	0.58
$0.3m$	0.90	0.64	0.33	0.93	0.66	0.32	0.96	0.70	0.44	0.97	0.86	0.39
$0.4m$	0.84	0.59	0.26	0.89	0.60	0.31	0.91	0.60	0.34	0.95	0.70	0.26
$0.5m$	0.82	0.43	0.28	0.83	0.49	0.28	0.82	0.49	0.27	0.90	0.50	0.22

(e) Adamic/Adar

	polbooks			*Enron*			*E-mail*			*polblogs*		
k:	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$	$0.3m$	$0.5m$	$0.7m$
t: $0.1m$	0.98	0.83	0.43	0.98	0.85	0.42	1.00	0.97	0.67	1.00	0.98	0.86
$0.2m$	0.94	0.67	0.37	0.96	0.73	0.36	0.99	0.82	0.54	0.99	0.94	0.57
$0.3m$	0.90	0.59	0.33	0.93	0.65	0.31	0.95	0.74	0.45	0.97	0.85	0.41
$0.4m$	0.83	0.55	0.34	0.89	0.59	0.29	0.90	0.60	0.34	0.94	0.66	0.27
$0.5m$	0.81	0.49	0.29	0.84	0.51	0.28	0.84	0.49	0.28	0.91	0.51	0.23

CHAPTER 4: IDENTITY DISCLOSURE ANALYSIS

The link disclosure corresponds to the scenario where the sensitive relationship between two individuals is disclosed. The identity disclosure corresponds to the scenario where the identity of an individual who is associated with a node is revealed. In this chapter, we assume all individuals (nodes) and relationships (links) among them are sensitive. To prevent identity disclosures, one natural approach is to publish a node-anonymized version of the network that permits useful analysis without disclosing the identity of the individuals represented by the nodes. However, as pointed out in [8, 46], this simple technique of anonymizing graphs by replacing the identifying information of the nodes with random ID's does not guarantee identity/link privacy since adversaries may potentially construct a highly distinguishable subgraph with edges to a set of targeted nodes, and then to re-identify the subgraph and consequently the targets in the released anonymized network.

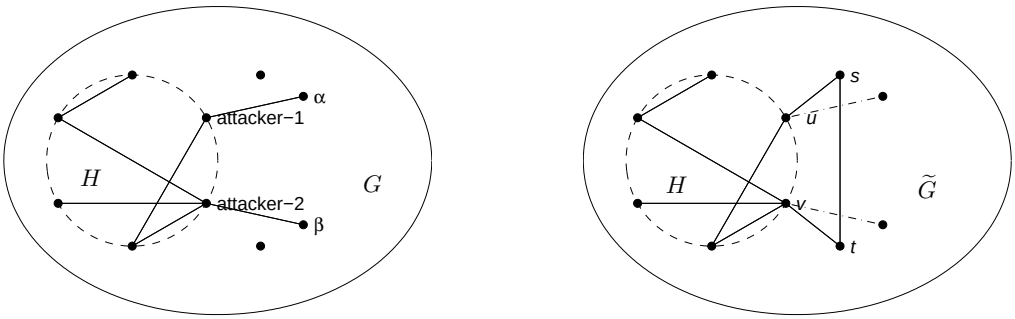
Adversaries usually rely on background knowledge in order to de-anonymize nodes and learn the link relations between de-anonymized individuals from the released perturbed graph. It is challenging to model all types of background knowledge of adversaries in the scenario of publishing social networks with privacy preservation. In [114], the authors listed several types of background knowledge: attributes of vertices, vertex degrees, specific link relationships between some target individuals, neighborhoods of some target individuals, embedded subgraphs, graph metrics (e.g., betweenness, closeness, centrality). We first focus on one most widely used type of background knowledge, *vertex degree* and quantify both identity disclosure and link disclosure when adversaries know the degrees of target individuals, leaving other other

types of background knowledge for future work.

Let Ω denote the set of all individual identifiers in the network: $\Omega = \{Alice, Bob, \ldots, Zack\}$, and let $\psi(\cdot)$ be the mapping from the individual identifier to the node random id in the anonymized graph: for any $\alpha \in \Omega$, $\psi(\alpha)$ is the node index of the individual α, and $\psi^{-1}(i)$ is the identity of node i. One natural question for data owners is, compared to not releasing the graph, to what extent releasing an anonymized/randomized graph $\tilde{G}$ jeopardizes the privacy.

Resilience to Structural Attacks. Recall that in both *active attacks* and *passive attacks* [8], the adversary needs to construct a highly distinguishable subgraph H with edges to a set of target nodes, and then to re-identify the subgraph and consequently the targets in the released anonymized network. As shown in Figure 4.1(a), attackers form an subgraph H in the original graph G, and attacker 1 and 2 send links to the target individuals α and β. After randomization using either *Rand Add/Del* or *Rand Switch*, the structure of subgraph H as well G is changed. The re-identifiability of the subgraph H from the randomized released graph $\tilde{G}$ may significantly decrease when the magnitude of perturbation is medium or large. Even if the subgraph H can still be distinguished, as shown in Figure 4.1(b), link (u, s) and (v, t) in $\tilde{G}$ can be false links. Hence nodes s and t do not correspond to target individuals α and β. Furthermore (e.g. in the released graph with unchanged node identifier information), even individuals α and β have been identified, the observed link between α and β can still be a false link. Hence, the link privacy can still be protected. In summary, it is more difficult for the adversary to breach the identity privacy and link privacy.

Similarly for structural queries [46], because of randomization, the adversary cannot simply exclude from those nodes that do not match the structural properties of the target. Instead, the adversary needs to consider the set of all possible graphs implied by $\tilde{G}$ and k. Informally, this set contains any graph G_p that could result in $\tilde{G}$ under k perturbations from G_p, and the size of the set is $\binom{m}{k}\binom{\binom{n}{2}-m}{k}$. The candidate set of



(a) the original graph (b) the released graph

Figure 4.1: Resilient to subgraph attacks

a target node includes every node y if it is a candidate in some possible graph. The probability associated with a candidate y is the probability of choosing a possible graph in which y is a candidate. The computation is equivalent to compute a query answer over a probabilistic database and is likely to be intractable.

Disclosure Risk Measures. To quantify disclosure risk, we define two risk measures: prior risk measure $r(\omega)$ is defined as the adversary's prior confidence on the event ω without the released graph $\tilde{G}$; and the posterior risk measure $r(\omega|\tilde{G})$ is defined as the adversary's posterior confidence given the released graph $\tilde{G}$.

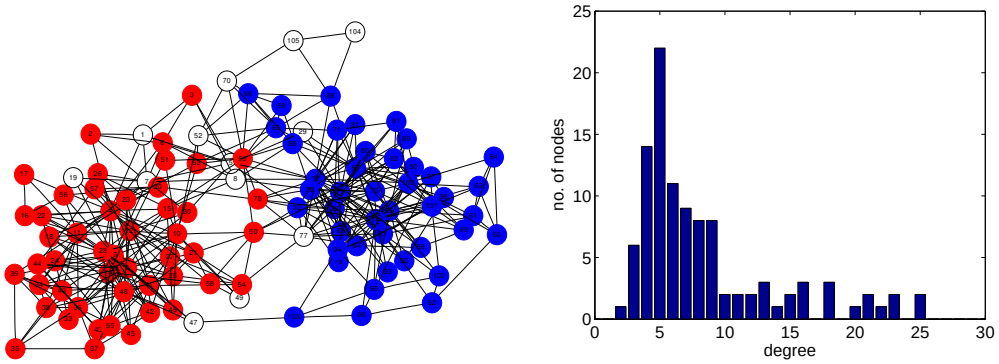
For identity disclosure, we assume the adversary has vertex degree background knowledge, i.e., the target individual's degree is known to adversaries. To make the notation concise, we use d_α to denote the degree of individual α. We use $r(\alpha)$ to denote the adversary's prior confidence on identification of the target individual α. Correspondingly, we use $r(\alpha|d_\alpha, \tilde{G})$ to denote the posterior risk of individual α given the released randomized graph $\tilde{G}$ and the degree of the target individual α (i.e., vertex degree background knowledge). We present our quantification results in Section 4.1.1.

For link disclosure, adversaries need to first identity target individual nodes (incorporating the vertex degree background knowledge, d_α, d_β, with the released graph $\tilde{G}$) and then compute the posteriori belief of existence of the sensitive link (α, β). We use

$R(a_{\alpha\beta})$ and $R(a_{\alpha\beta}|d_\alpha, d_\beta, \tilde{G})$ to denote the prior risk and posterior risk respectively. We present our results in Section 4.1.2. In Section 4.2, we compare *Rand Add/Del* with the K-degree approach [63]. Some results in this chapter are also reported in [107].

4.1 Disclosure Analysis in *Rand Add/Del*

Throughout this section, we illustrate our theoretical results using empirical evaluations on *polbooks* network. Figure 4.2(b) shows the histogram of its degree sequence. For example, there are 22 nodes with degree 5 and one node with degree 20. In the remainder of this section, we use one node (random id 15, identifier label "Breakdown") with degree 5 and the node (random id 30, identifier label "The Price of Loyalty") with degree 20 to illustrate our results.



(a) Network topology (b) Degree distribution

Figure 4.2: The politics book network and the histogram of its degree sequence.

4.1.1 Identity Disclosure

In this section, we focus on identity disclosure in the randomized graph. We study the adversary's strategy and then quantify identity disclosure. We assume that the adversary has vertex degree background knowledge, i.e., the degree of the target individual is known. The adversary needs to take a guess on the mapping function ψ based on his background knowledge and the released graph $\tilde{G}$. In other words, the adversary wants to re-identify which node is corresponding to the target individual α

using the background knowledge of degree d_α. To re-identify α in the node set, the adversary can utilize the randomized degree sequence $\tilde{\boldsymbol{d}} = (\tilde{d}_1, \tilde{d}_1, \ldots, \tilde{d}_n)$. Hence, we can write the posterior risk measure $r(\alpha|d_\alpha, \tilde{G})$ as $r(\alpha|d_\alpha, \tilde{\boldsymbol{d}})$. Let $\hat{\psi}(\cdot)$ denote the adversary's guess of the mapping.

Without the released randomized graph, the background knowledge (such as the true degree of a target individual) cannot be used to enhance the adversary's confidence on the identity mapping. Hence, the prior risk measure $r(\alpha|d_\alpha) = \frac{1}{n}$. Next we deduct the posterior risk measure $r(\alpha|d_\alpha, \tilde{\boldsymbol{d}})$.

Recall that, in *Rand Add/Del* scheme, each true edge can remain in the graph with a probability $p_{11} = \frac{m-k}{m}$, and each non-existing link can be added with a probability $p_{10} = \frac{k}{N-m}$, where $N = \binom{n}{2}$. Let d_i and $\tilde{d}_i$ denote the degree of node i in the G and $\tilde{G}$ graph respectively, and $\hat{d}_i$ is the adversary's estimator of d_i.

Lemma 4.1 shows the calculation of $\Pr(\tilde{d}_i = x|d_i)$, i.e., the probability of a node's degree $\tilde{d}_i$ after randomization given its original degree d_i.

Lemma 4.1: The distribution of $\tilde{d}_i$ is given by

$$\Pr(\tilde{d}_i = x|d_i) = \sum_{t=0}^{x} B(t; d_i, p_{11}) B(x - t; n - 1 - d_i, p_{10}), \tag{4.1}$$

where $B(t; n, p)$ denotes the probability mass function of the binomial distribution with parameter n and p. The expectation and variance of $\tilde{d}_i$ are given by:

$$\mathbf{E}(\tilde{d}_i) = p_{11} d_i + p_{10}(n - 1 - d_i), \tag{4.2}$$

$$\mathbf{V}(\tilde{d}_i) = d_i p_{11}(1 - p_{11}) + (n - 1 - d_i) p_{10}(1 - p_{10}). \tag{4.3}$$

Proof. Let d_i^+ denote the remaining true edges after *Add/Del* process, and d_i^- denote the added links by the process. Since each existing or non-existing link are processed independently, d_i^+ and d_i^- follow the binomial distributions $B(d_i, p_{11})$ and $B(n - 1 -$

$d_i, p_{10})$ respectively:

$$\Pr(d_i^+ = t|d_i) = B(t; d_i, p_{11}) = \binom{d_i}{t} p_{11}^t (1-p_{11})^{d_i - t}. \tag{4.4}$$

$$\Pr(d_i^- = t|d_i) = B(t; n-1-d_i, p_{10}) = \binom{n-1-\tilde{d}_i}{t} p_{10}^t (1-p_{10})^{n-1-\tilde{d}_i - t}. \tag{4.5}$$

Since $\tilde{d}_i = d_i^+ + d_i^-$, the distribution of $\tilde{d}_i$ is just the convolution of (4.4) and (4.5) and we get (4.1). Note the d_i^+ and d_i^- are independent, then

$$\mathbf{E}(\tilde{d}_i) = \mathbf{E}(d_i^+) + \mathbf{E}(d_i^-) = p_{11} d_i + p_{10}(n-1-d_i),$$
$$\mathbf{V}(\tilde{d}_i) = \mathbf{V}(d_i^+) + \mathbf{V}(d_i^-) = d_i p_{11}(1-p_{11}) + (n-1-d_i) p_{10}(1-p_{10}).$$

□

Rearrange (4.2), we can have the following result:

Lemma 4.2: Given a randomized graph, the moment estimator (ME) of d_i is given by:

$$\hat{d}_i = \frac{\tilde{d}_i - p_{10}(n-1)}{p_{11} - p_{10}}, \tag{4.6}$$

and $\hat{d}_i$ is the unbiased estimator of d_i.

The unbias property is straightforward from (4.2).

By combining Lemma 4.1 and Lemma 4.2, we can calculate the posterior probability $\Pr(d_\alpha|\tilde{d}_i)$ (i.e., the likelihood of the observed node i having the degree d_α in the original graph).

Lemma 4.3: In the randomized graph $\tilde{G}$, the adversary observes a node i with degree $\tilde{d}_i$, then the adversary's confidence on $d_i = x$ is given by

$$\Pr(d_i = x|\tilde{d}_i) = \frac{\Pr(\tilde{d}_i|d_i = x)\Pr(d_i = x)}{\sum_{d=0}^{n-1} \Pr(\tilde{d}_i|d = x)\Pr(d = x)}. \tag{4.7}$$

When the original degree distribution is unavailable to the adversary, the estimated degree sequence from (4.6) can be applied instead.

Lemma 4.3 is a direct result from Bayes' theorem.

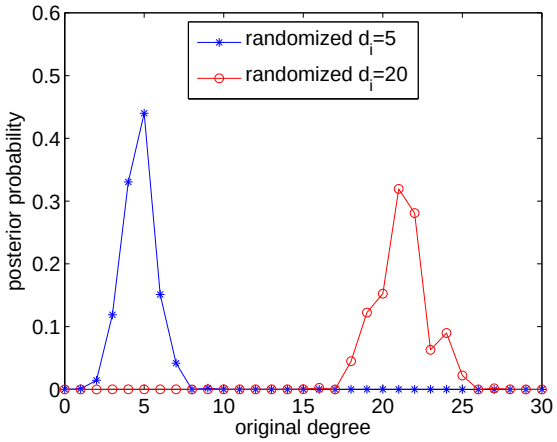


Figure 4.3: Values of $\Pr(d_i|\tilde{d}_i = 5)$ and $\Pr(d_i|\tilde{d}_i = 20)$ after applying *Rand Add/Del* on polbooks network (k=10%m),

Figure 4.3 shows values of two posterior probabilities: $\Pr(d_i|\tilde{d}_i = 5)$ and $\Pr(d_i|\tilde{d}_i = 20)$. Generally speaking, the distribution of $\Pr(d_i|\tilde{d}_i)$ is not symmetric, and it skews to the side with larger degree frequency. In Figure 4.3, for a node with $\tilde{d}_i = 20$,

$$\Pr(d_i = 21|\tilde{d}_i = 20) > \Pr(d_i = 20|\tilde{d}_i = 20) > \Pr(d_i = 19|\tilde{d}_i = 20),$$

this is because the adversary can estimate that, in the original graph $\Pr(d_i = 21) > \Pr(d_i = 20) > \Pr(d_i = 19)$, and Lemma 4.3 incorporates this information in the calculation. We can also observe that the posterior probability that the original degree value d_i is far away from the observed value $\tilde{d}_i$ tends to be zero. In other words, it is very unlikely that a node's degree has a significant change after perturbation.

Recall our node identification problem is that given the true degree d_α of a target individual α, the adversary aims to discover which node in the randomized graph corresponds to individual α. To the adversary, every node in the randomized graph is possible with probability $\Pr(d_\alpha|\tilde{d}_i)$.

Given a list of posterior probabilities $\Pr(d_\alpha|\tilde{d}_i)$ calculated using Lemma 4.3, the

adversary can make the following probabilistic decision:

$$\hat{\psi}(\alpha) = i, \text{ with probability } \frac{\Pr(d_i = d_\alpha | \tilde{d}_i)}{\sum_{j=1}^{n} \Pr(d_j = d_\alpha | \tilde{d}_j)}. \tag{4.8}$$

Result 4.1: Assume the node identities are unknown to the adversary. For any individual $\alpha \in \Omega$, the prior risk measure is

$$r(\alpha | d_\alpha) = \frac{1}{n}. \tag{4.9}$$

The posterior risk measure, which equals to the accuracy of the probabilistic decision in (4.8), is then given by:

$$r(\alpha | d_\alpha, \tilde{\boldsymbol{d}}) = \Pr[\hat{\psi}(\alpha) = \psi(\alpha)] = \frac{\Pr(d_\alpha | \tilde{d}_\alpha)}{\sum_{j=1}^{n} \Pr(d_j = d_\alpha | \tilde{d}_j)}. \tag{4.10}$$

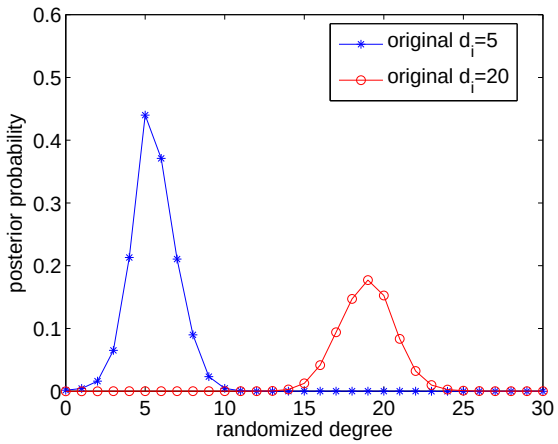


Figure 4.4: Apply *Rand Add/Del* on polbooks network (k=10%m), values of $\Pr(d_i = 5|\tilde{d}_i)$ and $\Pr(d_i = 20|\tilde{d}_i)$ when $\tilde{d}_i$ varies.

In our polbooks example, recall that we select two individuals: α (label "Breakdown") with known degree 5 and β (label "The Price of Loyalty") with known degree 20. From Figure 4.2(b), we can see that there are 22 nodes with degree 5 and only one node with degree 20. Figure 4.4 shows values of $\Pr(d_i = 5|\tilde{d}_i)$ and $\Pr(d_i = 20|\tilde{d}_i)$. Using Equation 4.10, we can easily calculate identity disclosure risk, $r(\alpha|d_\alpha = 5) = 0.135$ and $r(\beta|d_\beta = 20) = 0.024$. It is intuitive to learn that identify disclosure risk given the

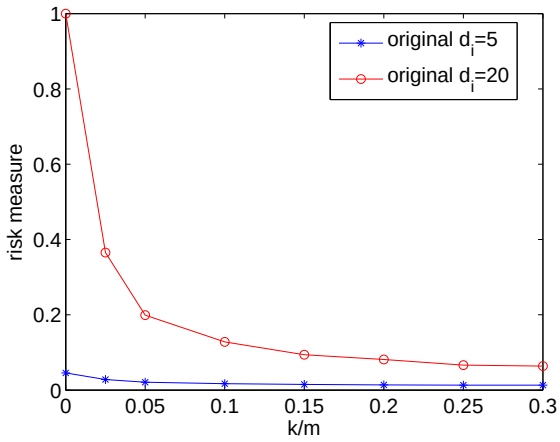


Figure 4.5: $r(\alpha|d_\alpha)$ vs. k after applying *Rand Add/Del* on polbooks network

vertex degree background knowledge is dependent on the degree distribution $\Pr(d_i)$ of the original graph.

Another question is how the identity risk disclosure $r(\alpha|d_\alpha)$ varies with the magnitude of randomization. In Figure 4.5, we show how two identity disclosure risks, $r(\alpha|d_\alpha = 5)$ and $r(\beta|d_\beta = 20)$, vary as the perturbation magnitude (k) changes. We can observe that both identity disclosure risks decrease when k increases. The risk value $r(\alpha|d_\alpha = 5)$ is consistently low even if very few or no perturbations are introduced. This is because there are 22 nodes with the degree 5 in the original graph. However, for $r(\beta|d_\beta = 20)$, we can see that randomization can significantly decrease its disclosure risk: the disclosure risk is 100% when we release the anonymized graph without edge randomization while the disclosure risk decreases 0.39 (0.2) when we apply *Rand Add/Del* with $k = 2.5\%m(5\%m)$.

4.1.2 Link Disclosure

The adversary's goal is to predict whether there is a sensitive link between two target individuals $\alpha, \beta \in \Omega$ by exploiting the released graph and individual degrees d_α, d_β. Given the true degrees of α and β and one released graph $\tilde{G}$, let $R(\alpha, \beta|d_\alpha, d_\beta, \tilde{G})$ denote the posterior risk measure on the link between α and β when the node identities are unknown to the adversary. Similarly, $R(\alpha, \beta)$ is the prior risk measure on

link disclosure.

Lemma 4.4: For *Rand Add/Del* scheme, the prior and posterior risk measures of the existence of a link between node i and j are given by:

$$\Pr(a_{ij} = 1) = \frac{m}{N}; \tag{4.11}$$

$$\Pr(a_{ij} = 1|\tilde{a}_{ij}) = \begin{cases} \frac{m-k}{m}, & \text{if } \tilde{a}_{ij} = 1, \\ \frac{k}{N-m}, & \text{if } \tilde{a}_{ij} = 0. \end{cases} \tag{4.12}$$

where $N = n(n-1)/2$.

Lemma 4.4 shows the link disclosure risks on the simple scenario where node identities are available to adversaries, i.e., for any target individual $\alpha \in \Omega$, the adversary knows its corresponding index, $\psi(\alpha) = i$, in the released randomized graph.

In general, the adversary does not know individuals' corresponding node indices in the released graph. Instead, the adversary may only have vertex degree background knowledge, i.e., the degrees of target individuals are known.

Result 4.2: In the scenario where node identities are unknown to the adversary, for any two individuals $\alpha, \beta \in \Omega$, the prior risk measure and the posterior risk measure given $\tilde{G}$ on the link between α and β after applying *Rand Add/Del* scheme are given by:

$$R(a_{\alpha\beta}) = \frac{m}{n^2 N}, \tag{4.13}$$

$$R(a_{\alpha\beta}|d_\alpha, d_\beta, \tilde{G}) = \frac{m-k}{m} \left(\frac{\Pr(d_\alpha|\tilde{d}_\alpha)}{\sum_{j=1}^{n} \Pr(d_j = d_\alpha|\tilde{d}_j)} \right) \left(\frac{\Pr(d_\beta|\tilde{d}_\beta)}{\sum_{j=1}^{n} \Pr(d_j = d_\beta|\tilde{d}_j)} \right). \tag{4.14}$$

Proof. Since our risk measures are essentially the accuracy of the adversary's predic-

tions, risk measures can be expressed as:

$$R(a_{\alpha\beta}) = r(\alpha)r(\beta)\Pr(a_{ij}=1) \tag{4.15}$$

$$R(a_{\alpha\beta}|d_\alpha, d_\beta, \tilde{G}) = r(\alpha|d_\alpha, \tilde{\boldsymbol{d}})r(\beta|d_\beta, \tilde{\boldsymbol{d}})\Pr(a_{ij}=1|\tilde{a}_{ij}). \tag{4.16}$$

Combining (4.11), (4.12), (4.9), and (4.10) into (4.15) and (4.16), we have the result on the link risk for *Rand Add/Del* when node identities are unknown. □

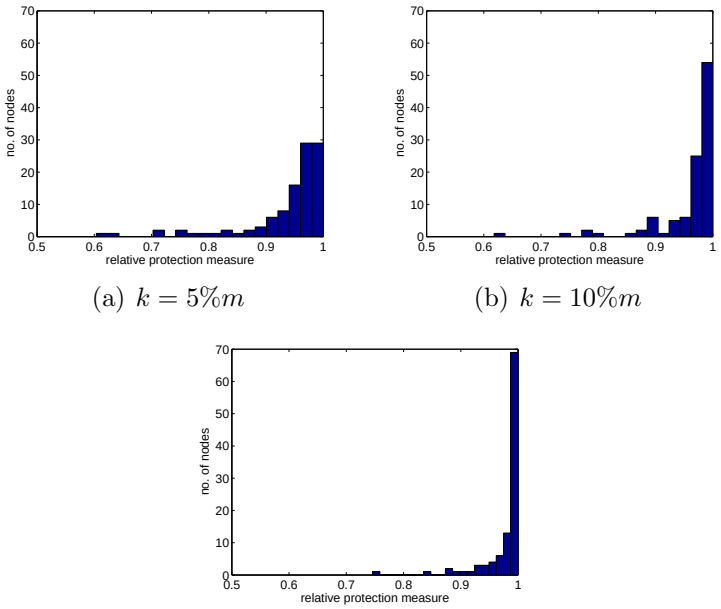
4.1.3 Privacy Protection vs. Perturbation k

From the data owner point of view, we are interested in how much perturbation should be introduced to protect privacy. To measure the privacy protection, we thus further define protection measures: the absolute protection measure $\tau_a(\omega)$ and the relative protection measure of $\tau_r(\omega)$. We are interested in relationships between identity (link) privacy protection and the perturbation magnitude k.

Identity Privacy Protection. The absolute and relative identity protection measures are straightforwardly defined as:

$$\tau_a(\alpha|\tilde{\boldsymbol{d}}) = 1 - r(\alpha|d_\alpha, \tilde{\boldsymbol{d}}), \quad \tau_r(\alpha|\tilde{\boldsymbol{d}}) = \frac{1 - r(\alpha|d_\alpha, \tilde{\boldsymbol{d}})}{1 - 1/n}.$$

Figure 4.6 shows the histogram distributions of relative protection measures $\tau_r(\alpha|\tilde{G})$ under three different perturbation magnitudes ($k = 5\%, 10\%, 20\%m$). We can easily observe that more nodes are protected when k increases. We can also observe that the distribution generally has skewness, which indicates the majority of nodes are resilient to vertex degree background knowledge attack even under a relatively moderate perturbation. The calculation of $r(\alpha|d_\alpha, \tilde{\boldsymbol{d}})$ in (4.10) needs an instance of the randomized graph. In practice, the data owner may expect to determine k before applying *Rand Add/Del* such that the randomized data satisfies some privacy protection threshold. Hence, we should use the expected randomized degree sequence shown in



(a) $k = 5\%m$ (b) $k = 10\%m$

(c) $k = 20\%m$

Figure 4.6: Histogram of $\tau_r(\alpha|\tilde{G})$ for 105 nodes in polbooks network, under *Rand Add/Del* scheme. The skewness of the distribution increases, indicating more nodes are well protected as k increases.

(4.2) to evaluate the protection measure and choose k such that

$$J(k) = \min_{\alpha \in \Omega} \tau_r[\alpha|\, \mathbf{E}(\tilde{\boldsymbol{d}})] \geq 1 - \varepsilon.$$

Link Privacy Protection. Similarly, the link privacy protection measures are shown as:

$$\Gamma_a(a_{\alpha\beta}|\tilde{G}) = 1 - R(a_{\alpha\beta}|d_\alpha, d_\beta, \tilde{G}),$$
$$\Gamma_r(a_{\alpha\beta}|\tilde{G}) = \frac{1 - R(a_{\alpha\beta}|d_\alpha, d_\beta, \tilde{G})}{1 - R(\alpha, \beta)}.$$

Figure 4.7 shows the histogram of $\Gamma_r(a_{\alpha\beta}|\tilde{G})$ for polbooks network after we apply *Rand Add/Del* scheme ($k = 10\%m$). We can see that all Γ_r values are greater than 90%, and most links have their relative protection measure values close to 1, indicating that the protection of *Rand Add/Del* with $k = 10\%m$ almost achieves the same protection

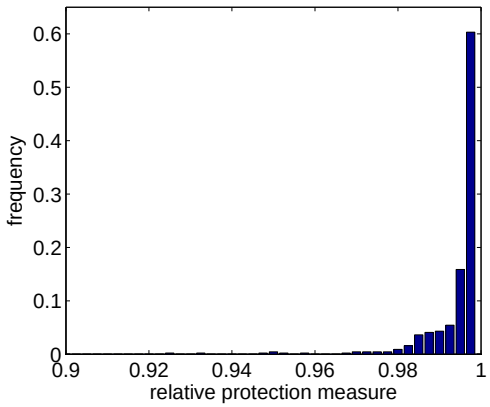


Figure 4.7: Histogram of $\Gamma_r(a_{\alpha\beta})$ for polbooks network,*Rand Add/Del* ($k = 10\%m$)

Table 4.1: Perturbation parameter k that meet the protection requirement for *polbooks* network

$1-\varepsilon$	k for identity protection	k for link protection
0.5	27	8
0.6	32	9
0.7	59	12
0.8	110	16
0.9	257	37

as without a released graph. Formally, we expect to choose a k such that

$$J(k) := \min_{a_{\alpha\beta}} \Gamma_r(a_{\alpha\beta}|\tilde{G}) \geq 1-\varepsilon. \tag{4.17}$$

Note that we use $\mathbf{E}(\tilde{\boldsymbol{d}})$ and $\tilde{a}_{\alpha\beta} = 1$ in calculating (4.17).

Table 4.1 shows the minimal k that meets the identity (link) protection requirement in (4.17) for polbooks network. We can see that *Rand Add/Del* scheme can generally achieve both identity protection and link protection with small or medium perturbations, e.g., $k = 59$ (or $k = 12$) for the relative protection threshold 0.7 of identity privacy (or link privacy). We can also observe that *Rand Add/Del* needs much fewer perturbations to achieve the link protection than the identity protection. This is because the adversary needs to identify the target two individuals before predicting the existence of a link between these two individuals.

4.2 Comparison with K-degree Generalization Scheme

In this section, we compare the *Rand Add/Del* scheme with the representative generalization based scheme (*K-degree*) in terms of the tradeoff between privacy protection and utility loss. Since the K-degree scheme is designed to protect the re-identification of individuals, we focus on identity privacy protection in empirical evaluations.

4.2.1 Identity Privacy Protection vs. Utility Loss

Graph Characteristics vs. Utility. To achieve utility, we expect the released randomized graph should also keep structural properties not much changed or those properties can be reconstructed from the randomized graph. In this section, we use the following representative real space features: harmonic mean of the shortest distance h, modularity Q, and transitivity C. We also consider the two spectral features: the eigenvalues of the adjacency matrix A λ_1, and the second eigenvalue of the Laplacian matrix μ_2.

Table 4.2 shows our empirical evaluations on three networks: Polbooks, Polblogs, and Enron. For each network, we vary K from 2 to 10 and apply both *Rand Add/Del* and *K-degree Generalization* schemes. For *Rand Add/Del*, we use the absolute identity protection measure, $\tau_a(\alpha|\tilde{\boldsymbol{d}}) \geq 1-1/K$, to determine the perturbation magnitude k and then generate a randomized network using k. We can observe that both *Rand Add/Del* and *K-degree* schemes generally decrease structural properties. For example, both Q (indicating the goodness of the community structure) and μ_2 (showing how good the communities separate, with smaller values corresponding to better community structures) increase along K, which indicates the goodness of the community structure is affected due to edge modification. We can also observe from Table 4.2 that *K-degree* scheme generally better preserves structural features than *Rand Add/Del*. This is because that *K-degree* scheme examines the degree sequence of nodes and

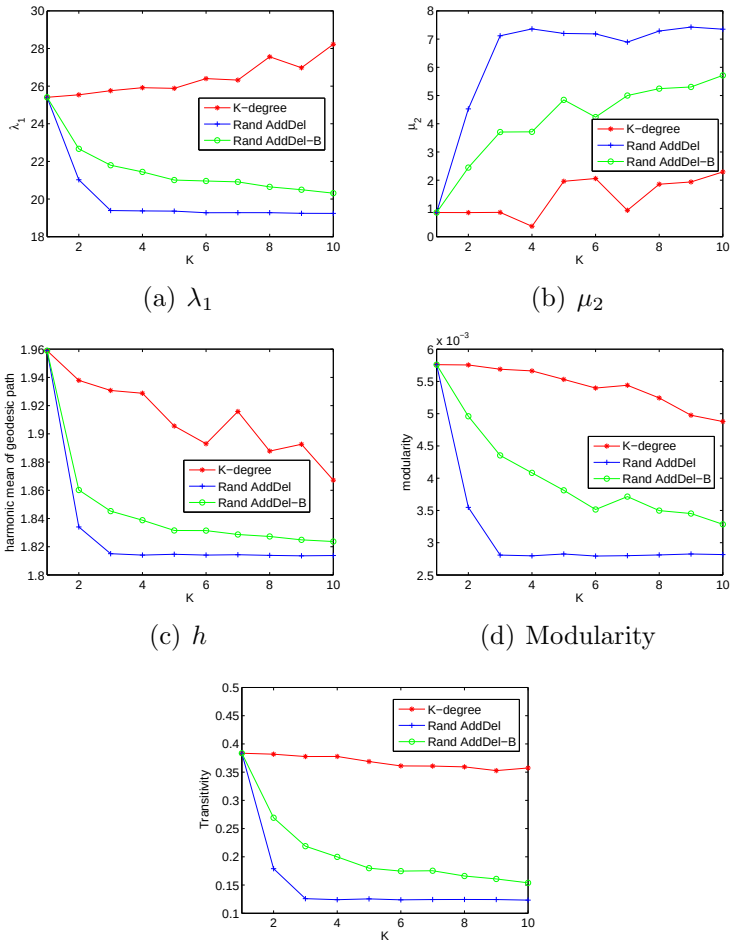
chooses a subset of nodes (that violates the *K-degree* anonymity property) for edge modification while *Rand Add/Del* scheme treats all nodes (edges) equally during randomization. We expect that reconstruction methods can be designed for the purely randomized graph so features derived from the reconstructed graph (rather than directly from the released randomized graph) can be more accurate. It is our belief that it is very hard, if not impossible, to figure out reconstruction methods on the released data randomized using *K-degree* scheme. We will investigate reconstruction methods in our future work.

4.2.2 Further Improvement

Since *Rand Add/Del* randomly adds and deletes edges, a large number of perturbations are applied to those nodes in low risks. As a result, we sacrifice graph utility without further improving identity protection. One natural idea is that we can divide the graph into several blocks according to the degree sequence and apply *Rand Add/Del* separately to each block using different randomization parameters k.

In many real-world networks, we have fewer nodes with high degrees while more nodes with low degrees. By simply partitioning the graph into blocks according to the degree sequence, we expect to introduce fewer perturbations (with better utility preservation) to achieve the same privacy protection. For each block b, we say an existing (or non-existing) link (i, j) is in block b if node i or j is in the block. Let n_b be the number of nodes and and m_b be the number of links in block b. We randomly add and delete k_b links, then each existing link remains in the randomized graph with probability $p_{11}^{(b)} = 1 - \frac{k_b}{m_b}$, and each non-existing link is added with probability $p_{10}^{(b)} = \frac{k_b}{N_b - m_b}$ where $N_b = \binom{n_b}{2} - n_b(n - n_b)$. We can use the same methodologies in calculating the identity/link risks except for replacing the overall p_{11} and p_{10} with $p_{11}^{(b)}$ and $p_{10}^{(b)}$. We call this method blockwise random add/delete strategy, or simply *Rand Add/Del-B* for short.

Figure 4.8 shows preliminary results of *Rand Add/Del-B* on Enron network. In



(a) λ_1 (b) μ_2

(c) h (d) Modularity

(e) Transitivity

Figure 4.8: Identity Protection K vs. Feature Change on Enron data

this experiment, we simply divide the graph into two blocks: nodes with degree greater than 30 are in the first block while the rest nodes with high degree frequency values are in the second block. We can observe from Figure 4.8 that this simple strategy can better preserve graph features than *Rand Add/Del*. We expect to achieve even better utility preservation when we have better block partitions (e.g., using histogram partition algorithms). As we discussed previously, we will also investigate reconstruction methods on the released data using *Rand Add/Del-B* scheme.

4.3 Summary

In this chapter, we quantify both identity disclosure and link disclosure risks associated with *Rand Add/Del* and *Rand Switch* procedures based on one most widely used type of background knowledge, vertex degree. We compare our *Rand Add/Del* with another representative edge modification scheme K-degree generalization scheme proposed in [63] in terms of the tradeoff between disclosure risks and utility loss. Our empirical results show that generalized graph via the K-degree generalization scheme generally better preserves structural features than the randomized graph via the *Rand Add/Del*. It is also worth pointing out that the K-degree generalization scheme is designed to only protect the re-identification of individuals while the *Rand Add/Del* can provide both identity and link privacy protection.

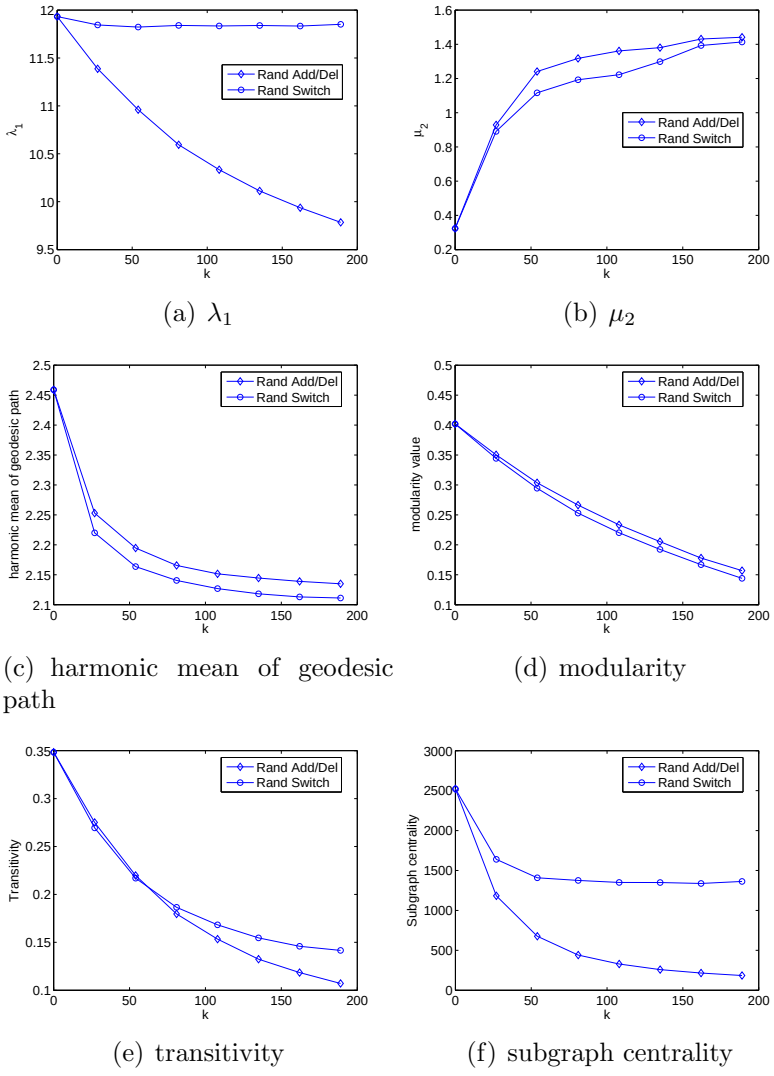
Table 4.2: Identity protection K vs. Feature Changes between *Rand Add/Del* scheme (denoted as Rand) and *K-degree Generalization* scheme (denoted as K-deg); Rows with $K = 1$ show the feature values of the original networks

K	λ_1		μ_2		h		Q		C	
	Rand	K-deg	Rand	K-deg	Rand	K-deg	Rand	K-deg	Rand	K-deg
polbooks										
1	11.93		0.32		2.45		0.40		0.34	
2	11.64	12.00	0.61	0.43	2.31	2.35	0.37	0.39	0.30	0.33
3	11.51	12.05	0.79	0.45	2.28	2.32	0.36	0.39	0.29	0.33
4	11.04	12.11	1.16	0.60	2.20	2.28	0.31	0.38	0.22	0.32
5	10.50	12.22	1.43	0.60	2.16	2.28	0.26	0.38	0.17	0.33
6	10.33	12.30	1.16	0.79	2.16	2.23	0.24	0.36	0.15	0.30
7	10.15	12.31	1.41	0.63	2.14	2.27	0.21	0.37	0.13	0.31
8	9.83	12.64	1.53	0.65	2.13	2.26	0.15	0.37	0.10	0.32
9	9.72	12.72	1.43	0.97	2.13	2.20	0.14	0.34	0.10	0.29
10	9.75	12.85	1.61	0.88	2.13	2.19	0.14	0.35	0.1	0.30
polblogs										
1	74.08		0.168		2.506		0.405		0.226	
2	30.19	74.89	9.30	0.168	2.35	2.500	0.067	0.402	0.027	0.225
3	28.55	74.50	10.58	0.168	2.35	2.484	0.024	0.401	0.022	0.223
4	28.50	75.16	10.72	0.168	2.35	2.494	0.020	0.401	0.022	0.224
5	28.49	75.10	11.11	0.168	2.35	2.475	0.018	0.396	0.022	0.221
6	28.47	76.32	10.86	0.168	2.35	2.469	0.019	0.394	0.022	0.222
7	28.46	75.82	11.09	0.168	2.35	2.461	0.018	0.395	0.022	0.22
8	28.46	76.67	11.14	0.168	2.35	2.462	0.016	0.389	0.022	0.219
9	28.46	77.42	10.68	0.168	2.35	2.486	0.019	0.387	0.022	0.221
10	28.46	78.42	10.72	0.168	2.35	2.458	0.015	0.385	0.022	0.221
Enron										
1	17.83		0.80		2.278		0.0074		0.344	
2	13.90	18.16	1.60	0.84	2.096	2.25	0.0046	0.0072	0.127	0.33
3	12.69	18.29	3.20	0.86	2.079	2.24	0.0037	0.0072	0.081	0.33
4	12.65	18.45	2.99	1.00	2.079	2.17	0.0037	0.0069	0.078	0.31
5	12.66	19.31	3.04	0.85	2.078	2.17	0.0037	0.0066	0.080	0.31
6	12.63	19.41	2.89	0.84	2.078	2.19	0.0037	0.0065	0.078	0.31
7	12.60	20.04	3.04	0.82	2.078	2.19	0.0037	0.0069	0.078	0.31
8	12.60	19.92	3.11	0.82	2.079	2.12	0.0037	0.0063	0.079	0.29
9	12.61	20.42	2.84	1.45	2.079	2.13	0.0037	0.0061	0.079	0.30
10	12.62	21.39	2.96	0.98	2.077	2.05	0.0037	0.0058	0.078	0.29

CHAPTER 5: FEATURE PRESERVING RANDOMIZATION

Edge randomization may significantly affect the utility of the released randomized graph. To preserve utility, certain aggregate characteristics (a.k.a., feature) of the original graph should remain basically unchanged or at least some properties can be reconstructed from the randomized graph. However, as we show below, many topological features are lost due to *Rand Add/Del* or *Rand Switch*.

Figure 5.1 shows the trend of the change of graph characteristics (including two spectral, λ_1, μ_2 and four real, harmonic mean of geodesic path, modularity, transitivity, and subgraph centrality) as *Rand Add/Del* and *Rand Switch* perturbation strategies are applied to graph *polbooks*. We can observe that, except the λ_1 in *Rand Switch* procedure, the graph features can be greatly changed as the randomization magnitude parameter k increases. For example, μ_2 and modularity measure Q are very different from the original value when k approaches 200, indicating the community structure is not resilient to random perturbation. The harmonic mean of the geodesic path shows the similar trend. This is intuitively reasonable, as average vertex-vertex distance may change sharply when edges across communities are switched with edges within communities. Note that we have 441 edges in this graph, even the medium randomization ($k = 100$) significantly decreases the utility of the released graph. Generally more perturbation can lead to stronger privacy protection, but it also greatly changes many features of the network, decreasing the information utility. For example, network resilience and community structure are of particular importance in epidemiology where removal of vertices or edges in a contact network may correspond to vaccination of individuals against a disease. Then the epidemio-



(a) λ_1 (b) μ_2

(c) harmonic mean of geodesic path (d) modularity

(e) transitivity (f) subgraph centrality

Figure 5.1: Graph characteristic vs. perturbation with varying k for *Rand Add/Del* and *Rand Switch*

logical solution developed from the randomly perturbed graph may not be applicable to the real graph.

In this chapter, we investigate how to perturb graphs without changing much network structural features. In Section 5.1, we develop the spectrum preserving randomization procedures which preserves some eigenvalues of graph matrices. In Section 5.2, we introduce the Markov chain based randomization procedure, which can preserve any graph feature specified by the users. Some of the results in this chapter are also reported in [104, 106].

5.1 Spectrum Preserving Randomization

5.1.1 Theoretical Analysis on Spectral Perturbation

The theory of graph perturbations is concerned primarily with changes in eigenvalues which result from local modifications of a graph such as adding or deleting an edge. In the following, we let A and $\tilde{A}$ be the adjacency matrices of the original graph G and the perturbed graph G' with spectra $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$ respectively.

Lemma 5.1: [25] $\tilde{\lambda}_1 < \lambda_1$ whenever G' is obtained from G by deleting an edge or vertex. Similarly, $\tilde{\lambda}_1 > \lambda_1$ whenever G' is obtained from G by adding an edge or a non-isolated vertex.

Lemma 5.1 shows any proper subgraph of G has smaller index value λ_1 and any supgraph of G has larger index value λ_1. This is also one reason why we only focus on the perturbation strategies that keep the number of edges unchanged. Otherwise, the index of the graph λ_1 may be significantly changed, which will affect many real space graph characteristics.

Theorem 5.1: Weyl's Theorem [49]. Given two $n \times n$ symmetric matrices A and E, assume $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and $\varepsilon_1 \geq \varepsilon_2 \geq \cdots \geq \varepsilon_n$ are their eigenvalues respectively. Let $\tilde{A} = A + E$, and $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$ are its eigenvalues. Then the Weyl's inequalities are

$$\tilde{\lambda}_{i+j-1} \leq \lambda_i + \varepsilon_j \leq \tilde{\lambda}_{i+j-n} \tag{5.1}$$

for $1 \leq i, j, i+j-1, i+j-n \leq n$.

Weyl's theorem states that the eigenvalues of a matrix are perfectly conditioned, i.e., no eigenvalue can move more than the range specified by (5.1).

Some graph features (e.g., the number of vertices n, the number of edges m) remain unchanged after randomization and are assumed to be available to attackers. We also assume that the number of perturbations k is available to both data miners and

attackers. The reason is that k denotes the magnitude of perturbation which may be needed to analyze the perturbed graph by data miners. In this section, we present to what extent the graph spectrum may change with respect to those graph invariants, specifically, k and n for *Rand Add/Del* and k, n and d_i for *Rand Switch* where d_i is the degree of vertex i.

When $k = 1$, we call the perturbation matrix as the elementary perturbation matrix (EPM). Obviously, the perturbation matrix E when $k > 1$ is the sum of EPMs along the perturbation.

For *Rand Add/Del*, we have two different cases. One is that we add the edge (i, p) and delete an existing edge (i, q). In this case, the EPM has the form as below:

$$E_{(i,p,q)} = \tilde{A} - A = \begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \oplus 0_{n-3}. \tag{5.2}$$

Specifically, $e_{ip} = e_{pi} = 1$, and $e_{iq} = e_{qi} = -1$, where e_{ij} denotes the component of E. The other case is that we add the edge (i, j) and then remove one existing edge (p, q) where i, j, p, q are distinct. Then,

$$E_{(i,j,p,q)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \oplus 0_{n-4}. \tag{5.3}$$

Specifically, $e_{ij} = e_{ji} = 1$, and $e_{pq} = e_{qp} = -1$.

For *Rand Switch*, when we switch one pair of edges, (t, w), (u, v) to (t, v) and (u, w),

the EPM is:

$$E_{(t,w,u,v)} = \begin{pmatrix} 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \end{pmatrix} \oplus 0_{n-4} \tag{5.4}$$

Specifically, $e_{tw} = e_{wt} = e_{uv} = e_{vu} = -1$, and $e_{tv} = e_{vt} = e_{uw} = e_{wu} = 1$. We can easily derive $\varepsilon_1 = 2$,$\varepsilon_n = -2$, and $\varepsilon_i = 0$ $(2 \leq i \leq n-1)$.

However, when $k > 1$, it is hard to derive directly the eigenvalues of E based on the released k. In the following, we show our result based on the Gershgorin Circle Theorem [49].

Theorem 5.2: Gershgorin Circle Theorem. For an $n \times n$ matrix A, define $R_i = \sum_{j=1,j\neq i}^{n} |a_{ij}|$. Then each eigenvalue of A must be in at least one of the disks in the complex plane: $C_i(A) = \{z : |z - a_{ii}| \leq R_i\}$.

Result 5.1: Let $\varepsilon_1 \geq \varepsilon_2 \geq \cdots \geq \varepsilon_n$ be the eigenvalues of E. For all $i(1 \leq i \leq n)$, we have

$$\varepsilon_n \leq \left|\lambda_i - \tilde{\lambda}_i\right| \leq \varepsilon_1 \tag{5.5}$$

or more loosely

$$\left|\lambda_i - \tilde{\lambda}_i\right| \leq \|E\|_2 , \tag{5.6}$$

where for *Rand Add/Del*,

$$\|E\|_2 \leq \min\{2k, n-1\}, \tag{5.7}$$

and for *Rand Switch*,

$$\|E\|_2 \leq 2\min\left\{k, \max_i \left(\min\{d_i, n-1-d_i\}\right)\right\} \tag{5.8}$$

Proof. (5.5) and (5.6) can be easily derived from the Weyl's theorem.

Notice that the diagonal elements of E are always 0. Hence,

$$C_i(E) = \{z : |z - e_{ii}| \leq R_i\} = \{z : |z| \leq R_i\} .$$

All these circles are concentric, and all the eigenvalues of A are thus in the circle of the largest radius: $\|E\|_2 \leq \max_i\{R_i\}$. and $R_i = \sum_{j\neq i} |e_{ij}|$ is actually the totaly number of added and deleted edges of vertex i.

Hence, for *Rand Add/Del*, when $k < n/2$, the worst case is that all the perturbations involve the same vertex; when $k \geq n/2$, the worst case happens when a certain vertex is removed all original edges to its neighbors and adds new edges to all the rest vertices. In this case, $\max_i\{R_i\} \leq \min\{2k, n-1\}$, and (5.7) follows.

For *Rand Switch*, if one edge is deleted, there must be an edge added to the same vertex. Therefore

$$\frac{1}{2}R_i \leq \min\{d_i, n-1-d_i\},$$

through which we immediately get

$$\max_i R_i \leq 2\min\left\{k, \max_i\left(\min\{d_i, n-1-d_i\}\right)\right\},$$

and (5.8) follows. □

Actually, the bound given in (5.8) is the loose bound in the worst case. It may not accurately reflect the magnitude of spectrum change. In Section 4, we develop our spectrum preserving randomization approach which can control the change of spectrum during the randomization process. Note that all the above results can be easily extended to the Laplacian matrix with some simple adjustment since $\tilde{L} - L = A - \tilde{A} = -E$.

5.1.2 Spectrum Preserving Randomization

It has been shown that the eigenvalues of a network are intimately connected to many important topological features. For example, The eigenvalues of A encode information about the cycles of a network as well as its diameter. The maximum degree, chromatic number, clique number, and extend of branching in a connected graph are all related to λ_1. In [96], the authors studied how a virus propagates in a

real work and proved that the epidemic threshold for a network is closely related to λ_1. Refer to [84] for more relationships between the spectral and real characteristics of graphs.

Since many graph structures are shown to have strong association with the spectrum, a very nature idea is whether we can figure out a perturbation strategy such that one or some particular eigenvalues will not significantly change. Hence the new strategy is more probable to better preserve structural characteristics without much scarifying the privacy protection.

Table 5.1: Conditions on adjusting λ_1 and μ_2 for *Spctr Add/Del*

Condition	Action
$x_i x_j - x_p x_q > 0$	$\tilde{\lambda}_1 > \lambda_1$
$x_i x_j - x_p x_q < 0$, and $\lambda_1 - \lambda_2 > \frac{x_i^2+x_j^2+x_p^2+x_q^2}{2(x_p x_q - x_i x_j)}$	$\tilde{\lambda}_1 < \lambda_1$
$y_i y_j - y_p y_q > 0$	$\tilde{\mu}_2 < \mu_2$
$y_i y_j - y_p y_q < 0$, and $\mu_3 - \mu_2 > \frac{y_i^2+y_j^2+y_p^2+y_q^2}{2(y_p y_q - y_i y_j)}$	$\tilde{\mu}_2 > \mu_2$

Table 5.2: Conditions on adjusting λ_1 and μ_2 for *Spctr Switch*

Condition	Action
$(x_t - x_u)(x_v - x_w) > 0$	$\tilde{\lambda}_1 > \lambda_1$
$(x_t - x_u)(x_v - x_w) < 0$, and $\lambda_1 - \lambda_2 > \frac{x_t - x_u}{x_w - x_v} + \frac{x_w - x_v}{x_t - x_u}$	$\tilde{\lambda}_1 < \lambda_1$
$(y_t - y_u)(y_v - y_w) > 0$	$\tilde{\mu}_2 < \mu_2$
$(y_t - y_u)(y_v - y_w) < 0$, and $\mu_3 - \mu_2 > \frac{y_t - y_u}{y_w - y_v} + \frac{y_w - y_v}{y_t - y_u}$	$\tilde{\mu}_2 > \mu_2$

From matrix perturbation community, researchers have achieved results on the intermediate eigenvalue problem of the second type, i.e., how to determine E such that the eigenvalue λ_1 of $A + E$ can be greater or less than that of A. Specifically, Cvetkovic et al.[25] gave results on how to increase or decrease λ_1 of the adjacency

matrix by constructing the noise matrix E based on the principal eigenvector values of the adjacency matrix. We list their results in the first two rows of Table 5.1 and Table 5.2. For example, according to row 1 in Table 5.1, if we add edge (i, j) and delete edge (p, q) and $x_i x_j - x_p x_q > 0$ stands, λ_1 necessarily increases. Note that x_i denotes the ith component in the principal eigenvector of λ_1.

In our work, we also need to know whether the eigenvalue μ_2 of the Laplacian matrix L of a particular graph G increases or decreases when an edge is relocated. We derive sufficient conditions on how to adjust μ_2 of the Laplacian matrix for two random strategies *Add/Del* and *Switch*. We summarize our results in the last two rows of Table 5.1. Note that μ_2 is the important eigenvalue of the Laplacian matrix L. We use μ_i and $\tilde{\mu}_i$ to denote the ith smallest eigenvalue of L and $\tilde{L}$ respectively, and $\boldsymbol{u}_2$ denotes the eigenvector of μ_2. y_i is the ith component of $\boldsymbol{u}_2$. Next, we give the proof of the conditions for adjusting μ_2.

Proof. Let $\boldsymbol{u}_i$ and $\tilde{\boldsymbol{u}}_i$ be the eigenvector corresponding to μ_i and $\tilde{\mu}_i$. Consider the minimum problem:

$$\min_{x \in S} \left\{ \boldsymbol{x}^T \tilde{L} \boldsymbol{x} \right\},$$

where $S = \left\{ \boldsymbol{x} : \boldsymbol{x}^T \tilde{\boldsymbol{u}}_1 = 0, \text{and } \|\boldsymbol{x}\|_2 = 1 \right\}$.

Since $\boldsymbol{u}_1 = \tilde{\boldsymbol{u}}_1$, $\boldsymbol{u}_2 \in S$. Then

$$\min_{\boldsymbol{x} \in S} \left\{ \boldsymbol{x}^T \tilde{L} \boldsymbol{x} \right\} \leq \boldsymbol{u}_2^T \tilde{L} \boldsymbol{u}_2 = \mu_2 - \boldsymbol{u}_2^T E \boldsymbol{u}_2$$

On the other hand, take $\boldsymbol{x}$ to be $\tilde{\boldsymbol{u}}_2$, $\tilde{\mu}_2 = \min_{\boldsymbol{x} \in S} \left\{ \boldsymbol{x}^T \tilde{L} \boldsymbol{x} \right\}$, hence $\tilde{\mu}_2 \leq \mu_2 - \boldsymbol{u}_2^T E \boldsymbol{u}_2$. When $\boldsymbol{u}_2^T E \boldsymbol{u}_2 > 0$, $\tilde{\mu}_2 < \mu_2$ always holds. With the concrete form of EPM, in *Add/Del* strategy:

$$\boldsymbol{u}_2^T E \boldsymbol{u}_2 = 2(y_i y_j - y_p y_q),$$

and in *Switch*:

$$\boldsymbol{u}_2^T E \boldsymbol{u}_2 = 2(y_t - y_u)(y_v - y_w).$$

For the rest part of the table, we focus on the Switch strategy. and *Add/Del* strategy can be proved similarly by using the corresponding perturbation matrix E.

Denote $\lambda_i(M)$ for ith eigenvalues of matrix M sorted in non-decreasing order: $\lambda_1(M) \leq \lambda_2(M) \leq \cdots \leq \lambda_n(M)$. We take $t = 1, v = 2, u = 3, w = 4$ without loss of generality. Then, with the second part of the theorem, we have $(y_1 - y_3)(y_2 - y_4) < 0$, and

$$E = \begin{pmatrix} 0 & 1 & 0 & -1 & & \\ 1 & 0 & -1 & 0 & \vdots & \\ 0 & -1 & 0 & 1 & & \\ -1 & 0 & 1 & 0 & & \\ & \cdots & & & 0_{(n-4)\times(n-4)} & \end{pmatrix},$$

Based on Laplacian matrix, we construct our own $\bar{E}$ and $\bar{L}$ needed in the proof: $\bar{E} = (\delta + 2)I - E$, and $\bar{L} = L - (\delta + 2)I$, where $\delta > 0$ is a parameter. Then,

- $\bar{E}$ is positive definite;
- $\lambda_i(\bar{L}) = \lambda_i(L) - (\delta + 2)$, and $\lambda_i(\bar{L})$ and $\lambda_i(L)$ have the same eigenvector;
- $\bar{L}+\bar{E} = L-E = \tilde{L}$, and therefore $\mu_2 = \lambda_2(\tilde{L}) = \lambda_2(\bar{L}+\bar{E}) \geq \lambda_2(\bar{L}+\bar{E}P_2)$ where P_2 is the orthogonal projection onto the subspace spanned by $\{\bar{E}^{-1}\boldsymbol{u}_1, \bar{E}^{-1}\boldsymbol{u}_2\}$. (see [25] for more details).

With the similar deduction outlined in [25], we can calculate $\lambda_2(\bar{L} + \bar{E}P_2)$ and thus get a lower bound of $\tilde{\mu}_2$:

$$\tilde{\mu}_2 \geq \min\{\mu_2 - 2 - \delta + \gamma, \mu_3 - 2 - \delta\}, \tag{5.9}$$

where

$$\gamma = \frac{\delta(2 + \delta)(4 + \delta)}{\delta(\delta + 4) - 2b\delta + 2a} \tag{5.10}$$

and $a = (y_1 + y_2 - y_3 - y_4)^2, b = (y_1 - y_3)(y_4 - y_2) > 0$. γ is an increasing function of

δ with range $(0, \infty)$. We thus can always choose $\delta > 0$ such that $\gamma = \mu_3 - \mu_2$, then we rewrite (5.9) as$\tilde{\mu}_2 \geq \mu_3 - 2 - \delta$.

Next we deduct the condition under which this lower bound is always greater than μ_2, or equivalently the following inequalities and equation always stands:

$$\begin{cases} \tilde{\mu}_2 \geq \mu_3 - 2 - \delta > \mu_2 \\ \gamma = \dfrac{\delta(2+\delta)(4+\delta)}{\delta(\delta+4) - 2b\delta + 2a} \\ \gamma = \mu_3 - \mu_2 \end{cases} \tag{5.11}$$

It is not difficult to show that when $\gamma = \mu_3 - \mu_2 > 2 + \frac{a}{b}$, and (5.11) stands. Since

$$2 + \frac{a}{b} = \frac{(y_1 - y_3)}{(y_4 - y_2)} + \frac{(y_4 - y_2)}{(y_1 - y_3)},$$

when $\mu_3 - \mu_2 > \frac{(y_1 - y_3)}{(y_4 - y_2)} + \frac{(y_4 - y_2)}{(y_1 - y_3)}$, $\tilde{\mu}_2 > \mu_2$ stands. The rest parts of the result are proved. □

Based on the derived conditions, we develop our spectrum preserving approach which can improve the simple edge randomization by considering the change of spectrum in the randomization process. Here we can determine which edges we should add/remove or switch so that we can control the move of target eigenvalues. As a result, real graph characteristics (or graph utility) are expected to be better preserved. We show our *Spctr Switch* algorithm in Algorithm 1.

In Row 2 of Algorithm 1, we only calculate the first one or two eigenvalues of the corresponding graph matrices. It is not necessary or desirable to calculate the entire eigen-decomposition. Note that calculation of the eigenvectors of an $n \times n$ matrix takes in general a number of operations $O(n^3)$. An efficient Lanczos method [38] can be applied to find the second eigenvector of a sparse matrix with $m/(\lambda_3 - \lambda_2)$, where m is the number of edges in the graph. Row 4 gives the loop condition of repeated switch operations (we will discuss details on $J_2(k)$ and the input privacy protection threshold ε in Section 5.2). Rows from 6 to 11 present how to switch

Algorithm 1 Spectrum Preserving Graph Randomization through Edge Switch

Input: graph data G, protection threshold ε

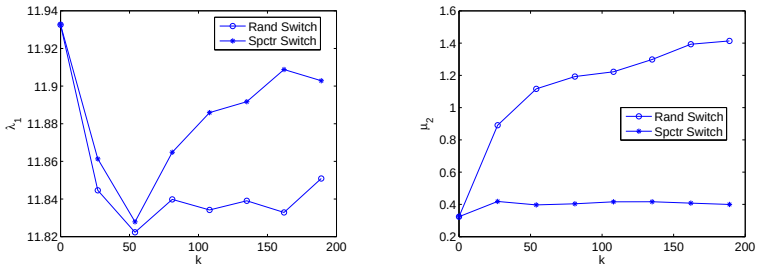
1. Derive the adjacency matrix A and the Laplacian matrix L.
2. Calculate the eigenvalues and eigenvectors $(\lambda_1, \lambda_2, \boldsymbol{e}_1)$ of A and $(\mu_2, \mu_3, \boldsymbol{u}_2)$ of L respectively.
3. $k = 0$
4. While $J_2(k) \leq 1 - \varepsilon$
5. From graph G, randomly pick one edge (t, w);
6. If $k/2 == 0$
7. Find all the edge combinations such that $\tilde{\lambda}_1 > \lambda_1$ and $\tilde{\mu}_2 > \mu_2$;
8. Randomly pick one (u, v), switch (t, w) and (u, v) to (t, v) and (u, w) ;
9. otherwise
10. Find all the edge combinations such that $\tilde{\lambda}_1 < \lambda_1$ and $\tilde{\mu}_2 < \mu_2$;
11. Randomly pick one (u, v), switch (t, w) and (u, v) to (t, v) and (u, w) ;
12. $k = k + 1$

based on the sufficient conditions listed in Table 5.2. Algorithm can be modified to *Spctr Add/Del* with some minor changes: replacing $J_2(k)$ with $J_1(k)$ in Row 4; replacing the switch process with the *Add/Del* process in Row 8 and 11; and finally, in Row 7 and 10 referring to Table 5.1 for the conditions under which the eigenvalues increase or decrease.

It is ideal to derive the sufficient conditions on how much one or some particular eigenvalues will change. This is the problem of estimating changes in eigenvalues under a wide range of perturbations. The eigenvalues of the perturbed graph can be determined as implicit functions of algebraic and geometric invariants of the original graph. However, this problem has not been solved in the matrix perturbation field.

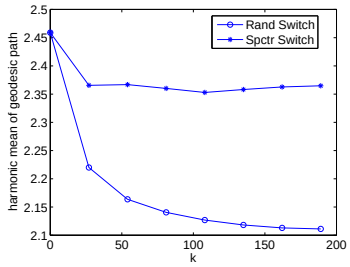
5.1.3 Empirical Evaluation

In this section, we focus on four real space characteristics of a graph: harmonic mean of the shortest distance h, modularity Q, transitivity C, and subgraph centrality SC. Figure 5.2 shows spectral randomization can significantly better preserve both graph spectrum and real space characteristics of the political book graph data

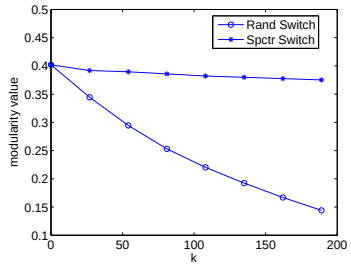


(a) λ_1

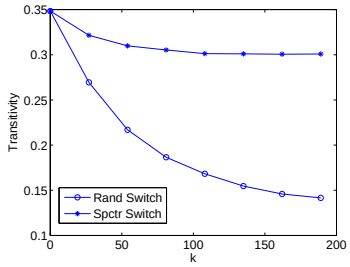
(b) μ_2



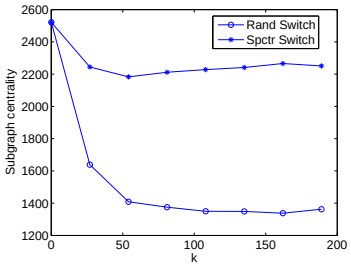
(c) harmonic mean of geodesic path



(d) modularity



(e) transitivity



(f) subgraph centrality

Figure 5.2: Graph characteristic vs. varying k between *Spctr Switch* and *Rand Switch*

set than the previous random perturbation which does not consider spectrum preserving during the perturbation process. Due to space limitations, we only include comparison between *Spctr Switch* and *Rand Switch*. We can see that *Spctr Switch* can significantly better keep both spectral characteristics and real characteristics close to those computed from the original graph even when we increase the number of switches k to 180. Note that the spectrum preserving approach adjusts both λ_1 and μ_2. The intuition here is that the more eigenvalues we control in perturbation, the more real space characteristics we can preserve in the randomized graph.

Table 5.3: Change of the measures for the US political blogs graph where the values in bold font denote the relative change from *Spctr Switch* while those in regular font denote the relative change from *Rand Switch*

k	$\lambda_1(\%)$	$\mu_2(\%)$	$h(\%)$	$Q(\%)$	$C(\%)$	$SC(\%)$
300	0.35, **0.33**	15.24, **15.68**	1.24, **1.13**	4.25, **3.87**	4.83, **4.55**	22.59, **21.67**
600	0.55, **0.51**	25.75, **22.81**	1.94, **1.70**	8.31, **6.91**	9.07, **7.69**	33.05, **30.77**
900	0.68, **0.58**	28.66, **29.83**	2.44, **2.01**	12.16, **9.33**	12.73, **9.88**	39.42, **34.06**
1200	0.77, **0.60**	32.01, **35.18**	2.81, **2.17**	15.82, **11.26**	15.91, **11.49**	43.23, **34.57**
1500	0.83, **0.58**	37.78, **47.38**	3.09, **2.26**	19.31, **12.94**	18.69, **12.65**	45.36, **33.04**
1800	0.85, **0.49**	28.93, **38.11**	3.31, **2.27**	22.61, **14.22**	21.12, **13.35**	46.50, **27.76**
2100	0.82, **0.41**	37.89, **30.05**	3.46, **2.25**	25.78, **15.49**	23.12, **13.89**	45.13, **22.58**
2400	0.79, **0.31**	50.45, **33.37**	3.59, **2.25**	28.82, **16.72**	24.88, **14.35**	43.68, **15.90**
2700	0.75, **0.23**	50.55, **20.22**	3.70, **2.24**	31.77, **17.92**	26.44, **14.78**	42.00, **10.55**
3000	0.69, **0.14**	54.27, **20.35**	3.77, **2.19**	34.53, **19.01**	27.66, **15.07**	39.32, **2.48**

We also conduct evaluation on a relatively large data set *polblogs*. Table 5.3 shows the relative change of the spectrum λ_1, μ_2 and the real characteristics (including the harmonic mean of geodesic path h, modularity Q, transitivity C, and subgraph centrality SC) between *Spctr Switch* and *Rand Switch* when we vary k from 300 to 3000. It is easy to observe that *Spctr Switch* preserve both spectrum and real characteristics of the graph much better than *Rand Switch*.

5.2 Markov Chain Based Feature Preserving Randomization

The degree sequence and topological features are of great importance to the graph structure. One natural idea is that it can better preserve the data utility if the released graph $\tilde{G}$ preserves the original degree sequence and a certain topological feature, such as transitivity or average shortest distance. On the other hand, to preserve data utility, data owners may want to preserve some particular feature $\boldsymbol{S}$ within a precise range in the released graph. All the graphs that satisfy the degree sequence $\boldsymbol{d}$ and the feature constraint $\boldsymbol{S}$ form a graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ (or $\mathcal{G}_{\boldsymbol{d}}$ if no feature constraint). Starting with the original graph, series of switches form a Markov chain that can explore the graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$. In [104], we developed an algorithm that can generate any graph in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ with equal probability.

Markov Chain. Suppose we have a finite Markov chain on the random variable X, X has finite states $\{x_1, x_2, \ldots, x_M\}$, and X^t is the random variable at time t. Denote

$$p_{ij} = \Pr(X^{t+1} = x_j | X^t = x_i),$$

as the probability that a process at state space x_i moves to state x_j in a single step and naturally $\sum_j p_{ij} = 1$. $\Pr = \{p_{ij}\}_{M \times M}$ is the transition matrix of the Markov chain with row sums equal to 1.

Lemma 5.2: [69] Suppose that a finite Markov chain on random variable X has M states $x_1, x_2, \ldots, x_M$, and it satisfies: 1) any two of its states are accessible from each other, and 2) any state has a positive probability to stay in itself. Then, the Markov chain has the unique **stationary distribution** $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_M)^T$ regardless of the initial state, where:

$$\pi_i = \lim_{t \to \infty} \Pr(X^t = x_i).$$

Moreover, $\boldsymbol{\pi}$ satisfies $\boldsymbol{\pi} = P^T \boldsymbol{\pi}$, i.e., $\boldsymbol{\pi}$ is the eigenvector of P^T with eigenvalue 1.

We first revisit previous switching based method (shown in Algorithm 2) on generating graphs without feature constraints. We then extend this method to generate graphs with feature range constraints.

5.2.1 Graph Generation without Feature Constraints

Algorithm 2 Uniform graph generator [92]

Input: initial graph G^0
Output: G^k as one sample

1: **for** $t \leftarrow 1$ to a large number k **do**
2: $\quad G^t \leftarrow SingleSwitch(G^{t-1})$;
3: **end for**
4: **return** G^k;

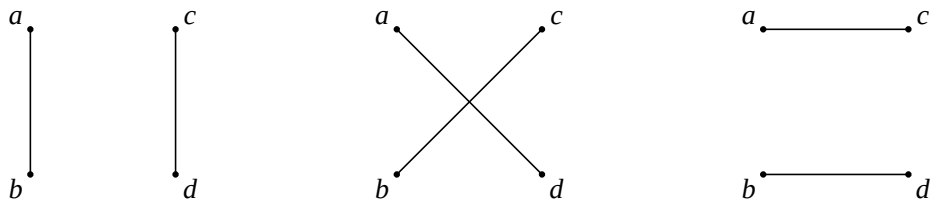
It has been well studied on how to generate graphs **uniformly** from the ensemble of all graphs that have the given degree sequence from the original graph. We show it in Algorithm 2. The algorithm uses a Markov chain to generate a random graph.

Procedure 1 Single switch

$G^{t+1} \leftarrow SingleSwitch(G^t)$

1: $r \leftarrow$ a random number from $(0, 1)$;
2: **if** $r \geq 1/2$ **then**
3: Randomly pick up two edges (a, b) and (c, d) in G^t;
4: **if** edge (a, b) and (c, d) are switchable **then**
5: $G^{t+1} \leftarrow$switch (a, b) and (c, d) in G^t;
6: **end if**
7: **end if**

The method starts from the original graph and involves carrying out a series of Monte Carlo switching steps whereby a pair of edges (a-b, c-d) is selected at random and is exchanged to give (a-d, b-c) or (a-c, b-d), illustrated in Figure 5.3. The switches preserve the degree sequence for all the graphs along the chain. The exchange is only performed if it generates no multiple edges or self-edges (we call this *switchable* in Procedure 1. The entire process is repeated k times. In the following, we explain that Algorithm 2 can generate graphs uniformly from the ensemble of all graphs that have the given degree sequence from the original graph.



(a) (b) (c)

Figure 5.3: Switch edges

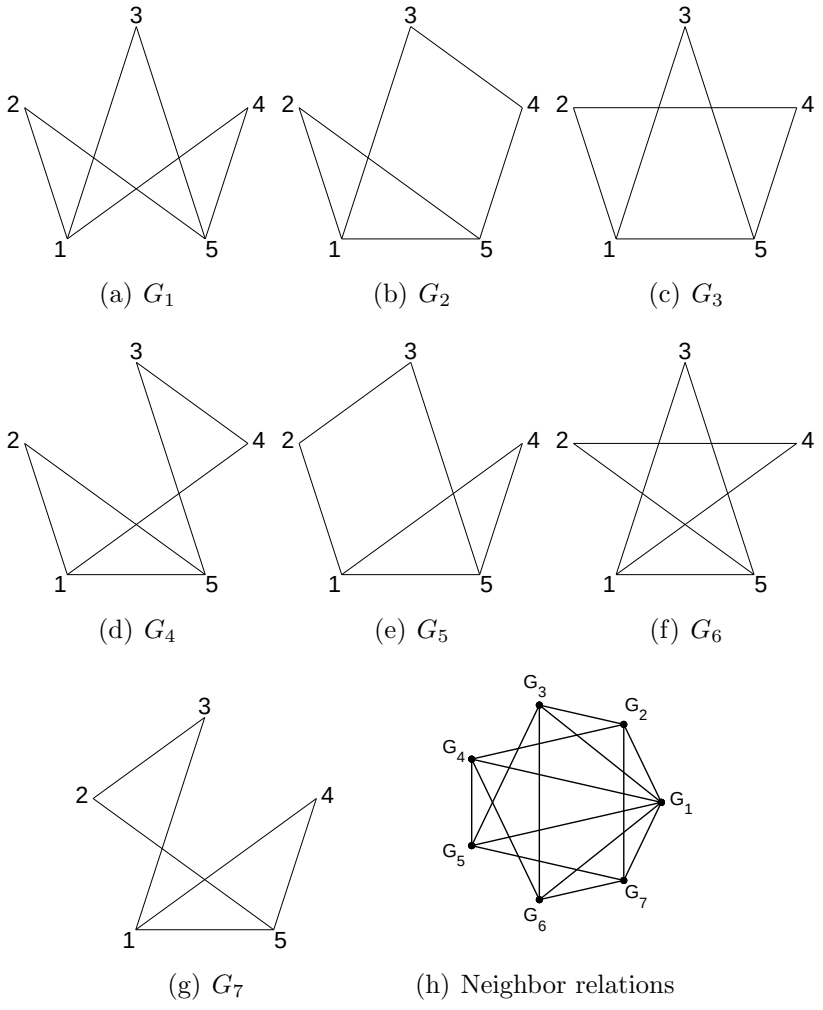
Theorem 5.3: Let $\mathcal{G}_{\boldsymbol{d}}$ be the set of all the graphs with degree sequence $\boldsymbol{d} = \{d_1, d_2, \ldots, d_n\}$. Given the starting point $G^0 \in \mathcal{G}_{\boldsymbol{d}}$, the stationary distribution of the Markov Chain in Algorithm 2 is the uniform distribution over $\mathcal{G}_{\boldsymbol{d}}$.

Each graph in $\mathcal{G}_{\boldsymbol{d}}$ corresponds to a state in the Markov chain. Line 1 and 2 in Procedure 1 makes all states have positive probabilities to remain in itself. Also, any two graphs in $\mathcal{G}_{\boldsymbol{d}}$ are accessible from each other by switchings [92], and with Lemma

5.2, the Markov chain has the unique stationary distribution $\boldsymbol{\pi}$ satisfying $\boldsymbol{\pi} = P^T\boldsymbol{\pi}$. For two graphs G_i and G_j in $\mathcal{G}_{\boldsymbol{d}}$,

$$p_{ij} := \Pr[G^{t+1} = G_j | G^t = G_i] = \frac{1}{2m(m-1)} \tag{5.12}$$

if the two graphs can be reached from each other by a single switch, and $p_{ij} = 0$ otherwise. Naturally $p_{ij} = p_{ji}$, i.e., $P^T = P$, and hence π is the eigenvector of P with eigenvalue 1. Since P has its row sums equal to 1, P has the uniform stationary distribution.



(a) G_1 (b) G_2 (c) G_3

(d) G_4 (e) G_5 (f) G_6

(g) G_7 (h) Neighbor relations

Figure 5.4: 7 graphs with degree sequence $\{3, 2, 2, 2, 3\}$

Example: Consider all the 7 graphs with the degree sequence $\{3, 2, 2, 2, 3\}$, shown

in Figure 5.4. Figure 5.4(h) shows their neighbor relations.

With (5.12), we can easily set the transition matrix of the Markov chain for this example. In P the off-diagonal entry $p_{ij}(i \neq j)$ is

$$p_{ij} = \begin{cases} \frac{1}{60}, & \text{if } G_i \text{ is adjacent to } G_j; \\ 0, & \text{if } G_i \text{ is not adjacent to } G_j; \end{cases} \tag{5.13}$$

and diagonal entries are set so that the row sum is equal to 1. It's easy to verify that P in (5.13) has uniform stationary distribution.

We start with graph G_1 and apply Algorithm 2 to generate $N = 1000$ graphs ($k = 500$). The fraction of graphs of each type is shown as below.

Graph	G_1	G_2	G_3	G_4	G_5	G_6	G_7
Count	128	138	158	143	140	144	149

(5.14)

The χ^2-statistics can be easily calculated as

$$\chi_6^2 = \sum_{i=1}^{7} \frac{[\text{Count}(G_i) - N/7]^2}{N/7} = 3.65 \tag{5.15}$$

and the corresponding p-value[1] is 0.7245, which significantly indicates the uniformity of the generated samples. □

Discussion: It is worth pointing out that not all transition matrices can generate uniformly sampled graphs. For example, to generate a random graph, one might apply the naive approach: start with G^0, for G^t, find all switchable edge pairs, randomly pick up one pair, switch them and get G^{t+1}; repeat the above steps. However, this naive approach cannot produce the uniform distribution because it actually finds all the neighbors of G^t and those graphs with more neighbors have higher probability to be generated.

One open theoretical question is how to determine the number of steps k or provide

[1] p-value is the fraction of test statistic values that are more extreme than that satisfying uniform distribution.

bounds for the mixing of the Markov chain so that the chain can approach stationarity. Theoretical bounds on the mixing time exist only for specific near-regular sequences. However, it has been shown that for many networks, $k = 10m$ appear to be adequate [70], and in [94] the author studied how to accelerate the chain. In our empirical evaluation, we simply set $k = 20m$ to ensure stationarity. Another problem of applying Markov chain is that there may exist dependence among the generated samples. There are various methods to reduce the dependence [35].

Estimate Feature Distribution over $\mathcal{G}_{\boldsymbol{d}}$. Since graphs obtained by Algorithm 2 are from the uniform stationary distribution. One immediate application of the uniform graph generator is to estimate statistic of features of graphs in $\mathcal{G}_{\boldsymbol{d}}$ or approximately construct feature distributions. Let $S(\cdot)$ be a graph feature, and $G_1, G_2, \ldots, G_N$ are N samples obtained by Algorithm 2, then the unbiased estimator of $E[S(G)]$ and $Var[S(G)]$ over $\mathcal{G}_{\boldsymbol{d}}$ are given by:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} S(G_i), \quad \hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}[S(G_i) - \hat{\mu}]^2.$$

Example continued: In our previous example, the transitivity values are

$$C(G_1) = 0, C(G_2) = \cdots = C(G_7) = 1/3,$$

and the mean and variance of transitivity over $\mathcal{G}_{\boldsymbol{d}}$ can be calculated as $E(C) = 0.2857$ and $Var(C) = 0.0136$. The estimated mean and variance of transitivity from the sample group (5.14) is $\hat{\mu} = 0.2907$ and $\hat{\sigma}^2 = 0.0124$. The 95% confidence interval for $\hat{\mu}$ is $[0.2838, 0.2976]$, and we can see that the true mean value falls within the confidence interval. □

Furthermore, we can use the sample distribution to approximate the population distribution. Let $f(x)$ be the p.d.f. of S over $\mathcal{G}_{\boldsymbol{d}}$. One method to estimate $f(x)$ using

the generated samples is the kernel density estimator:

$$\hat{f}_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K \left[\frac{x - S(G_i)}{h} \right] \tag{5.16}$$

where $K(\cdot)$ denotes the p.d.f. of the standard normal distribution and bandwidth h is the smoothing parameter.

5.2.2 Graph Generation with Feature Range Constraints

In this section, we study the problem of generating a synthetic graph whose feature S value is within a precise range of that of the original graph[2]. This is of great importance for privacy preserving social network analysis where we aim to preserve both utility and link privacy in the released perturbed graph.

We would emphasize that graphs generated by Algorithm 2 cannot preserve the utility of the original graph in general. Table 5.4 shows our empirical evaluation on four real-world social networks. We generate 3000 samples in $\mathcal{G}_{\boldsymbol{d}}$ for each graph data using our uniform graph generator . For each feature (λ_1, μ_2, harmonic mean of geodesic path h, transitivity C), we calculate its sample mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$. We also include the feature values of the original graphs. We can observe that there are usually large variations (in terms of feature standard deviation) in generated graphs. So how to generate graphs satisfying feature constraints is of great importance.

Formally, let $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ denote the ensemble of graphs with the given degree sequence $\boldsymbol{d}$ and the prescribed feature constraint $\boldsymbol{S}$. Given an initial graph G^0 with its S feature value s_0 and a constraint range $[s_-, s_+]$, we expect to generate a random graph $G \in \mathcal{G}_{\boldsymbol{d}}$ that satisfies $S(G) \in [s_-, s_+]$. One simple method is to check $S(G^t)$ value at every switch step. Algorithm 3 outlines this algorithm[3].

[2]In many practical situations, it is infeasible to require that the features (such as the harmonic mean of the shortest distance or the transitivity measure) are maintained exactly.

[3]Note that when $s_0 \notin [s_-, s_+]$, we can simply call uniform generator to reach a graph where $s_0 \in [s_-, s_+]$ and then run Algorithm 3

Table 5.4: Features of 4 graphs, including the graph value and the sample mean and standard deviation

Graphs:		dolphins	Karate	Enron	polbooks
n		62	34	151	105
m		159	78	869	441
λ_1	$\hat{\mu}$	6.90	7.08	17.54	11.90
	$\hat{\sigma}$	0.09	0.13	0.14	0.15
	G	7.19	6.73	17.83	11.93
μ_2	$\hat{\mu}$	0.45	0.71	0.91	1.62
	$\hat{\sigma}$	0.19	0.17	0.08	0.17
	G	0.17	0.47	0.81	0.32
h	$\hat{\mu}$	2.26	1.86	2.05	2.11
	$\hat{\sigma}$	0.03	0.02	0.01	0.01
	G	2.53	1.91	2.18	2.46
C	$\hat{\mu}$	0.11	0.22	0.15	0.13
	$\hat{\sigma}$	0.02	0.03	0.01	0.01
	G	0.31	0.26	0.34	0.35

Algorithm 3 Graph generator with feature range constraint

Input: G^0, $[s_-, s_+]$, $S(G^0) \in [s_-, s_+]$
Output: G^k as one sample

1: **for** $t \leftarrow 1$ to a large number k **do**
2: $G^t \leftarrow SingleSwitch(G^{t-1})$;
3: **if** $S(G^t) \notin [s_-, s_+]$ **then**
4: $G^t \leftarrow G^{t-1}$;
5: **end if**
6: **end for**
7: **return** G^k;

One interesting question is that when we preserve one feature of the graph, whether other features can also be preserved. We conduct some empirical evaluations to address this problem. We generate $N = 500$ synthetic graphs by Algorithm 3 for each of four feature range constraints, $\boldsymbol{S}_{\lambda_1}$, $\boldsymbol{S}_{\mu_2}$, $\boldsymbol{S}_h$ and $\boldsymbol{S}_C$. The range is $S(G) \pm 0.5\hat{\sigma}$, where $S(G)$ is the feature of the true graph and $\hat{\sigma}$ is the standard deviation of feature S in $\mathcal{G}_{\boldsymbol{d}}$ (shown in Table 5.4).

For those synthetic graphs, we also compute the means and standard deviations of other three uncontrolled features. Table 5.5 shows the means and standard deviations

of the feature values of the generated graphs for four networks. By comparing with Table 5.4, we can see that when λ_1 is constrained (the $\boldsymbol{S}_{\lambda_1}$ column) for polbooks, the μ_2, h or C of the generated graphs is not close to the original graph's. Instead, their distributions are similar to that of the synthetic graphs generated with no constraints. However, when μ_2 or h is constrained for polbooks, other three features are also well preserved.

We also observe that preserving μ_2 or h does not always preserve other features. For Enron data set, when μ_2 or h is confined within the range, other three features can be very different from the original graph's. This phenomenon indicates that constraining different features has different strength in preserving data utility, and this effect changes on different data sets.

Another question regarding preserving graph features is that whether attackers can exploit the feature constraint information to breach the individual privacy. We examine this problem in the next section.

5.2.3 Link Privacy Analysis

We are interested in how well graph generation can preserve the link privacy. Specifically we investigate how attackers exploit the released graph as well as feature constraints [4] to breach link privacy. In Section 5.2.3, we present one attacking method and empirically show its effectiveness in breaching link privacy.

Attacking Method. Let G and $\tilde{G}$ denote the original graph and the released graph respectively. To simplify the notation, we also use G and $\tilde{G}$ to denote their corresponding adjacency matrices.

The attacker can calculate the posterior probability of existence of a link by exploiting the $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ (or $\mathcal{G}_{\boldsymbol{d}}$ when there is no feature constraints). Naturally, if many graphs in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ have an edge at (i, j), the original graph is also very likely to have the

[4] We assume that data owners need to release the switch strategy and the feature constraints $\boldsymbol{S}$ for data mining purposes.

Table 5.5: Feature means and standard deviations of synthetic graphs with feature constraints

	dolpins				Karate			
	$\boldsymbol{S}_{\lambda_1}$	$\boldsymbol{S}_{\mu_2}$	$\boldsymbol{S}_h$	$\boldsymbol{S}_C$	$\boldsymbol{S}_{\lambda_1}$	$\boldsymbol{S}_{\mu_2}$	$\boldsymbol{S}_h$	$\boldsymbol{S}_C$
$E(\lambda_1)$	–	6.96	7.20	7.74	–	7.16	7.35	7.21
$\sigma(\lambda_1)$	–	0.09	0.09	0.23	–	0.13	0.09	0.09
$E(\mu_2)$	0.34	–	0.01	0.27	0.84	–	0.40	0.64
$\sigma(\mu_2)$	0.20	–	0.03	0.18	0.12	–	0.13	0.17
$E(h)$	2.32	2.28	–	2.41	1.83	1.88	–	1.88
$\sigma(h)$	0.05	0.02	–	0.06	0.01	0.02	–	0.02
$E(C)$	0.14	0.12	0.15	–	0.18	0.24	0.27	–
$\sigma(C)$	0.02	0.02	0.03	–	0.02	0.03	0.03	–
	polbooks				Enron			
	$\boldsymbol{S}_{\lambda_1}$	$\boldsymbol{S}_{\mu_2}$	$\boldsymbol{S}_h$	$\boldsymbol{S}_C$	$\boldsymbol{S}_{\lambda_1}$	$\boldsymbol{S}_{\mu_2}$	$\boldsymbol{S}_h$	$\boldsymbol{S}_C$
$E(\lambda_1)$	–	11.6	11.9	14.9	–	17.6	18.4	21.3
$\sigma(\lambda_1)$	–	0.11	0.14	0.50	–	0.16	0.17	0.14
$E(\mu_2)$	1.62	–	0.19	1.36	0.91	–	0.10	0.84
$\sigma(\mu_2)$	0.18	–	0.04	0.16	0.10	–	0.10	0.12
$E(h)$	2.11	2.29	–	2.23	2.07	2.06	–	2.16
$\sigma(h)$	0.01	0.02	–	0.02	0.01	0.01	–	0.02
$E(C)$	0.14	0.24	0.27	–	0.16	0.16	0.18	–
$\sigma(C)$	0.01	0.01	0.02	–	0.01	0.01	0.01	–

edge (i, j), and hence

$$\Pr[G(i,j) = 1 | \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}] = \frac{1}{|\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}|} \sum_{G_s \in \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}} G_s(i,j). \tag{5.17}$$

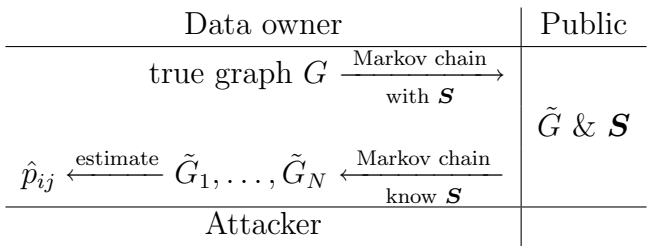


Figure 5.5: Graph publishing and attacking process

The attacking method works as follows. Starting with the released graph $\tilde{G}$, attackers apply the same randomization strategy to generate N samples $\tilde{G}_s$ ($s = 1, 2, \ldots, N$). Then attackers calculate the posterior probability of existence of a link

for all node pairs as $\hat{p}_{ij} = \frac{1}{N}\sum_{s=1}^{N} \tilde{G}_s(i,j)$ and choose top t as predicted links. Figure 5.5 illustrates this attacking methods.

The attacking method works because the convergence of the Markov chain to the stationary distribution does not depend on the initial point. In other words, starting with the released graph $\tilde{G}$, attackers can also explore the graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ similarly as starting from the original graph. Since the single switch procedure can uniformly generate graphs in $\mathcal{G}_{\boldsymbol{d}}$, for those graphs accessible by the Algorithm 3, they are also equally likely to be generated. Due to this property, $\hat{p}_{ij}$ is an unbiased estimator of the posterior probability.

Intuitively, the more strict the constraint is, the closer graphs in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ is to the original graph. Figure 5.6 shows the attacker's precisions when the range constraint on μ_2 for polbooks varies from $S(G)\pm 0.5\hat{\sigma}$ to $S(G)\pm 2\hat{\sigma}$. We compute the precisions of top t predictions, where t varies from $0.1m$ to m. We can see that the precision decreases as the range increases. When the range is $S(G) \pm 2\hat{\sigma}$, the precision approaches that without constraints. This is obvious, for as the constraints becomes wider, the graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ grows larger and eventually equal to $\mathcal{G}_{\boldsymbol{d}}$.

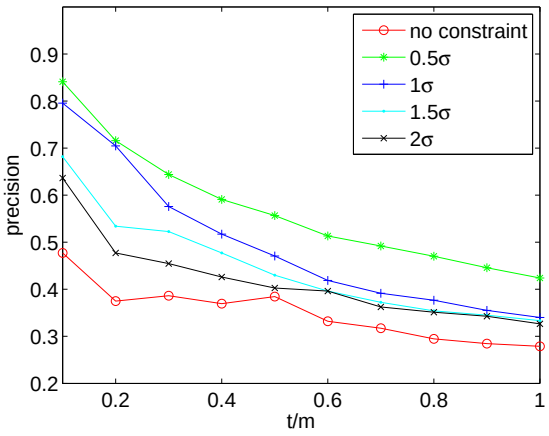


Figure 5.6: Precision of Top t predictions with μ_2 confined within different ranges for polbooks.

Figure 5.7 shows the precisions of top t predictions using four different features.

We can see that for all the cases, the attacker can achieve high accuracy, especially for those top $0.2m$ candidate links. Even when t is increased to m, the precision is much higher than random guess (with random guess the accuracy should be equal to the sparse ratio 0.08 for polbooks). Moreover, when μ_2 or h is confined within the range, the attacker can achieve even higher accuracy, and is almost sure that the top $0.2m$ candidate links are true links in the original graph. These results indicate that, by exploiting the graph space, the attacker can effectively breach the individual privacy.

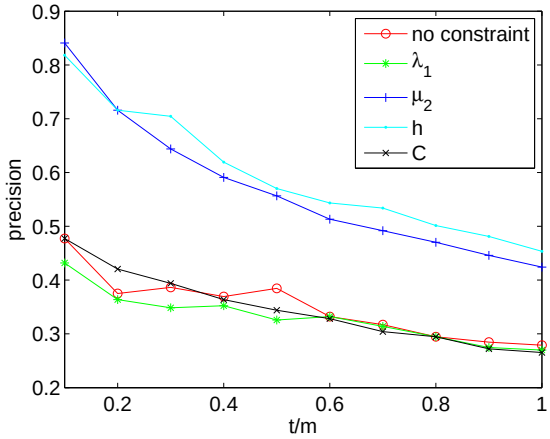
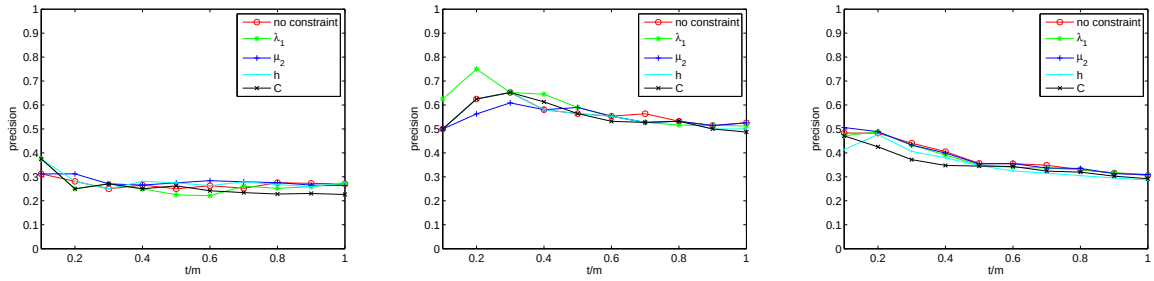


Figure 5.7: Precision of top t predictions for polbooks

We can also observe in Figure 5.7 that, when λ_1 or transitivity (C) are confined within the range, the attacker does not achieve accuracy higher than the case with no constraints, indicating that preserving features does not always jeopardize private information. We will discuss this phenomenon in the next section.

Features vs. Privacy. From Figure 5.7, we observe that preserving some feature in the released graph can significantly violate the privacy, while preserving others may not. We should also point out that, one feature that jeopardizes privacy in one graph does not necessarily jeopardize privacy in another. We evaluate the attacking method on other three networks. We can observe from Figure 5.8(c) that, for the Enron network, unlike the polbook, the attacker can not achieve higher precision when μ_2

or h are preserved. In this section, we discuss about what causes this phenomenon.



(a) dolphins (b) Karate (c) Enron

Figure 5.8: Precisions of top t predictions for different networks

Intuitively, we can measure the distance between two graphs in the graph space by the number of different edges they have. Then, two graphs that have approximately equal feature values are very likely to have shorter distance to each other.

One measure to denote the distance of two graphs is $\|G_1 - G_2\|_F^2$, where $\|\cdot\|_F$ is the Frobenius norm. Since $\tilde{G}_s$ and G have the same number of edges, it is easy to check that $\frac{1}{4}\|\tilde{G}_s - G\|_F^2$ is the number of different edges, and we can then define the relative distance measure between the original graph and the synthetic graph:

$$d(\tilde{G}_s, G) = \frac{\|\tilde{G}_s - G\|_F^2}{2\|G\|_F^2} = \frac{\|\tilde{G}_s - G\|_F^2}{4m}. \tag{5.18}$$

We can see that $d(\tilde{G}_s, G)$ is the proportion of different edges.

Table 5.6 lists the means and standard deviations of $d(\tilde{G}_s, G)$ of the attacker's N samples for different graphs. We can see that, for polbooks, when λ_1 or C is confined within the range, the mean of $d(\tilde{G}_s, G)$ is not much different from the case without constraints. However, when μ_2 or h is preserved, the mean of $d(\tilde{G}_s, G)$ is significantly smaller than the case without constraints, indicating that graphs whose μ_2 or h is constrained have less edges different from the original graph, and thus release more private information. This is consistent with our previous result that the attacker can achieve higher attacking precision when these two features are preserved for polbooks. However, for Enron network, the means of $d(\tilde{G}_s, G)$ are approximately equal in all

cases, indicating that preserving any of the features does not produce graphs closer to the original one.

Actually, as shown in our next result, the average distance of the graph space to the true graph directly affects the attacker's precision:

Result 5.2: Let $\bar{d}$ denote the expectation of $d(\tilde{G}_s, G)$ over $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$:

$$\bar{d} = E[d(\tilde{G}_s, G)] = \frac{1}{|\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}|} \sum_{\tilde{G}_s \in \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}} d(\tilde{G}_s, G).$$

When the sample size is large ($N \to \infty$), for the true edges ($ij \in G$), we have

$$\sum_{i<j, ij \in G} \hat{p}_{ij} \to m(1 - \bar{d}). \tag{5.19}$$

Proof. Let $\widetilde{G}_s$, $s = 1, 2, \ldots, N$ be the N samples uniformly from the $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$.

$$\frac{1}{N}\sum_{s=1}^{N} \|\widetilde{G}_s - G\|_F^2 = \frac{1}{N}\left|\sum_{s=1}^{N} (\widetilde{G}_s - G)^{\cdot 2}\right| = \left|\frac{1}{N}\left(\sum_s \widetilde{G}_s^{\cdot 2} - 2G \otimes \sum_s \widetilde{G}_s + NG^{\cdot 2}\right)\right|, \tag{5.20}$$

where $\otimes$ and $^{\cdot 2}$ denote the entry-wise multiplication and square respectively, and $|\cdot|$ denotes the sum of all the elements in the matrix. Since $\widetilde{G}_s$ and G are 0-1 matrices, we have $\widetilde{G}_s^{\cdot 2} = \widetilde{G}_s$ and $G^{\cdot 2} = G$, then continue with (5.20), we have

$$\begin{aligned}
\frac{1}{N}\sum_{s=1}^{N} \|\widetilde{G}_s - G\|_F^2 =& \left|\frac{1}{N}\sum_s \widetilde{G}_s - 2G \otimes \left(\frac{1}{N}\sum_s \widetilde{G}_s\right) + G\right| \\
=& \sum_{i,j} p_{ij} - 2\sum_{ij \in E} p_{ij} + 2m \\
=& 4m - 2\sum_{ij \in E} p_{ij} \quad (\text{note } \textstyle\sum_{ij} p_{ij} = 2m).
\end{aligned}$$

Therefore,

$$\frac{1}{N}\sum_{s=1}^{N} d(\widetilde{G}_s, G) = \frac{1}{N}\sum_{s=1}^{N} \frac{\|\widetilde{G}_s - G\|}{4m} = 1 - \frac{1}{m}\sum_{i<j, ij \in E} \hat{p}_{ij}.$$

With the law of large number $\frac{1}{N}\sum_{s=1}^{N} d(\widetilde{G}_s, G) \to \bar{d}$ as $N \to \infty$, and we have reached

the conclusion of (5.19). □

From (5.19), we can see that if the constraint $\boldsymbol{S}$ specifies a graph space which has smaller average distance to the true graph (smaller $\bar{d}$), the true edges must have higher estimated posterior probability $\hat{p}_{ij}$. On the other hand, since

$$\sum_{i<j,ij\notin G} \hat{p}_{ij} + \sum_{i<j,ij\in G} \hat{p}_{ij} = \sum_{i<j} \hat{p}_{ij} = m,$$

higher $\hat{p}_{ij}$ for true edges implies that the missing edges in G must have lower $\hat{p}_{ij}$. Therefore, when the attacker sorts the node pairs (i, j) by $\hat{p}_{ij}$ in descending order, the top t candidates contain more true edges and are thus more accurate.

Table 5.6: Means and standard deviations of $d(\tilde{G}_s, G)$ over different spaces with and without range constraints

constraint	no $\boldsymbol{S}$	$\boldsymbol{S}_{\lambda_1}$	$\boldsymbol{S}_{\mu_2}$	$\boldsymbol{S}_h$	$\boldsymbol{S}_C$
dolphins					
$E(d)$	.852	.848	.850	.844	.849
$\sigma(d)$	.025	.024	.025	.030	.025
Karate					
$E(d)$	.655	.650	.654	.651	.656
$\sigma(d)$	.038	.042	.037	.036	.038
polbooks					
$E(d)$	.843	.844	.736	.700	.824
$\sigma(d)$	.015	.015	.017	.018	.033
Enron					
$E(d)$	.825	.823	.824	.821	.812
$\sigma(d)$	.011	.009	.011	.010	.023

5.2.4 Relaxed Graph Generation with Feature Range Constraints

In this section and Section 5.2.5, we present two graph generation algorithms for the purpose of statistical testing. In the statistical testing, the graph generation has stricter requirements. For example, the generator should be able to access all potential graphs so that the testing result is not biased. In some other cases, the feature values of the generated graphs should follow some prescribed distribution.

All these problems involve constructing a Markov chain with a required stationary distribution. The Metropolis-Hastings method [45] is one of the standard methods of converting a Markov chain with one stationary distribution to another Markov chain with a different stationary distribution.

Metropolis-Hastings Method. Suppose on the random variable X we have a Markov chain $\mathcal{M}$ with transition matrix P and the stationary distribution $\boldsymbol{\pi}$, and we want to construct a Markov chain $\mathcal{M}^*$ whose stationary distribution is $\boldsymbol{q} = \{q_1, q_2, \ldots, q_M\}$. The Metropolis-Hastings method works as follows: suppose at time t, $X^t = x_i$, run Markov chain $\mathcal{M}$ and $X^{t+1} = x_j$, then move to x_j with probability

$$\alpha_{ij} = \min\left(1, \frac{q_j p_{ji}}{q_i p_{ij}}\right), \tag{5.21}$$

and stay in x_i otherwise. Particularly, if P is symmetric,

$$\alpha_{ij} = \min\left(1, q_j/q_i\right). \tag{5.22}$$

Generally speaking, the graph generator with feature range constraint shown in Algorithm 3 may not access all the graphs that satisfies the constraint. To overcome this problem, we develop a relaxed algorithm in this section. The relaxed algorithm, shown in Algorithm 4, can access all the graphs in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ and achieve approximate uniformity.

Algorithm 4 Relaxed graph generator with feature range constraint

Input: G^0, $[s_-, s_+]$, $q(\cdot) = \psi[S(\cdot)]$
Output: G^k as one sample

1: **for** $t \leftarrow 1$ to k **do**
2: $\quad G^t \leftarrow SingleSwitch(G^{t-1})$;
3: $\quad$ **if** $rand() \geq \min\left(1, \frac{q(G^t)}{q(G^{t-1})}\right)$ **then**
4: $\quad\quad G^t \leftarrow G^{t-1}$
5: $\quad$ **end if**
6: **end for**
7: **return** G^k;

We modify Algorithm 2 into a Markov chain with $q(\cdot)$ as its stationary distribution.

In generating graphs, $q(G)$ is the probability that a graph G is produced by the relaxed generator, and $q(G)$ should be high for those graphs in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ and should be low for those graphs not in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$. Generally speaking, we can choose

$$q(G) = \frac{\psi[S(G)]}{K}, \tag{5.23}$$

where $\psi(\cdot)$ is a positive function over the real axis such that it decreases on $[s_0, +\infty)$ and increases on $(-\infty, s_0]$ and K is a normalizer to ensure $\sum_{G\in\mathcal{G}_{\boldsymbol{d}}} q(G) = 1$. Notice that Line 3 indicates Algorithm 4 only depends on the ratio of two probabilities, we can simply set

$$q(G) \leftarrow \psi[S(G)]. \tag{5.24}$$

The transition matrix in Algorithm 2 is symmetric, and we can thus set the acceptance ratio $q(G^t)/q(G^{t-1})$ as (5.22). The connectivity of the Markov chain in Algorithm 4 is guaranteed for the acceptance ratio must be positive. Hence the chain can reach any graph in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$.

One way of choosing $\psi(\cdot)$ is to choose the p.d.f. of a normal distribution with mean equal to s_0:

$$\psi(s) = \begin{cases} \frac{1}{\sigma_1\sqrt{2\pi}} \exp\left[-\frac{(s-s_0)^2}{2\sigma_1^2}\right], & \text{if } s \geq s_0 \\ \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left[-\frac{(s-s_0)^2}{2\sigma_2^2}\right], & \text{if } s < s_0 \end{cases} \tag{5.25}$$

where $\sigma_1 = \frac{s_0 - s_-}{2}$ and $\sigma_1 = \frac{s_+ - s_0}{2}$. When $s_0 \notin [s_-, s_+]$, we can simply substitute s_0 with $\frac{s_- + s_+}{2}$ in (5.25). When we set $\psi(\cdot)$ as

$$\psi(s) = \begin{cases} 1 & \text{if } s \in [s_-, s_+] \\ 0 & \text{otherwise} \end{cases} \tag{5.26}$$

we get Algorithm 3. We can see that Algorithm 3 is a special case of the relaxed generator.

Theoretical Discussion. One theoretical question regarding to our relaxed generator is what are the feature distributions of the generated graphs. Actually, for the

relaxed generator, the distribution of $S(G)$depends on both our choice of $\psi(\cdot)$ and the natural distribution $f(x)$ of feature S.

Property 5.1: Suppose that graph G is generated by the relaxed graph generator with feature range constraint (Algorithm 4) whose $q(\cdot)$ is set as (5.23), then $S(G)$ has the distribution with p.d.f. $\frac{1}{E_f[\psi(s)]}\psi(s)f(s)$ where $E_f[\psi(s)]$ denote the expectation of $\psi(s)$ under p.d.f. $f(\cdot)$.

Proof. Note that $f(s)|\mathcal{G}_{\boldsymbol{d}}|$ is the number of graphs in $\mathcal{G}_{\boldsymbol{d}}$ whose S value equal to s, and each such graph will be generated with probability $\frac{\psi(s)}{K}$. Hence we have

$$\Pr[s(G) = s] = \frac{\psi(s)}{K} f(s)|\mathcal{G}_{\boldsymbol{d}}|, \tag{5.27}$$

Then for any interval $[a, b]$, we have

$$\Pr[a \leq S(G) \leq b] = \frac{|\mathcal{G}_{\boldsymbol{d}}|}{K} \int_a^b \psi(s)f(s)ds. \tag{5.28}$$

Let the range be the whole real axis, then

$$1 = \Pr[S(G) \in \mathbb{R}] = \frac{|\mathcal{G}_{\boldsymbol{d}}|}{K} \int_{\mathbb{R}} \psi(s)f(s)ds = \frac{|\mathcal{G}_{\boldsymbol{d}}|}{K} E_f[\psi(s)],$$

and we have $K = |\mathcal{G}_{\boldsymbol{d}}|E_f[\psi(s)]$. Combining this with (5.28), we have the property proved. □

From Property 5.1, we can know that for any two graphs $G_1, G_2 \in \mathcal{G}_{\boldsymbol{d}}$ satisfying $\psi[S(G_1)] = \psi[S(G_2)]$, they have the same probability to be generated by Algorithm 4. If $\psi(\cdot)$ is a continuous function, $q(G_1) \approx q(G_2)$ when $S(G_1) \approx S(G_2)$.

We also know that not all graphs generated by the relaxed generator have their S values within the range. According to Property 5.1, if graph G is from the relaxed generator, we have

$$\Pr(G \in \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}) = \frac{1}{E_f[\psi(s)]} \int_{s_-}^{s_+} \psi(s)f(s)ds. \tag{5.29}$$

We can see that low value of $f(x)$ over $[s_-, s_+]$ reduces the probability in (5.29). Given the graph space $\mathcal{G}_{\boldsymbol{d}}$ and the range $[s_-, s_+]$, $f(x)$ over the range is determined, and we can then increase $\psi(\cdot)$ over the range to improve the probability in (5.29). When we choose $\psi(\cdot)$ as (5.26), we have that the relaxed generator will then always on a graph within the range, for the probability in (5.29) is always equal to 1.

Figure 5.9 illustrates two choices of $\psi(\cdot)$. $\psi(\cdot)$ is the p.d.f. of a normal distribution as shown in (5.25). To make the discussion easy, we assume $s_0 = \frac{s_- + s_+}{2}$, then $\sigma_1 = \sigma_2$. If we choose a small σ as $\psi_1(\cdot)$, $\psi_1(\cdot)$ is large over $[s_-, s_+]$ and the relaxed generator has higher probability to generate a graph in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$. However, the value of $\psi(\cdot)$ changes more dramatically within the range, which reduces the uniformity of the generated graphs. When σ is large as $\psi_2(\cdot)$, $\psi(\cdot)$ does not change greatly over the range and we can guarantee the uniformity, but it reduces the probability that the generated graph is in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$.

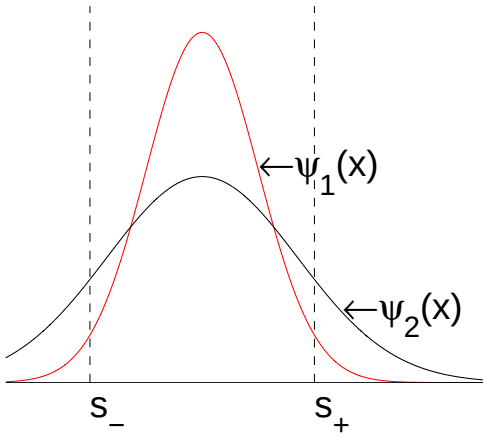


Figure 5.9: Choice of $\psi(\cdot)$

5.2.5 Graph Generation with Feature Distribution Constraints

In this Section, we study the generator that can generate graphs whose feature value satisfies a prescribed distribution.

Let $g(x)$ denote the p.d.f. of the target distribution of feature S. On the other hand, S has its own p.d.f. $f(x)$ over $\mathcal{G}_{\boldsymbol{d}}$. Algorithm 5 outlines the graph generator

with feature distribution constraint.

Algorithm 5 Graph generator with feature distribution constraint

Input: G^0, $g(\cdot)$, $f(\cdot)$
Output: G^k as one sample

1: **for** $t \leftarrow 1$ to k **do**
2: $G^t \leftarrow SingleSwitch(G^{t-1})$;
3: **if** $rand() \geq \min\left(1, \frac{g[S(G^t)]f[S(G^{t-1})]}{g[S(G^{t-1})]f[S(G^t)]}\right)$ **then**
4: $G^t \leftarrow G^{t-1}$
5: **end if**
6: **end for**
7: **return** G^k;

From Property 5.1, we know that given any input function $\psi(x)$, the generated distribution of S value has the p.d.f. as $\frac{1}{E_f[\psi(x)]}\psi(x)f(x)$. By replacing $\psi(x)$ with $\frac{g(x)}{f(x)}$ in (5.29), we have

$$E_f[\psi(s)] = E_f\left[\frac{g(x)}{f(x)}\right] = \int_{-\infty}^{+\infty} \frac{g(x)}{f(x)} f(x)dx = 1.$$

Then also from (5.29) we have

$$\begin{aligned}\Pr[S(G) \leq x] =& \frac{1}{E_f[\psi(s)]} \int_{-\infty}^{x} \frac{g(t)}{f(t)} f(t)dt \\ =& \int_{-\infty}^{x} g(t)dt,\end{aligned}$$

and then the p.d.f. of S value is equal to $g(x)$. Hence, by setting $q(\cdot)$ in (5.22) as

$$q(G) \leftarrow g[S(G)]/f[S(G)], \tag{5.30}$$

we can achieve the target distribution in Algorithm 5. We would like to point out that, if we know that statistic S has uniform distribution over $\mathcal{G}_{\boldsymbol{d}}$, (5.30) is reduced to set $q(G) \leftarrow g[S(G)]$. However, generally speaking, statistic S is not uniformly distributed and we can not just set $q(G) \leftarrow g[S(G)]$.

Example continued: Continue with the previous example. In $\mathcal{G}_{\boldsymbol{d}}$, G_1 has transitivity value 0 and the remaining six graphs have the same transitivity value $\frac{1}{3}$. The

probability function of transitivity in $\mathcal{G}_{\boldsymbol{d}}$ is

C value	0	1/3
$f(x)$	1/7	6/7

Suppose that we want to generate a series of graphs in $\mathcal{G}_{\boldsymbol{d}}$ so that the required probability function $g(x)$ on transitivity to be:

C value	0	1/3
$g(x)$	0.4	0.6

(5.31)

We set $q(\cdot)$ as in (5.30): $q(G_i) = g[C(G_i)]/f[C(G_i)]$. Let Q denote the transition matrix of this Markov chain, then $Q_{ij} = p_{ij} \times \min\{1, q(G_j)/q(G_i)\}$ for $i \neq j$, and set the diagonal entries so that Q has row sums equal to 1:

$$Q = 10^{-2} \times \begin{pmatrix} 97.50 & 0.42 & 0.42 & 0.42 & 0.42 & 0.42 & 0.42 \\ 1.67 & 93.33 & 1.67 & 1.67 & 0 & 0 & 1.67 \\ 1.67 & 1.67 & 93.33 & 0 & 1.67 & 1.67 & 0 \\ 1.67 & 1.67 & 0 & 93.33 & 1.67 & 1.67 & 0 \\ 1.67 & 0 & 1.67 & 1.67 & 93.33 & 0 & 1.67 \\ 1.67 & 0 & 1.67 & 1.67 & 0 & 93.33 & 1.67 \\ 1.67 & 1.67 & 0 & 0 & 1.67 & 1.67 & 93.33 \end{pmatrix}.$$

We can verify that the stationary distribution of the new Markov chain is

$$(0.4, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1),$$

which makes the transitivity distributed as shown in (5.31). We apply Algorithm 5 on this example, and sample group (5.32) lists 1000 samples ($k = 500$):

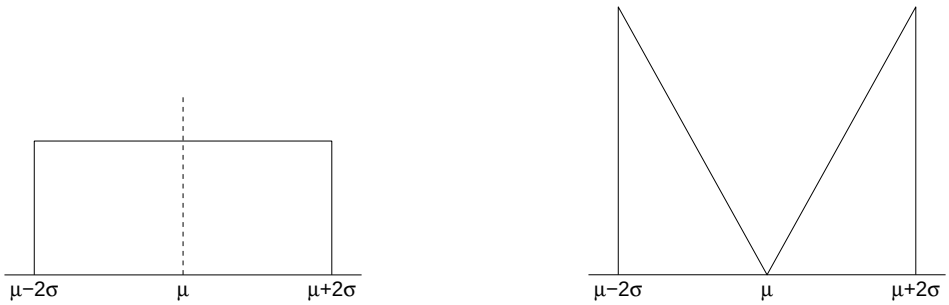
Graph	G_1	G_2	G_3	G_4	G_5	G_6	G_7
Count	399	90	93	109	97	105	107

(5.32)

Comparing sample group (5.32) with the stationary distribution, we have the χ^2-

statistics equal to 3.1325 with its p-value equal to 0.7920, which indicates the generated graphs well match the target distribution. □

5.2.6 Empirical Evaluation



(a) uniform (b) double-triangle

Figure 5.10: Target distributions $g(x)$

We apply Algorithm 5 on graph polbooks to simulate two distributions for four features: λ_1, μ_2, harmonic mean of shortest distance (h), and transitivity (C). The first distribution is the uniform distribution on interval $[\hat{\mu} - 2\hat{\sigma}, \hat{\mu} + 2\hat{\sigma}]$, where $\hat{\mu}$ and $\hat{\sigma}$ are the sample mean and standard deviation of graph polbooks from Table 5.4. The second distribution is a double-triangle-shaped distribution:

$$g(x) = \frac{|x - \hat{\mu}|}{4\hat{\sigma}^2}, \ x \in [\hat{\mu} - 2\hat{\sigma}, \hat{\mu} + 2\hat{\sigma}].$$

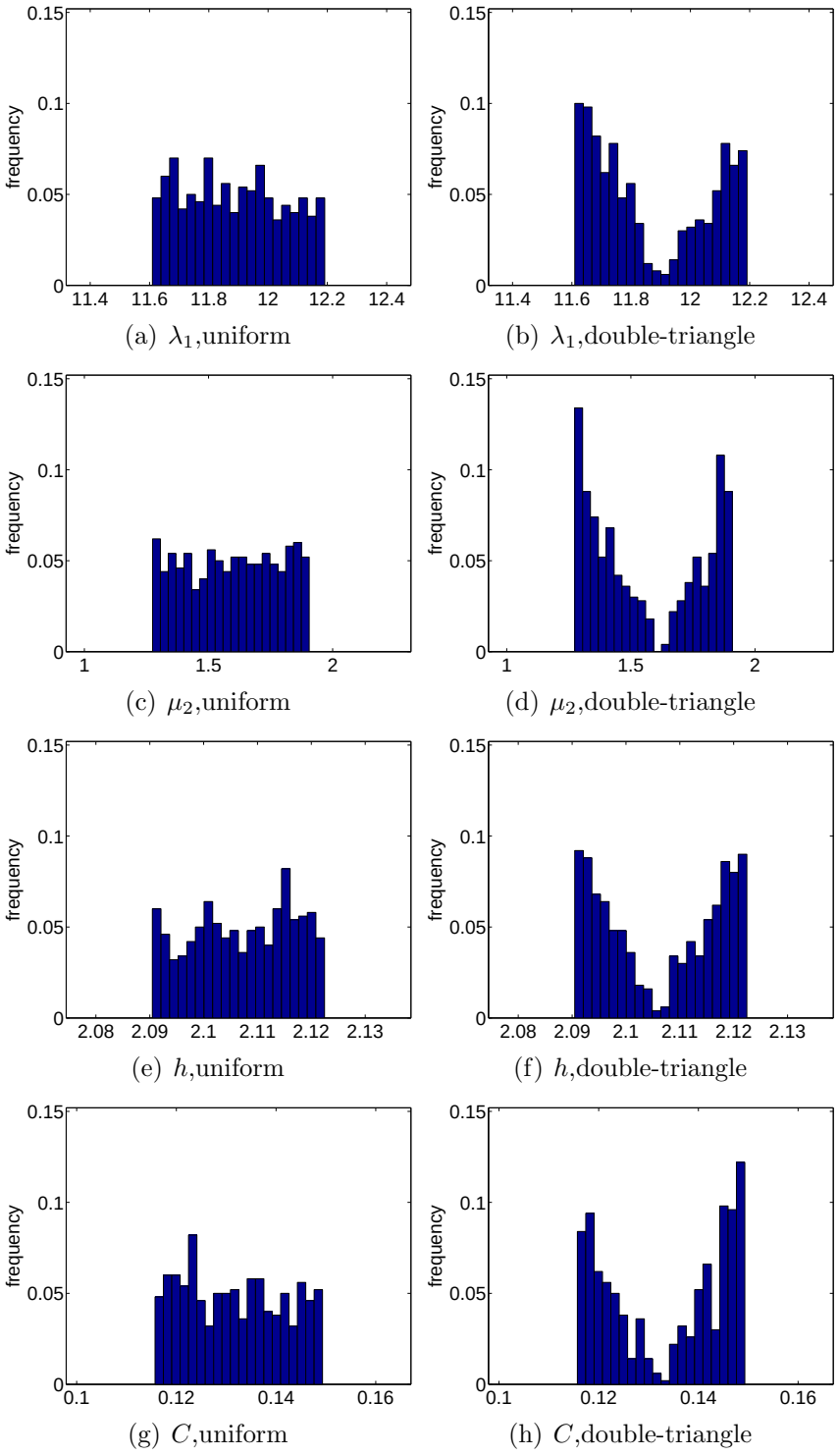
The shapes of the two target distributions are shown in Figure 5.10. Both of them are very different from the features' natural distributions $f(x)$. When applying Algorithm 5, we need to know the natural distribution of those features $f(x)$, and we use the kernel density estimator shown in (5.16) to estimate $f(x)$ from the 3000 uniformly generated samples. Figure 5.11 shows the distributions of the four features of the 500 generated samples ($k = 6000$) using Algorithm 5. We can observe from Figure 5.11 that all the four features of generated samples match well the target distributions (shown in Figure 5.10).

In many practical cases that some feature distribution $f(\cdot)$ over $\mathcal{G}_{\boldsymbol{d}}$ is unknown,

the cost of estimating $f(\cdot)$ can be high since we need to generate a large number of uniformly sampled graphs. To reduce the cost, we may simply specify $f(\cdot)$ as some a-priori distribution (e.g., normal or uniform distribution) although it may sacrifice the accuracy of feature target distribution of the generated samples.

5.3 Summary

In this chapter, we develop two types of feature-preserving graph randomization procedures. The first type is the spectrum preserving graph randomization procedures, *Spctr Add/Del* and *Spctr Switch*, which can better preserve graph characteristics via preserving two eigenvalues of graph matrices, λ_1 and μ_2. The second type of graph randomization procedure we presented is a simple switching based graph generator which can preserve any feature of a real graph specified by users. We then investigate the potential disclosure of sensitive links due to the preserved features. Based on Metropolis-Hastings sampling, our graph generator can be easily modified to generate synthetic graphs whose features satisfy a given distribution. This is of great importance for significance testing of network analysis results.



(a) λ_1,uniform (b) λ_1,double-triangle

(c) μ_2,uniform (d) μ_2,double-triangle

(e) h,uniform (f) h,double-triangle

(g) C,uniform (h) C,double-triangle

Figure 5.11: Feature distributions of generated graphs with feature distribution constraints shown in Figure 5.10 for *polbooks*.

CHAPTER 6: SPECTRAL ANALYSIS OF SOCIAL NETWORKS

In Chapter 5, we discussed the spectrum-preserving randomization procedures that can preserve graph features via preserving some graph spectra. In this chapter, we conduct more studies on the relation between the graph spectrum and the topological patterns of the graph. One general question is that what information about the graph the spectrum decodes. The real-world social networks are very different from the random ones. As we apply randomization to a real-world graph, the graph approaches to a random one. Then, can the graph spectrum captures the change from a real-world graph to a random one? How can we measure the difference between a real-world graph (or a randomized one) and a pure random one? In this chapter, we develop a consistent framework to measure the randomness at the edge, node, subgraph, and the overall graph level. The measures are based on the adjacency spectral space. We further develop a community partition algorithm utilizing the adjacency spectral geometry.

Let λ_i be the i-th largest eigenvalues of A and $\boldsymbol{x}_i$ the corresponding eigenvectors, and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. The spectral decomposition of A is $A = \sum_i \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^T$. Let $\boldsymbol{x}_i$ be the unit eigenvector of λ_i and let x_{ij} denote the j'th entry of $\boldsymbol{x}_i$.

As shown in (6.1), the eigenvector $\boldsymbol{x}_i$ is represented as a column vector. The row vector $(x_{1u}, x_{2u}, \cdots, x_{nu})$ represents the coordinates of node u in the n-dimensional spectral space. In Section 6.1, we show that only the coordinates of node u in the first k-dimensional spectral space determine the randomness of u where k indicates the number of communities within the graph. Hence we define $\boldsymbol{\alpha}_u = (x_{1u}, x_{2u}, \ldots, x_{ku}) \in \mathbb{R}^{1\times k}$ as the spectral coordinate of node u in the k-dimensional space.

$$
\begin{array}{c}
\begin{array}{cccccc} \boldsymbol{x}_1 & & \boldsymbol{x}_i & & \boldsymbol{x}_k & \boldsymbol{x}_n \end{array} \\
\downarrow \\
\boldsymbol{\alpha}_u \rightarrow \left(\begin{array}{c|c|c|c}
x_{11} \cdots & x_{i1} & \cdots\ x_{k1} & \cdots\ x_{n1} \\
\vdots & \vdots & \vdots & \vdots \\
\hline
x_{1u} \cdots & x_{iu} & \cdots\ x_{ku} & \cdots\ x_{nu} \\
\hline
\vdots & \vdots & \vdots & \vdots \\
x_{1n} \cdots & x_{in} & \cdots\ x_{kn} & \cdots\ x_{nn}
\end{array}\right)
\end{array}
\tag{6.1}
$$

In Section 6.1, we introduce the graph spectral geometry and its relationship with the graph topology, especially the community structure. Based on the graph spectral patterns, in Section 6.2, we develop a consistent framework to measure the randomness contained in the graph at the edge, node, sub-graph to the overall graph level. We compare the graph patterns in the adjacency spectral space with those in other spectral spaces in Section 6.3. We report our evaluation results in Section 6.4. In Section 6.5, we present a community partition algorithm based on the adjacency spectral geometry. Some of the results in this Chapter are also reported in [105, 108].

6.1 Graph Spectral Geometry

In this section, we explore how the spectral coordinate ($\boldsymbol{\alpha}$) of a node point locates in the projected spectral space. Especially we show that node points locate along k quasi-orthogonal lines when graph G contains k communities [1].

Proposition 6.1: For a graph with k communities, the coordinate of node u in k-dimensional space, $\boldsymbol{\alpha}_u = (x_{1u}, x_{2u}, \ldots, x_{ku}) \in \mathbb{R}^{1\times k}$, denotes the likelihood of node u's attachment to these k communities. Node points within one community form a line that goes through the origin in the k-dimensional space. Nodes in k communities form k quasi-orthogonal lines in the spectral space.

[1] Communities are loosely defined as collections of individuals who interact unusually frequently.

Proof. Consider the division of a graph G into k non-overlapping communities $G_1, G_2, \ldots, G_k$. Let $\boldsymbol{s}_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ be the index vector of community G_i, and s_{ij} equals to 1 if node j belongs to community G_i and 0 otherwise. Note that $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ are mutually orthogonal, i.e., $\boldsymbol{s}_i^T \boldsymbol{s}_j = 0$.

For community G_i, we can define its density as

$$D(G_i) := \frac{\#\text{ of edges in } G_i}{\#\text{ of nodes in } G_i}.$$

It can be expressed as

$$D(G_i) = \frac{\boldsymbol{s}_i^T A \boldsymbol{s}_i}{\boldsymbol{s}_i^T \boldsymbol{s}_i}$$

where A is the adjacency matrix of graph G. The density for this division of the graph is

$$\sum_{i=1}^{k} D(G_i) = \sum_{i=1}^{k} \frac{\boldsymbol{s}_i^T A \boldsymbol{s}_i}{\boldsymbol{s}_i^T \boldsymbol{s}_i} \tag{6.2}$$

The task of our graph partition is to maximize (6.2) subject to $s_{ij} \in \{0, 1\}$ and $\boldsymbol{s}_i^T \boldsymbol{s}_j = 0$, if $i \neq j$. This optimization problem is NP-complete. However, if we relax $s_{ij} \in \{0, 1\}$ to real space, based on the Wielandt's theory [90], we have that the target function reaches the maximum $\sum_{i=1}^{k} \lambda_i$ when taking $\boldsymbol{s}_i$ to be $\boldsymbol{x}_i$.

Replacing A with $\sum_i \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^T$ in (6.2), we can derive that $\sum_{i=1}^{k} D(G_i)$ would be maximized by choosing the $\boldsymbol{s}_i$ proportional to the ith eigenvector $\boldsymbol{x}_i$ of the adjacency matrix when we relax the basic constraint $s_{ij} \in \{0, 1\}$. Hence we can conclude that x_{ij} reflects the degree of node j's attachment to the community G_i. □

Property 6.1: A node u belongs to one community G_t if the tth entry of $\boldsymbol{\alpha}_u$, x_{tu}, is much greater than the rest entries and $x_{iu} \approx 0$ for $i \neq t$.

A node u does not belong to any community if all the entries of $\boldsymbol{\alpha}_u$ are close to 0, or equivalently, $\|\boldsymbol{\alpha}\|_2 \approx 0$. We call such nodes noise nodes.

Property 6.2: If nodes u and v belong to the same community, then

$$|\cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_v)| \approx 1.$$

If nodes u and v belong to two different communities respectively, then

$$| \cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_v) | \approx 0.$$

Otherwise, if node u belongs to one community G_t and bridging node v locates in the overlap of two communities G_t and G_w, then $| \cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_v) |$ is not close to either 0 or 1.

Explanation. Notice that

$$\cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_v) = \frac{\boldsymbol{\alpha}_u \boldsymbol{\alpha}_v^T}{\|\boldsymbol{\alpha}_u\|_2 \|\boldsymbol{\alpha}_v\|_2}.$$

When node u and v are in the same community G_t, x_{tu}, we have that x_{tv} is much greater than the rest entries in $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_v$. Hence

$$\begin{aligned} \frac{\boldsymbol{\alpha}_u \boldsymbol{\alpha}_v^T}{\|\boldsymbol{\alpha}_u\|_2 \|\boldsymbol{\alpha}_v\|_2} &= \frac{\sum_{i=1}^k x_{iu} x_{iv}}{\left(\sum_{i=1}^k x_{iu}^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^k x_{iv}^2\right)^{\frac{1}{2}}} \\ &\approx \frac{x_{tu} x_{tv}}{|x_{tu}||x_{tv}|} = \pm 1. \end{aligned}$$

In other words, points $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_v$ approximately locate along a straight line that goes through the origin.

Similarly, when node u and v are in two different communities G_t and G_w respectively, with $x_{wu} \approx 0$ and $x_{tv} \approx 0$, we have

$$\frac{\boldsymbol{\alpha}_u \boldsymbol{\alpha}_v^T}{\|\boldsymbol{\alpha}_u\|_2 \|\boldsymbol{\alpha}_v\|_2} \approx \frac{x_{tu} x_{tv} + x_{wu} x_{wv}}{|x_{tu}||x_{wv}|} \approx 0,$$

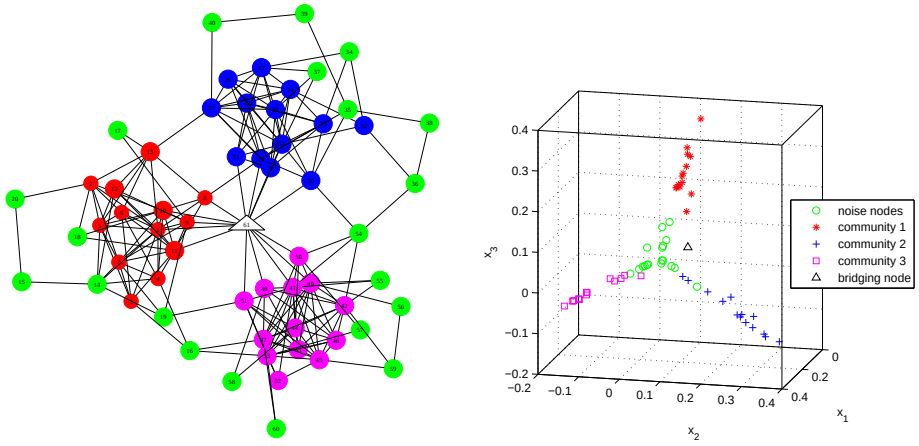
which means that $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_v$ are approximately orthogonal.

If a bridging node v is in the overlap of two communities S_t and S_w, both tth and wth entries in $\boldsymbol{\alpha}_v$ are not negligible. Hence, $\|\boldsymbol{\alpha}_v\|_2 \approx (x_{tv}^2 + x_{wv}^2)^{\frac{1}{2}}$. For a node u from G_t, we have

$$\frac{\left|\boldsymbol{\alpha}_u \boldsymbol{\alpha}_v^T\right|}{\|\boldsymbol{\alpha}_u\|_2 \|\boldsymbol{\alpha}_v\|_2} \approx \frac{|x_{tu} x_{tv}|}{|x_{tu}| \left(x_{tv}^2 + x_{wv}^2\right)^{\frac{1}{2}}} = \frac{|x_{tv}|}{\left(x_{tv}^2 + x_{wv}^2\right)^{\frac{1}{2}}}.$$

Since neither x_{tv} nor x_{wv} is close to 0, $|\cos(u, v)|$ is not close to either 1 or 0, which

indicates that bridging nodes locate between the quasi-orthogonal lines formed by communities, and are also away from the origin.



(a) The graph topology (b) The spectral geometry

Figure 6.1: A synthetic network with 3 communities and its spectral coordinates projected in 3-D plot.

Figure 6.1(b) shows the 3-D spectral geometries of a synthetic network as shown in Figure 6.1(a). In Figure 6.1(a), there exist three dense subgraphs (denoted by red, blue and pink color respectively), which are separated by one bridging node (node 61, denoted by a white triangle), in addition to some random nodes (denoted by green color). We can observe from Figure 6.1(b) that nodes in the three dense subgraphs are projected along three straight and quasi-orthogonal lines in the 3-D spectral space and nodes in green locate around the origin in the projected space. We can also observe that node 61 (white triangle), which bridges the three communities, locates away from the origin and among the three quasi-orthogonal lines.

6.2 A Framework of Measuring Graph Non-randomness

In this section, we present our framework which can quantify randomness at all granularity levels from edge, node, subgraph, to the overall graph. We begin with a study of edge non-randomness by spectral coordinates of its two connected nodes in the spectral space. We then define the node non-randomness as the sum of non-

randomness values of all edges that connect to it. Similarly, we define the overall graph (subgraph) non-randomness as the sum of non-randomness values of all edges within the the whole graph (subgraph). The formal definition is given below.

Definition 6.1: Denote $\boldsymbol{\alpha}_u = (x_{1u}, x_{2u}, \ldots, x_{ku}) \in \mathbb{R}^k$ as the spectral coordinate of node u and $\boldsymbol{\alpha}_v = (x_{1v}, x_{2v}, \ldots, x_{kv}) \in \mathbb{R}^k$ as the spectral coordinate of node v.

1. The edge non-randomness $R(u, v)$ is defined as

$$R(u, v) = \boldsymbol{\alpha}_u \boldsymbol{\alpha}_v^T = \sum_{i=1}^{k} x_{iu} x_{iv}.$$

2. The node non-randomness $R(u)$ is defined as

$$R(u) = \sum_{v \in \Gamma(u)} R(u, v),$$

   where $\Gamma(u)$ denotes the neighbor set of node u.

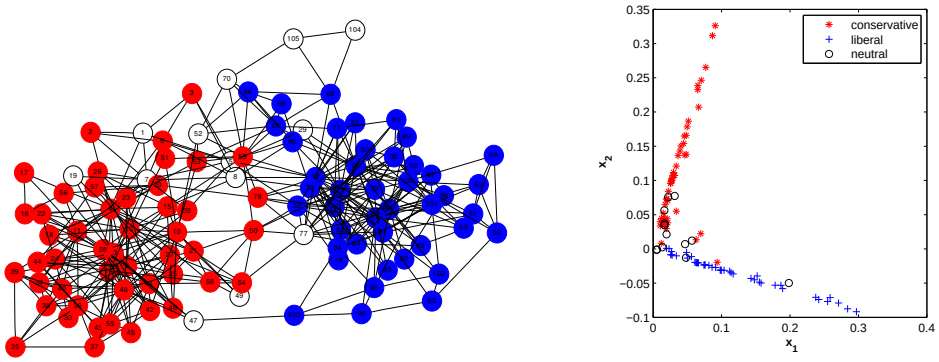
3. Let G_1 be a subgraph of $G(V, E)$ with node set $V_1 \subseteq V$ and edge set $E_1 \subseteq E$. The subgraph non-randomness $R(G_1)$ (with respect to the G) is defined as:

$$R(G_1) = \sum_{(u,v) \in E_1} R(u, v). \tag{6.3}$$

4. The graph non-randomness R_G is defined as

$$R_G = \sum_{(u,v) \in E} R(u, v).$$

Throughout this section, we use *polbooks* network as an example to illustrate how we define and calculate graph non-randomness at various levels. Figure 6.2(b) shows the 2-D spectral geometries of the politics book network data. We can observe from Figure 6.2(b) that the majority of vertices projected in the 2-D spectral space distribute along two straight and quasi-orthogonal lines. It indicates that there exist two communities with sparse edges connecting them. The first up-trend line consists of most nodes in



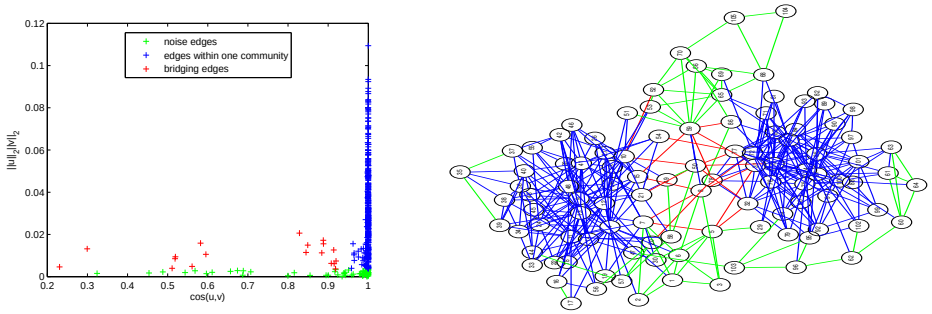
(a) Topology snapshot (b) 2-D spectral geometry

Figure 6.2: Politics book social network

red color while the second down-trend line consists of most nodes in blue color. White nodes distribute either around the origin or between two quasi-orthogonal lines in the projected space.

6.2.1 Edge Non-randomness: $R(u, v)$

From Section 6.1, we know that the spectral coordinates of a node reflect its relative attachment to different communities in G. When it comes to the measure of non-randomness of an edge that connects two nodes, intuitively, we need to incorporate the relationship of two nodes' spectral vectors.



(a) (b)

Figure 6.3: Snapshot of different types of edges characterized by edge non-randomness of politics book network

The edge non-randomness measure $R(u, v)$ in Definition 1 can be rewritten as

$$R(u, v) = \|\boldsymbol{\alpha}_u\|_2 \|\boldsymbol{\alpha}_v\|_2 \cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_u),$$

which is determined by the product of $\|\boldsymbol{\alpha}_u\|_2\|\boldsymbol{\alpha}_v\|_2$ and the cosine of the angle between $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_u$. Generally, $R(u, v)$ tends to be large when u and v are clearly belong to the same community (since $\cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_u) \approx 1$). $R(u, v)$ tends to be small when 1) u and v are from two different communities (since $\cos(\boldsymbol{\alpha}_u, \boldsymbol{\alpha}_u) \approx 0$); 2) or either node (or both nodes) is noisy (since $\|\boldsymbol{\alpha}_u\|_2\|\boldsymbol{\alpha}_v\|_2 \approx 0$). This intuitively reflects the formation of real world social networks: two individuals within the same community have relatively higher probability to be connected than those in different communities.

Figure 6.3(a) plots the distribution of edge non-randomness values, where x-axis is the cosine value between $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_v$ while y-axis denotes the product of the two vector lengths. Figure 6.3(b) shows a snapshot of different types of 441 edges characterized by edge non-randomness values of politics book network. We can observe that distributions of edge non-randomness values characterized by different regions reflect different types of edges in the original graph: edges with large cosine value (plotted along the vertex line $x = 1$ and denoted by the blue '+') mostly connect two nodes within the same community; edges with small vector length product (green '+' and plotted along the line $y = 0$) mostly connect to non-central nodes; edges plotted in other area forms bridging edges between the two communities. All the above is consistent with our previous explanations in Section 6.1.

6.2.2 Node Non-randomness: $R(u)$

A node's non-randomness is characterized by the non-randomness of edges connected to this node. This is well understood since edges in social networks often exhibit patterns that indicate properties of the nodes such as the importance, rank, or category of the corresponding individuals. Result 6.1 shows how to calculate the node non-randomness using the spectral coordinates as well as the first k eigenvalues of the adjacency matrix.

Result 6.1: The non-randomness of node u is the length of its spectral vector with

eigenvalue weighted on corresponding dimensions:

$$R(u) = \sum_{i=1}^{k} \lambda_i x_{iu}^2 = \boldsymbol{\alpha}_u \Lambda_k \boldsymbol{\alpha}_u^T, \tag{6.4}$$

where $\Lambda_k = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$.

Proof. Let $\boldsymbol{a}_u$ denote the u'th row of the adjacency matrix A. Since $\boldsymbol{x}_i$ satisfies $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$ and A is symmetric,

$$\begin{pmatrix} \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_n \end{pmatrix} \boldsymbol{x}_i = A\boldsymbol{x}_i = \lambda_i \begin{pmatrix} x_{i1} \\ \vdots \\ x_{in} \end{pmatrix}.$$

Hence, $\boldsymbol{a}_u \boldsymbol{x}_i = \lambda_i x_{iu}$, and we have

$$\begin{aligned} R(u) &= \sum_{v \in \Gamma(u)} R(u,v) = \sum_{v=1}^{n} \sum_{i=1}^{k} a_{uv} x_{iu} x_{iv} \\ &= \sum_{i=1}^{k} \left(x_{iu} \sum_{v=1}^{n} a_{uv} x_{iv} \right) \\ &= \sum_{i=1}^{k} x_{iu} \boldsymbol{a}_u \boldsymbol{x}_i = \sum_{i=1}^{k} \lambda_i x_{iu}^2 = \boldsymbol{\alpha}_u \Lambda_k \boldsymbol{\alpha}_u^T. \end{aligned}$$

□

We can see that the result is elegant since the node non-randomness is actually determined by its vector length weighted by eigenvalues of the adjacency matrix.

Using node non-randomness measure, we can easily separate singleton nodes [2] and noise nodes (with small $R(u)$ values) from those nodes strongly attached to some community (with large $R(u)$ values). We can also identify those nodes bridging across several groups by examining its relative positions to orthogonal lines corresponding to different communities.

Comparison with HITS. Our node non-randomness $R(u)$ can be used to identify

[2]The singletons are degree-zero nodes who joined the network but have never made an interaction with another user in the social network.

those non-random individuals. However, it is different from those traditional link based object ranking methods based on centrality measures. For example, HITS algorithm [55] uses the principle eigenvector to assign *authority/hub* scores to each node. For undirected social networks, since A is now symmetric, *authority* and *hub* scores are the same, which are the principle eigenvector of A^2. Denote $A = X\Lambda X^T$ as the eigen-decomposition of A. Since X is orthogonal, $A^2 = X\Lambda^2 X$, the *authority/hub* scores from HITS algorithm in undirected networks are equivalent to the entries of $\boldsymbol{x}_1$. Therefore, if we are sure that the graph has only one community, our measure is reduced to the HITS score. However, many real-world graphs contain more then one community.

Table 6.1: Comparison of top 10 non-random nodes identified by $R(u)$ and HITS.

HITS	label	$R(u)$	label
85	liberal	9	conservative
74	liberal	13	conservative
73	liberal	85	liberal
31	liberal	74	liberal
67	liberal	73	liberal
75	liberal	4	conservative
76	liberal	31	liberal
77	neutral	67	liberal
87	liberal	12	conservative
72	liberal	75	liberal

Table 6.1 compares the difference between the top 10 non-random nodes identified by our measure and the those identified by HITS for *polbooks* network. We can observe from the Table 6.1 that top 10 nodes identified by our measures include important nodes from two communities while HITS only identifies nodes from one community. This is because HITS uses $\boldsymbol{x}_1$ only, the scores only reflect relative positions of points along the x_1-axis in Figure 6.2(b). Hence they can only discover central nodes in one community (labeled as *liberal*) with the highest density. On the contrary, our node non-randomness measure, which uses the weighted vector length in the k-dimensional

spectral space, can successfully discover non-random nodes from all k communities. This empirical evaluation indicates our node non-random measure is different from the traditional centrality measures used to rank nodes.

6.2.3 Graph Non-randomness R_G and Relative Non-randomness R_G^*

In our framework, the graph non-randomness R_G is defined as the sum of non-randomness values of all edges within the graph. Result 6.2 shows R_G can be directly calculated using the first k eigenvalues.

Result 6.2: The graph non-randomness of the overall graph G can be calculated as

$$R_G = \sum_{(u,v)\in E} R(u,v) = \sum_{u\in G} R(u) = \sum_{i=1}^{k} \lambda_i \tag{6.5}$$

Proof. The second equation is straightforward. For the third equation, denote X as $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ where each column is an eigenvector of A: $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$, hence we have

$$\sum_{(u,v)\in E} R(u,v) = \sum_{u,v} a_{uv} \boldsymbol{\alpha}_u \boldsymbol{\alpha}_v^T = \text{trace}(X^T A X) = \sum_{i=1}^{k} \lambda_i.$$

□

The above result is elegant since we can use the sum of the first k eigenvalues to determine the non-randomness of the overall graph. Recall that k indicates the number of communities in the graph. In this paper, we assume the value of k is either specified by domain users or discovered by those graph partition methods. There are tons of work on how to partition graph into k communities (refer to a survey paper [16]).

Chung and Graham indicated the use of the largest eigenvalue λ_1 as an index of the non-randomness of the overall graph since the first eigenvalue of random graphs characterizes the frequency of subgraphs [21]. Our analysis shows that λ_1 may not be an appropriate measure to quantify the graph non-randomness for real-world social

networks since they usually contain more than one communities. Actually, we can see that the index of graph non-randomness using λ_1 is a special case of our proposed measure R_G with $k = 1$.

All real networks lie somewhere between the extremes of complete order and complete randomness. While the absolute non-randomness measure R_G can indicate how random a graph G is, it is more desirable to give a relative measure so that graphs with different size and density can be compared. One intuitive approach is comparing the graph's non-randomness value with the expectation of non-randomness value of all random graphs generated by ER model. We can use the standardized measure defined as

$$R_G^* = \frac{R_G - E(R_G)}{\sigma(R_G)}$$

where $E(R_G)$ and $\sigma(R_G)$ denote the expectation and standard deviation of the graph non-randomness under ER model. Our Theorem 6.1 shows the distribution of R_G.

Theorem 6.1: For a graph G with $k(\ll n)$ communities where each community is generated by ER model with parameter $\frac{n}{k}$ and p, then R_G has an asymptotically normal distribution with mean $(n-2k)p+k$ and variance $2kp(1-p)$ where $p = \frac{2km}{n(n-k)}$.

Proof. In G each community has n/k nodes, and hence

$$p = \frac{2m}{k\frac{n}{k}(\frac{n}{k}-1)} = \frac{2km}{n(n-k)}.$$

Let λ_i be the largest eigenvalue of the ith community ($i = 1, 2, \ldots, k$), then $R_G = \sum_{i=1}^{k} \lambda_i$. Since λ_i has the asymptotical normal distribution with mean $(\frac{n}{k} - 2)p + 1$ and variance $2p(1-p)$[34], then R_G also has the asymptotical normal distribution with mean and variance as in the theorem. □

With Theorem 6.1, we directly have the following result.

Result 6.3: The relative non-randomness of the overall graph $G(n, m)$ can be cal-

culated as

$$R_G^* = \frac{R_G - [(n-2k)p + k]}{\sqrt{2kp(1-p)}}, \tag{6.6}$$

where $p = \frac{2km}{n(n-k)}$.

For any two graphs, G_1 and G_2, if $|R_{G_1}^*| < |R_{G_2}^*|$, we can conclude that G_1 is more random than G_2. Since the relative non-randomness measure R_G^* of ER graph approximately follows the standard normal distribution with mean 0 and standard variance 1, we can use $1 - \Phi(R_G^*)$ to indicate the similarity between this graph and a random graph, where $\Phi(x)$ denotes the cumulative distribution function of the standard normal distribution. Given a significance level α, when $R_G^* = \frac{R_G - [(n-2k)p+k]}{\sqrt{2kp(1-p)}} \geq \Phi^{-1}(1-\alpha)$, we can safely reject that G is a random graph.

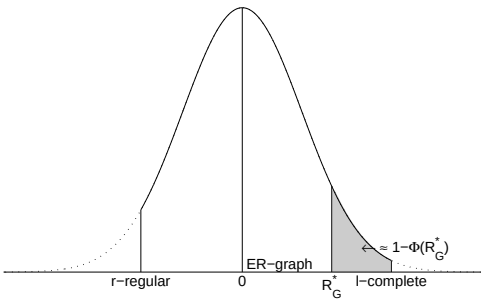


Figure 6.4: Relative non-randomness measure and its distribution

The relative measure indicates to what extent one real world graph is different from random graphs in terms of probability. As illustrated in Figure 6.4, when R_G^* is close to 0, the graph G tends to be more likely generated by ER model. From the statistical hypothesis testing point of view, we cannot reject the null hypothesis that G is generated by ER model. On the contrary, when R_G^* is far away from 0, it indicates the graph G is towards extreme ordered graph. We can safely reject the null hypothesis since $1 - \Phi(R_G^*)$ (denoted as the gray region in Figure 6.4) is significantly small.

Another interesting property illustrated in Figure 6.4 is that R_G^* of any graph is lower (upper) bounded by that of r-regular (l-complete) graph respectively. For graphs $G(n, m)$ with k communities, we define the r-regular graph as a graph with each node having r neighbors and the l-complete graph here as a graph where each community is a clique of l nodes.

Theorem 6.2: For any graph $G(n, m)$ with k communities, we have

$$R^*_{G_{r-regular}} \leq R^*_G \leq R^*_{G_{l-complete}}$$

where $R^*_{G_{r-regular}}$ and $R^*_{G_{l-complete}}$ denote the relative non-randomness value of r-regular graph and l-complete graph respectively. Similarly, we have

$$R_{G_{r-regular}} \leq R_G \leq R_{G_{l-complete}}$$

Their expressions are shown in Table 6.2.

Table 6.2: Non-randomness measure for different graphs with the same (n, m) and k communities

Graph, $p = \frac{2km}{n(n-k)}$	R_G	R_G^*
ER model	$(n-2k)p + k$	0
r-regular ($m = \frac{krn}{2}$)	kr	$-\frac{k}{\sqrt{2kp(1-p)}}$
l-complete ($m = \frac{kl(l-1)}{2}$)	$k(l-1)$	$\frac{kl-(n-2k)p-2k}{\sqrt{2kp(1-p)}}$

Proof. We first prove the case $k = 1$. When $k = 1$, $R_G = \lambda_1$. Let $d_{\min}$, $d_{\max}$ and $\bar{d}$ be the minimum, maximum, and the average degree. We have the following two inequalities [24]:

$$d_{\min} \leq \bar{d} = \frac{2m}{n} \leq \lambda_1 \leq d_{\max} \tag{6.7}$$

$$\lambda_1 \leq \sqrt{2m - n - 1} \tag{6.8}$$

Assume that $m = rn/2$ for some integer r, then we can construct a r-regular graph

with m edges. In r-regular graph $\bar{d} = \frac{2m}{n} = d_{\max} = r$, with Inequality (6.7), we have $R_G = \lambda_1 = r$. Since for any graph with the same parameters, we have $\lambda_1 \geq \frac{2m}{n}$. Hence the r-regular graph has the smallest non-randomness value.

The relative non-randomness measure is

$$R^*_{G_{r-regular}} = \frac{r - (n-2)p - 1}{\sqrt{2p(1-p)}}, \tag{6.9}$$

where $p = \frac{r}{n-1}$ for r-regular graph. When n is large, we can further simplify (6.9) as:

$$R^*_{G_{r-regular}} = -\frac{1}{\sqrt{2p(1-p)}}.$$

Assume that $m = \frac{l(l-1)}{2}$ for some integer l, then we can construct a complete graph with node $1, 2, \ldots, l$, leaving the rest nodes isolated. Then, $R_{G_{l-complete}} = l - 1$. Since any graph with the same parameters must involve no less than l non-isolated nodes, and with Inequality (6.8), we have

$$\lambda_1 \leq \sqrt{2m - l - 1} = l - 1.$$

Hence the l-complete graph reaches the upper bound. Its relative non-randomness is straightforwardly derived from the definition.

When $k > 1$, it is easy to verify that the minimum and maximum are reached when the graph has k equal-sized r-regular graphs or l-complete graphs. We have the theorem proved. □

Discussion. When it comes to a graph with one community, our graph non-randomness measure R_G is reduced as λ_1 as shown in 6.5. It has been shown in [34] that the largest eigenvalue has asymptotically the normal distribution with mean $(n-2)p + 1$ and variance $2p(1-p)$ when graph G follows ER model with parameter n and p. This can be considered as a special case of our results shown in Theorem 6.1.

Theorem 6.2 shows that r-regular graph and l-complete graph are most non-random graphs among all graphs $G(n, m)$. The relative non-randomness value of r-regular

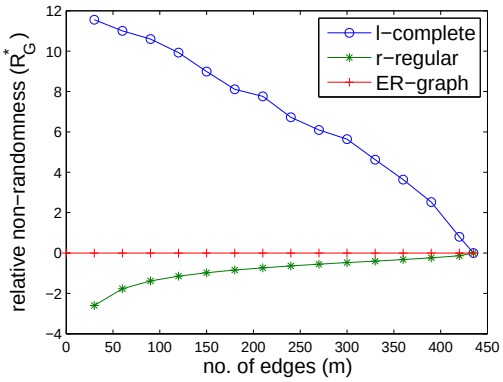


Figure 6.5: Upper and lower bounds of R^*_G for graphs with $n = 30$, $k = 1$, and varying m

graph reaches the largest negative value while that of l-complete graph reaches the largest positive value. Recall that the expectation of the relative non-randomness value of ER graphs is 0. Figure 6.5 illustrates how the relative non-randomness values of r-regular graph and l-complete graph vary when the density of graph increases. Note that the number of nodes across all graphs is fixed ($n = 30$). When we increase the number of edges, the range determined by the bounds decreases. In the extreme case of $m = 435$, both relative non-randomness values are zero since the graph is a fully complete graph.

6.2.4 Subgraph Non-randomness $R(G_1)$

Let G_1 denote a subgraph of $G(V, E)$ with node set $V_1 \subseteq V$ and edge set $E_1 \subseteq E$, $|V_1| = n_1$ and $|E_1| = m_1$. The subgraph non-randomness measure is $R(G_1) = \sum_{(u,v)\in E_1} R(u, v)$. It can indicate how much the subgraph G_1 contributes to the non-randomness of the whole graph.

For example, in the polbook network, a subgraph G_1 with 53 nodes and 230 edges (formed by choosing 50% of the nodes with the highest degrees and all the edges among them), $R(G_1) = 17.86$. Compared with the non-randomness of the whole graph $R_G = 23.55$, G_1 with only $\frac{230}{441} = 52.2\%$ edges accounts for $\frac{17.86}{23.55} = 75.8\%$ of the non-randomness of the whole graph. It indicates that G_1 makes a significant

contribution to the whole graph structure.

One natural question is what is the relationship between $R(G_1)$ and R_{G_1}. The later R_{G_1} denotes the graph non-randomness when we regard G_1 as an independent graph.

Lemma 6.1: Let G_1 be a subgraph of G, given the same k as G, we have $R(G_1) \leq R_{G_1}$.

Proof. First, we give a slightly different version of Corollary IV.4.4 in [90] as a lemma.

Lemma 6.2: Let $X \in R^{n\times k}$ have orthogonal columns. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of symmetric matrix A, and $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$ be the eigenvalues of $X^T AX$, then for $i = 1, 2, \ldots, k$, $\lambda_i \geq \mu_i$.

Let A_1 denote the adjacency matrix of G_1, and let μ_i be the i-th largest eigenvalue of A_1 with eigenvector $\boldsymbol{y}_i$. Then, we have

$$\begin{aligned} R(G_1) &= \sum_{(u,v)\in E_1} R(u,v) = \sum_{u,v\in V_1} a_{uv}\boldsymbol{\alpha_u}\boldsymbol{\alpha_v}^T \\ &= \text{trace}\left[X^T \begin{pmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} X\right] = \text{trace}(M). \end{aligned}$$

Let η_i $(i = 1, 2, \ldots, k)$ be the i-th largest eigenvalues of M. With Lemma 6.2, we know that $\eta_i \leq \mu_i$, then

$$R(G_1) = \text{trace}(M) = \sum_{i=1}^{k} \eta_i \leq \sum_{i=1}^{k} \mu_i = R_{G_1}.$$

We have Lemma 6.1 proved. □

Lemma 6.1 can be used to derive an upper bound of non-randomness of the whole graph G.

Result 6.4: We randomly select m_1 edges ($m_1 \leq m$) from the whole graph and form a subgraph G_1, then we have $R_G \leq \frac{m}{m_1} R_{G_1}$.

Proof. With Lemma 6.1, we have:

$$R_{G_1} \geq R(G_1) = \sum_{(u,v)\in E_1} R(u,v) \approx \frac{m_1}{m} \sum_{(u,v)\in E} R(u,v) = \frac{m_1}{m} R_G. \tag{6.10}$$

The approximation in (6.10) is because the m_1 edges are randomly selected, and $\sum_{(u,v)\in E_1} R(u,v)$ approaches $\frac{m_1}{m} \sum_{(u,v)\in E} R(u,v)$ as the graph size increases. Then, we immediately get $R_G \leq \frac{m}{m_1} R_{G_1}$. □

We can also derive a lower bound of R_G.

Result 6.5: Given a closed subgraph G_1, we have $R_G \geq R_{G_1}$. By closed subgraph, we mean that subgraph $G_1(V_1, E_1)$ satisfies $E_1 = E \cap (V_1 \times V_1)$.

Proof. To make the expression simple, we assume G_1 contains node $V_1 = \{1, 2, \ldots, n_1\}$. When G_1 is a closed subgraph, we can rewrite the symmetric adjacency matrix A as

$$A = \begin{pmatrix} A_1 & A_{12} \\ A_{12} & A_2 \end{pmatrix},$$

where A_2 is the adjacency matrix of the closed subgraph formed by node $\{n_1 + 1, \ldots, n\}$ and A_{12} represents the edges between these two subgraphs. Note λ_i is the i-th largest eigenvalue of A with eigenvector $\boldsymbol{x}_i$, and let $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k)$, then

$$R_{G_1} = \sum_{i=1}^{k} \mu_i = \text{trace}(Y^T A_1 Y) = \text{trace}\left[(Y^T \ \mathbf{0}) A \begin{pmatrix} Y \\ \mathbf{0} \end{pmatrix} \right].$$

With Lemma 6.2, we have

$$R_{G_1} = \sum_{i=1}^{k} \mu_i \leq \sum_{i=1}^{k} \lambda_i = R_G.$$

We have Result 6.5 proved. □

We can use this lower bound to determine whether the whole graph G is a random

one. Since

$$R_G^* = \frac{R_G - [(n-2k)p+k]}{\sqrt{2kp(1-p)}} \geq \frac{R_{G_1} - [(n-2k)p+k]}{\sqrt{2kp(1-p)}},$$

if $\frac{R_{G_1}-[(n-2k)p+k]}{\sqrt{2kp(1-p)}} \geq \Phi^{-1}(1-\alpha)$, we can reject that G is a random graph with the significance level α.

Recall the subgraph G_1 containing 50% of the nodes with the highest degrees in the polbook network, it is a closed subgraph with $R_{G_1} = 20.54$. Given the significance level $\alpha = 0.05$, the critical value is $r_0 = 19.69$. Since $R_G \geq R_{G_1} \geq r_0$, the whole graph G must be significantly different from a random one.

Results 6.4 and 6.5 can significantly reduce the computation cost for (non-)randomness testing of the overall graph G. R_G involves the calculation of the k largest eigenvalues of G. It generally takes $O(n^3)$ operations to compute eigenvalues and eigenvectors [38]. The derived upper and lower bounds only need to calculate R_{G_1} where G_1 is a much smaller graph.

6.3 Comparison with Other Graph Spectra

The graph spectrum has been well investigated in the graph analysis field. It has been shown that the eigenvectors of the Laplacian matrix and the normal matrix are also good indicators of community clusters [27, 72, 86, 97]. One important question is whether a similar non-randomness framework can also be derived using spectra of the Laplacian or normal matrix. In this section, we present our theoretical results and characterize differences among non-randomness measures derived using different spectra.

6.3.1 Laplacian Spectrum

Laplacian matrix of a graph is defined as $L = D - A$, where $D = \text{diag}\{d_1, d_2, \ldots, d_n\}$ and d_i is the degree of node i. Let μ_i be the i-th smallest eigenvalue of L with eigenvector $\boldsymbol{y}_i$. The smallest eigenvalue $\mu_1 = 0$ whose eigenvector is $\boldsymbol{y}_1 \equiv \mathbf{1}$. The eigenvectors of L are good indicators of the community structure. This fact can be

derived from the following minimization problem [84]: assign a k-dimensional vector $(p_{1u}, p_{2u} \ldots, p_{ku})$ to node u, so that the sum of the distances over the existing edges is minimized, i.e.,

$$\begin{aligned} &\min\ J_L(P) = \frac{1}{2}\sum_{u,v}\left\{a_{uv}\sum_{i=1}^{k}(p_{iu}-p_{iv})^2\right\} \\ &\text{s.t. } P^TP = I \end{aligned} \tag{6.11}$$

where $P = (\boldsymbol{p}_1|\boldsymbol{p}_2|\cdots|\boldsymbol{p}_k)$ and p_{iu} is the u-th entry of $\boldsymbol{p}_i$. The target function can be rewritten as $J_L(P) = \text{trace}(P^TLP)$. Then, taking $\boldsymbol{p}_i = \boldsymbol{y}_i$ gives the optimal solution, and the minimum value is given by $J(Y) = \sum_{i=1}^{k}\mu_i$, where $Y = (\boldsymbol{y}_1|\boldsymbol{y}_2|\cdots|\boldsymbol{y}_k)$.

$$\begin{array}{c} \\ \boldsymbol{\beta}_u \rightarrow \end{array} \begin{array}{c} \begin{array}{cccccc} \boldsymbol{y}_1 & & \boldsymbol{y}_i & & \boldsymbol{y}_k & \quad\boldsymbol{y}_n \end{array} \\ \begin{array}{cccccc} & & \downarrow & & & \end{array} \\ \left(\begin{array}{ccc|c|ccc} y_{11} & \cdots & y_{i1} & \cdots & y_{k1} & \cdots & y_{n1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ \hline y_{1u} & \cdots & y_{iu} & \cdots & y_{ku} & \cdots & y_{nu} \\ \hline \vdots & & \vdots & & \vdots & & \vdots \\ y_{1n} & \cdots & y_{in} & \cdots & y_{kn} & \cdots & y_{nn} \end{array}\right) \end{array} \tag{6.12}$$

With the same spirit of adjacency matrix, we can define $\boldsymbol{\beta} = (y_{1u}, y_{2u}, \ldots, y_{ku}) \in \mathbb{R}^{1\times k}$ to be the Laplacian matrix based spectral coordinates. $\boldsymbol{\beta}$ and $\boldsymbol{y}_i$ are shown in Formula (6.12). From (6.11), we know that the smaller distance between $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_v$ indicates the stronger community relation of node u and v, i.e., the edge is less likely to be a random one. The spectral geometry of $\boldsymbol{y}_2$ and $\boldsymbol{y}_3$ for the three-community synthetic graph in Figure 6.1(a) is shown in Figure 6.6(a). We neglect $\boldsymbol{y}_1$ for $\boldsymbol{y}_1 \equiv \mathbf{1}$. We can see that the three communities form three clusters in the spectral space with the bridging node at the middle, and those noise nodes are sparsely located. Therefore, we need to define the Laplacian matrix based edges non-randomness measure

via the Euclidean distance $\|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|$:

$$R^L(u,v) = c - \|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|_2^2. \tag{6.13}$$

We use a constant c to minus the squared Euclidean distance. A smaller non-randomness value indicates the edge is more likely to be a random one. Similarly, we can further derive the node non-randomness measure $R^L(u)$ as

$$\begin{aligned} R^L(u) = \sum_{v\in\Gamma(u)} R^L(u,v) &= cd_u - \sum_{v=1}^{n}\left\{a_{uv}\sum_{i=1}^{k}(y_{iu}-y_{iv})^2\right\} \\ &= cd_u - \sum_{i=1}^{k}\mu_i y_{iu}^2 + \sum_{i=1}^{k}\sum_{v=1}^{n} a_{uv}y_{iv}(y_{iu}-y_{iv}); \end{aligned} \tag{6.14}$$

and the graph non-randomness R_G^L as

$$\begin{aligned} R_G^L = \sum_{(u,v)\in E} R^L(u,v) &= 2cm - \sum_{u,v=1}^{n}\sum_{i=1}^{k} a_{u,v}(y_{iu}-y_{iv})^2 \\ &= 2cm - 2J_L(Y) = 2cm - 2\sum_{i=1}^{k}\mu_i. \end{aligned} \tag{6.15}$$

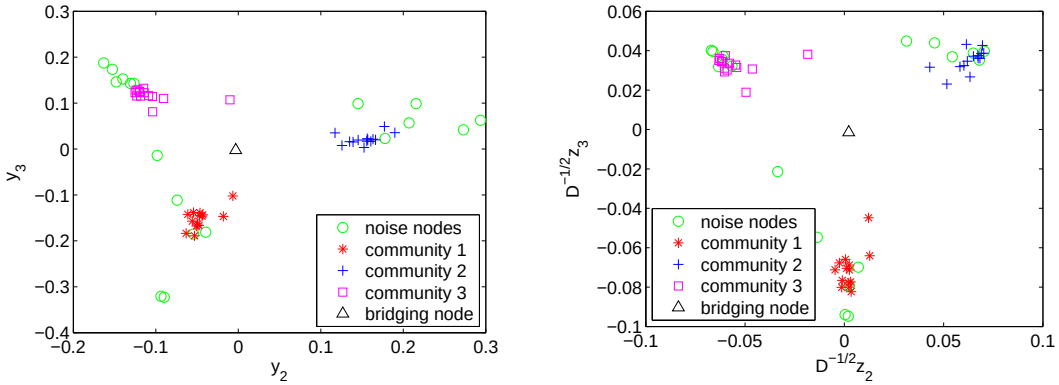
Then, from (6.15), we can see that the graph non-randomness measure is directly related to the eigenvalues of the Laplacian matrix. However, unlike the case of the adjacency matrix, the node non-randomness measure $R^L(u)$ defined in (6.14) does not have a concise expression. The third term of (6.14) contains the information of neighbor nodes, and we thus are unable to identify those non-random nodes simply via their vector lengths. From Figure 6.6(a), we can see those noise nodes are not particularly far from (or close to) the origin.

Another problem involved with the non-randomness measures is that it is difficult to choose the constant c properly. One idea is that choose c to be the maximum of $\|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|_2^2$. Obviously, $\|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|_2^2 \leq \|\boldsymbol{\beta}_u\|_2^2 + \|\boldsymbol{\beta}_v\|_2^2 \leq 2$, and when node u and v are two isolated nodes $\|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|_2^2 = 2$, and we can thus choose $c = 2$. However, the scales of the second and third term of (6.14) are usually small compared with

$2d_u$, then the node non-randomness measure based on the Laplacian matrix is almost solely determined by the node's degree, and the graph non-randomness measure is also almost determined by the first term $4m$.

The other extreme case is choosing $c = 0$, or equivalently $R^L(u, v) = \|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|_2^2$. This choice is also problematic. Consider the following two type of nodes: nodes connecting by many non-random edges (small edge non-randomness value), and nodes connecting by one random edge (large edge non-randomness value). These two types are very different: the former denotes the central node in the community, while the later is usually the noisy one. However, the node non-randomness values of these two types of nodes can be equal or very close, which makes impossible to distinguish them using the node non-randomness measure based on the Laplician spectrum.

We can also have some other ways to define the non-randomness measures, for example, define $R^L(u, v) = 1/\|\boldsymbol{\beta}_u - \boldsymbol{\beta}_v\|$. But they will not give a consistent frame work as the case of the adjacency matrix.



(a) L based (b) N based

Figure 6.6: Spectral geometry of the three-community synthetic network in Figure 6.1(a), based on Laplacian and normal matrix spectra.

6.3.2 Normal Spectrum

Normal matrix of a graph is defined as $N = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. Let ν_i be the largest eigenvalue of N, and $\boldsymbol{z}_i$ be its eigenvector. We have that the largest eigenvalue $\nu_1 = 1$

whose eigenvector $\boldsymbol{z}_1 = (\sqrt{d_1}, \ldots, \sqrt{d_n})^T$, $1 = \nu_1 \geq \nu_2 \geq \cdots \geq \nu_n \geq -1$, and $\nu_n = -1$ if and only if the graph is a bipartite one, in which case ν_i and $-\nu_i$ appear pairwise in the normal spectrum.

The relation between the normal spectrum and the graph structure can be shown in the same minimization problem as Problem (6.11) but with the constraint normalized by the nodes' degrees [84]:

$$\begin{aligned} &\min\ J_N(Q) = \frac{1}{2}\sum_{u,v}\left\{a_{uv}\sum_{i=1}^{k}(q_{iu} - q_{iv})^2\right\} \\ &\text{s.t. } Q^T D Q = I \end{aligned} \tag{6.16}$$

where $Q = (\boldsymbol{q}_1|\boldsymbol{q}_2|\cdots|\boldsymbol{q}_k)$ and q_{iu} is the u-th entry of $\boldsymbol{q}_i$. With the method of Lagrange multipliers and setting the derivatives equal 0, the necessary condition for optimality is given by $(D-A)\boldsymbol{q}_i = \zeta_i D\boldsymbol{q}_i$, where ζ_i is the Lagrange multiplier. With some simple deduction, we have $ND^{\frac{1}{2}}\boldsymbol{q}_i = (1-\zeta_i)D^{\frac{1}{2}}\boldsymbol{q}_i$. Therefore, the solution is given by $\zeta_i = 1 - \nu_i$ and $\boldsymbol{q}_i = D^{-\frac{1}{2}}\boldsymbol{z}_i$. The constraint in Problem (6.16) is naturally satisfied, and the minimum value is givn by $J_N(D^{-\frac{1}{2}}Z) = k - \sum_{i=1}^{k}\nu_i$, where $Z = (\boldsymbol{z}_1|\boldsymbol{z}_2|\cdots|\boldsymbol{z}_k)$.

$$\begin{array}{c} \\ \\ \boldsymbol{\gamma}_u \rightarrow \end{array} \begin{array}{c} \overset{D^{-\frac{1}{2}}\boldsymbol{z}_i}{\downarrow} \\ \left(\begin{array}{ccc|c|cc|ccc} \frac{z_{11}}{\sqrt{d_1}} & \cdots & & \frac{z_{i1}}{\sqrt{d_1}} & \cdots & \frac{z_{k1}}{\sqrt{d_1}} & \cdots & \frac{z_{n1}}{\sqrt{d_1}} \\ \vdots & & & \vdots & & \vdots & & \vdots \\ \hline \frac{z_{1u}}{\sqrt{d_u}} & \cdots & & \frac{z_{iu}}{\sqrt{d_u}} & \cdots & \frac{z_{ku}}{\sqrt{d_u}} & \cdots & \frac{z_{nu}}{\sqrt{d_u}} \\ \hline \vdots & & & \vdots & & \vdots & & \vdots \\ \frac{z_{1n}}{\sqrt{d_n}} & \cdots & & \frac{z_{in}}{\sqrt{d_n}} & \cdots & \frac{z_{kn}}{\sqrt{d_n}} & \cdots & \frac{z_{nn}}{\sqrt{d_n}} \end{array}\right) \end{array} \tag{6.17}$$

Since $\boldsymbol{q}_i = D^{-\frac{1}{2}}\boldsymbol{z}_i$, with the same logic as the case of Laplacian matrix, we need to define the normal matrix based spectral coordinate $\boldsymbol{\gamma}_u = \frac{1}{\sqrt{d_u}}(z_{1u}, z_{2u}, \ldots, z_{ku})$ as shown in Formula (6.17). Figure 6.6(b) shows the spectral geometry of $D^{-\frac{1}{2}}\boldsymbol{z}_2$ and $D^{-\frac{1}{2}}\boldsymbol{z}_3$ for the three-community synthetic graph, $D^{-\frac{1}{2}}\boldsymbol{z}_1 \equiv \mathbf{1}$. Similarly as the

Laplacian geometry, the three communities form three clusters with the bridging node in the middle, and the noisy nodes are scattered sparsely.

Similar to the case of Laplacian matrix, we can define the edge non-randomness measure based on normal matrix via the Euclidean distance:

$$R^N(u,v) = c - \|\boldsymbol{\gamma}_u - \boldsymbol{\gamma}_v\|_2^2. \tag{6.18}$$

We further define the node non-randomness measure as

$$\begin{aligned} R^N(u) = \sum_{v\in\Gamma(u)} R^N(u,v) &= cd_u - \sum_{v=1}^{n}\left\{a_{uv}\sum_{i=1}^{k}\left(\frac{z_{iu}}{\sqrt{d_u}} - \frac{z_{iv}}{\sqrt{d_v}}\right)^2\right\} \\ &= cd_u + \sum_{i=1}^{k}(2\nu_i - 1)z_{iu}^2 - \sum_{i=1}^{k}\sum_{v=1}^{n}\frac{a_{uv}z_{iv}^2}{d_v}; \end{aligned} \tag{6.19}$$

and the graph non-randomness measure as

$$\begin{aligned} R_G^N = \sum_{(u,v)\in E} R^N(u,v) &= 2cm - \sum_{u,v} a_{uv}\left\{\sum_{i=1}^{k}(\frac{z_{iu}}{\sqrt{d_u}} - \frac{z_{iv}}{\sqrt{d_v}})^2\right\} \\ &= 2cm - 2J_N(D^{-\frac{1}{2}}Z) = 2cm - 2k + 2\sum_{i=1}^{k}\nu_i. \end{aligned} \tag{6.20}$$

Similar as the case of the Laplacian matrix, $R^N(u)$ does not have a concise expression, and it is difficult to choose the constant c.

6.3.3 Modularity

Define the modularity matrix B as $b_{uv} = a_{uv} - \frac{d_u d_v}{2m}$, and let η_i be the i-th largest eigenvalue of B with eigenvector $\boldsymbol{s}_i$. The spectrum of B also has a close relation with the graph community structure. This is because finding the best community partition to maximize modularity Q can be written as follows:

$$\begin{aligned} &\max Q = \frac{1}{2m}\,\mathrm{trace}(J^T B J) \\ &\text{s.t. } J^T J \text{ is diagonal and } \mathrm{trace}(J^T J) = n \end{aligned} \tag{6.21}$$

where J is the $n \times k$ index matrix: $J_{ij} = 1$ if node i belongs to community j and 0 otherwise. Relaxing the 0-1 constraint, we have the i-th column of J is $\boldsymbol{s}_i$ except for the vector length, and $\max Q = \sum_{i=1}^{k} \eta_i$, suppressing a multiplicative constant. We can also similarly define the spectral coordinate based on modularity matrix B as $\boldsymbol{\delta}_u = (s_{1u}, s_{2u}, \ldots, s_{ku})$, and [74] suggests that the direction of $\boldsymbol{\delta}_u$ indicates the community partition. However, generally speaking, communities forms neither orthogonal lines nor clusters in the k-dimensional spectral space, and hence defining edge non-randomness measure via inner product or Euclidean distance is impropriate. Therefore, it is difficult to define a consistent framework based on matrix B.

It is worth pointing out that the authors in [74] defined the community centrality of node u (denoted by $CC(u)$ in our paper) to measure the node's contribution to the community structure, and $CC(u) = (\sum_{i=1}^{k} \eta_j s_{iu}^2)^{\frac{1}{2}}$. Then, $Q = \sum_{u=1}^{n} CC(u)^2$. The community centrality measure based on the modularity matrix B has a similar concise expression as our node non-randomness measure based on the adjacency matrix A. However, the community centrality actually measures to what extent the node's contribution to the community structure exceeds its expected value [74] while our node non-randomness incorporates randomness values of all its connected edges.

In the synthetic graph shown in Figure 6.1(a), the bridging node (node 61) has its community centrality value equal to 0.084, which ranks 57 among the 61 nodes. This means that the bridging nodes makes little contribution to the modularity Q. We can see that the the community centrality does not take the bridging effect into account, and is thus unable to distinguish the bridging nodes from the noise nodes. However, our node non-randomness can be used to separate the the bridging node from noise ones ($R(u) = 0.659$ with rank 16, much higher than those noise nodes).

6.4 Empirical Evaluations

Data Sets. We used several network data sets in our evaluation, *polbooks*, *polblogs*, *dolphins*, *karate*, *netsci*, and *Enron*. We also generated two synthetic graph with

the same size: synthetic-1 with only one community that is generated using the ER model with parameters $n = 1000$ and $p = 0.2$; and synthetic-1 with two disconnected communities each of which is generated via ER model and has 500 nodes and 49910 edges.

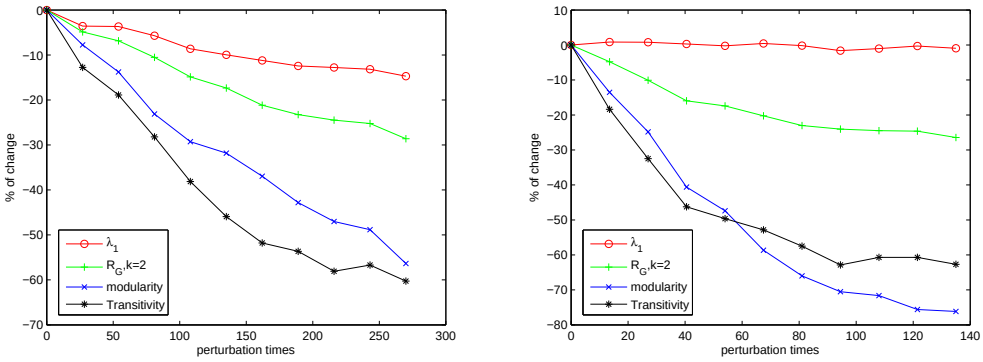
In this section, we focus on graph non-randomness of both synthetic networks and real social networks. We have also analyzed how edge non-randomness and node non-randomness distribute in real-world social networks and random graphs. Our results show that edge non-randomness and node non-randomness of real-world social networks usually display some high skewed distributions, obeying either a power law or an exponential law. On the contrary, random graphs display approximate normal distributions.

Graph Non-randomness of Various Social Networks. Table 6.3 shows graph statistics, and graph non-randomness values (calculated using R_G and R_G^*) of various social networks. We can observe that the relative non-randomness measures (R_G^*)) of real world social networks are significantly greater than zero while that of the synthetic random graph is very close to zero. Using R_G^*, we can relatively compare the randomness of graphs with different sizes and densities. For example, we can observe that the network of the dolphins contains less randomness than the karate data since R_G^* of the dolphins (1.61) is greater than that of the karate data (1.22). Furthermore, R_G^* also indicates to what extent the graph is different from random graphs. For karate graph, we have $R_G^* = 1.22$ and $1-\Phi(R_G^*) = 0.11$, which indicates how less likely the karate graph is generated by ER model. Similarly, for dolphins data, we have $R_G^* = 1.61$ and $1 - \Phi(R_G^*) = 0.054$.

We are also concerned with the connection between various real graph characteristics and our graph non-randomness measure (which is derived from graph spectrum). We conducted two types of perturbations on politics book: addition/deletion of randomly chosen edges, and switches of edges. For each perturbed graph, we calculated

Table 6.3: Graph non-randomness and characteristics of various social networks

Network	n	m	Q	R_G	R_G^*
synthetic	1000	99820	0.06	200	0.02
karate	34	78	0.44	11.7	1.22
dolphins	62	159	0.54	13.1	1.61
polbooks	105	441	0.53	23.5	6.87
Enron	151	869	0.51	41.2	4.18
polblogs	1222	16714	0.80	134	187
netsci	1589	2742	0.92	38.5	128



(a) Edge Add/Delete (b) Edge Switch

Figure 6.7: Graph characteristic vs. non-randomness measure for politics book network with various perturbations.

λ_1, R_G with $k = 2$, and two real graph characteristics: transitivity C and Modularity Q. The transitivity measure, C, is one type of clustering coefficient measure and characterizes the presence of local loops near a vertex. It is formally defined as $C = \frac{3N_\Delta}{N_3}$ where N_Δ is the number of triangles and N_3 is the number of connected triples.

Intuitively, when the magnitude of perturbation increases, we expect the graph tends to lose its structural properties. Figure 6.7(a) and 6.7(b) show our empirical evaluations on how Transitivity, Modularity, λ_1, and R_G are changed along perturbations on politics book. Furthermore, our graph non-randomness measure R_G can better reflect the change trend indicated by Transitivity and Modularity than λ_1. For example, in Figure 6.7(b), λ_1 remains almost unchanged even when the graph is significantly perturbed by random switches.

One interesting phenomenon is that the relative non-randomness measure always decreases when the magnitude of perturbations increases. Formally, we have the following theorem.

Theorem 6.3: Let Graph $G(n,m)$ be a graph with k communities and $p = \frac{2m}{n(n-1)} < \frac{1}{2}$, and graph is G' obtained by randomly adding edges to G: each non-existing edge is to be added with probability Δp, $\Delta p < p$ and $p + \Delta p < \frac{1}{2}$. Assume k communities will not merge. If $R_G - [(n-2k)p + k] \in O(pn)$, we have $E(R^*_{G'}) < R^*_G$, as the graph becomes large.

Proof. Let A and $\widetilde{A}$ be the adjacency matrix of G and G' respectively, $E = \widetilde{A} - A$. Let λ_i, $\tilde{\lambda}_i$ and ϵ_i be the i-th largest eigenvalue of A, $\widetilde{A}$ and E respectively. With Theorem IV-4.8 in [90], we have

$$R_{G'} = \sum_{i=1}^{k} \tilde{\lambda}_i \leq \sum_{i=1}^{k} \lambda_i + \sum_{i=1}^{k} \epsilon_i = R_G + \sum_{i=1}^{k} \epsilon_i,$$

and hence

$$E(R_{G'}) \leq R_G + E\left(\sum_{i=1}^{k} \epsilon_i\right). \tag{6.22}$$

We know that $E(\epsilon_1) = (n-2)\Delta p + 1$ and with the Semicircle Law [32], we have

$$E(\epsilon_i) \leq 2\sqrt{n\Delta p(1-\Delta p)}, i = 2, 3, \ldots, k. \tag{6.23}$$

Combining (6.6), (6.22) and (6.23), we have

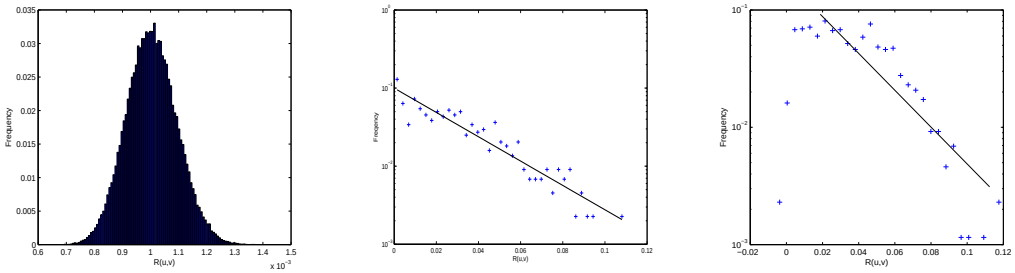
$$\begin{aligned}
E(R^*_{G'}) &= \frac{E(R^*_{G'}) - [(n-2k)(p+\Delta p) + k]}{\sqrt{2k(p+\Delta p)(1-p-\Delta p)}} \\
&\leq \frac{R_G + (n-2)\Delta p + 1 + 2(k-1)\sqrt{n\Delta p(1-\Delta p)}}{\sqrt{2k(p+\Delta p)(1-p-\Delta p)}} \\
&- \frac{(n-2k)(p+\Delta p) + k}{\sqrt{2k(p+\Delta p)(1-p-\Delta p)}} \\
&\leq \frac{R_G - [(n+2k)p + k] + M}{\sqrt{2k(p+\Delta p)(1-p-\Delta p)}} \\
&\quad (\text{let } M = 2(k-1)\left[\Delta p + \sqrt{n\Delta p(1-\Delta p)}\right] + 1)
\end{aligned}$$

Hence, to prove $E(R^*_{G'}) < R^*_G$, we need only to show

$$\frac{R_G - [(n+2k)p + k] + M}{R_G - [(n-2k)p + k]} < \frac{\sqrt{(p+\Delta p)(1-p-\Delta p)}}{\sqrt{p(1-p)}}. \tag{6.24}$$

Since $R_G - [(n+2k)p + k] \in O(pn)$ while $M \in O(\sqrt{\Delta pn})$, when n is large, the left-hand side of Inequality (6.24) is close to 1. Notice that $p < p + \Delta p < \frac{1}{2}$, the right-hand side of Inequality (6.24) is greater than 1 regardless of n, then when n goes large, we must have Inequality (6.24) stands. □

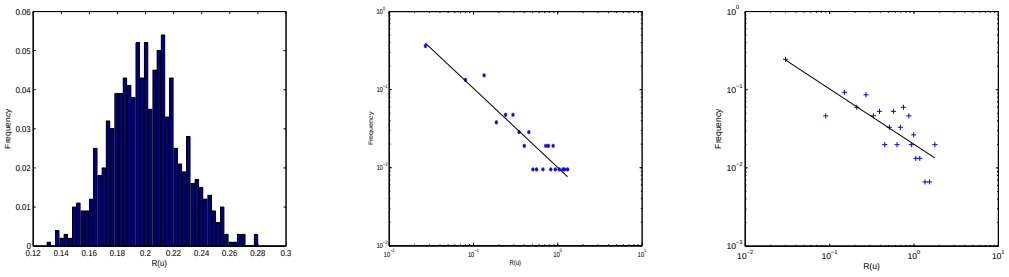
Distributions of Node Non-randomness and Edge Non-randomness. It is well known that the degree distributions in many real-world networks, such as the power-law distribution observed for the Internet and the the Web graph, differ significantly from the Poisson distribution of random graphs [16]. We are interested in the edge non-randomness distribution as well as the node non-randomness distribution in real-world networks and how they are different from synthetic random networks generated by the ER model.



(a) ER synthetic (b) Politics books (c) Enron

Figure 6.8: Edge non-randomness distributions

We conducted experiments using three networks: synthetic data generated by the ER model with $n = 1000$ and $p = 0.2$, politics books, and Enron network. Figure 6.8 (Figure 6.9) shows distributions of edge (node) non-randomness of these three networks. We can observe from Figure 6.8(a) and Figure 6.9(a) that the distributions of both edge non-randomness and node non-randomness follow approximately normal distributions.



(a) ER synthetic (b) Politics books (c) Enron

Figure 6.9: Node non-randomness distributions

The linear-log plot in Figure 6.8(b) indicates that edge non-randomness $R(u, v)$ of 441 edges in politics book has a highly skewed form, approximately obeying an exponential law. However, Figure 6.8(c) shows that edge non-randomness $R(u, v)$ of the majority edges in Enron email network only approximately obeys an exponential law. The log-log plot in Figure 6.9(b) indicates that node non-randomness $R(u)$ of 105 nodes in politics book clearly follows a power law distribution. However, there is no evidence to display the power law pattern for node non-randomness distribution of 151 nodes in Enron data as shown in Figure 6.9(c).

The distributions of both edge non-randomness and node non-randomness for real networks are quite different from those for random graphs. We also conducted evaluations on other real-world social networks. Although we cannot reach the conclusion that they definitely follow power law (or exponential law) distributions, our empirical evaluations did show that they are usually highly skewed, with a small number of edges (nodes) having an unusually large non-randomness values and a large number of edges (nodes) having small non-randomness values.

The Effect of k. In Section 6.2, we have shown that the graph non-randomness measure is determined by the sum of the first k eigenvalues, where k indicates the number of communities in the graph. In this experiment, we are interested in how different choices of k affect the graph non-randomness. We used the Enrol network and perturbed it by randomly adding/deleting edges. For each perturbed graph,

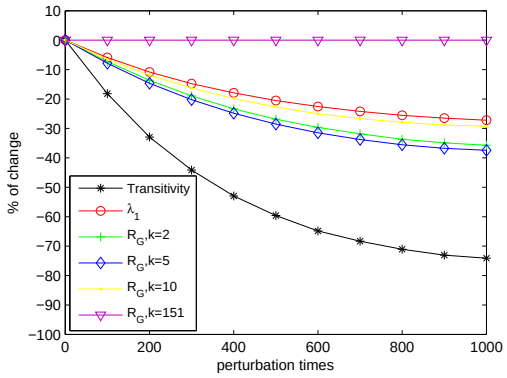


Figure 6.10: graph characteristics vs. non-randomness measures for Enron with Add/Delete perturbations

we calculated one real graph characteristics, the Transitivity measure C, the first eigenvalue λ_1, and our non-randomness measure R_G with three different k values (5, 50, 151). Note that there exist five roughly separated groups in Enron network. We expected R_G with $k = 5$ should best match the trend characterized by Transitivity measure.

Figure 6.10 shows our experiment results. We can observe that λ_1 did not match the perturbations as well as R_G with $k = 5$. R_G with $k = 50$ and $k = 151$ are totally unmatched with the perturbations. Specifically, R_G with $k = 151$ remains unchanged across all perturbed graphs while R_G with $k = 50$ displayed opposite trend from that suggested by Transitivity measure. We have shown in Section 6.2 that R_G is always zero when k equals the number of nodes n.

Evolution of Graph Non-randomness. We are interested in how the graph non-randomness may change for dynamic social networks. We performed the randomness analysis on the monthly email graphs from Enron data. In Table 6.4, we list graph relative non-randomness values for 12 graphs constructed from Enron dataset from June 2001 to May 2002. Each graph G_t is formed by the total email data in months from 1 to t. We regard there's an edge between node u and v in G_t when there is at least three communications between u and v during this period. We use m_t to denote

Table 6.4: Enron dynamic relative non-randomness ($k = 3$)

	m_t	$R^*_{G_t}$	$R^*_{H_t}$	$R^*_{G_t} - R^*_{G_{t-1}}$	$R^*_{H_t} - R^*_{G_{t-1}}$
G_1	87	12.99	–	–	–
G_2	187	12.41	4.84	−0.58	−8.16
G_3	327	12.45	4.07	0.04	−8.35
G_4	429	9.81	7.14	−2.63	−5.31
G_5	627	7.89	2.01	−1.92	−7.81
G_6	726	7.22	4.29	−0.67	−3.60
G_7	765	7.12	5.82	−0.10	−1.40
G_8	805	5.91	5.74	−1.20	−1.38
G_9	826	5.69	5.22	−0.22	−0.70
G_{10}	851	5.19	4.85	−0.51	−0.84
G_{11}	879	4.88	4.27	−0.31	−0.92
G_{12}	922	4.36	3.56	−0.52	−1.32

Table 6.5: Comparison of top k non-random nodes across monthly networks of Enron data

$\mid S_t \mid$	J_t										
	2	3	4	5	6	7	8	9	10	11	12
10	0.67	0.67	0.54	0.67	0.54	0.43	0.43	0.82	0.54	0.54	0.67
20	0.60	0.60	0.54	0.74	0.60	0.33	0.38	0.60	0.54	0.60	0.67
30	0.71	0.58	0.58	0.82	0.62	0.54	0.58	0.71	0.62	0.54	0.76
40	0.63	0.63	0.63	0.82	0.74	0.57	0.57	0.78	0.57	0.57	0.74
50	0.69	0.64	0.67	0.85	0.75	0.56	0.69	0.85	0.69	0.64	0.85

the number of edges of G_t. We can easily observe that $G_{t-1} \subseteq G_t$ and $m_{t-1} < m_t$. We use $R^*_{G_t}$ $(k = 3)$ to denote the relative non-randomness of G_t. We can observe that for most real Enron data sets, $R^*_{G_t} < R^*_{G_{t-1}}$ (except G_3), showing that the relative non-randomness of the graph decreases along the time.

One interesting question here is how those newly added edges in each month are different from randomly added edges. To answer this question, we construct synthetic data sets H_t by randomly adding $m_t - m_{t-1}$ edges to G_{t-1}. We can see in Table 6.4 that $R^*_{H_t}$ is always less than $R^*_{G_{t-1}}$ since the randomly added edges increase the graph non-randomness. Specifically, the 39 newly added edges in the real graph G_7 decreases the relative non-randomness by 0.10. However, when we randomly add 39 edges to G_6, the relative non-randomness of H_7 decreases by 1.40. This difference indicates those 39 newly added edges in G_7 are significantly different from randomly added

edges. On the contrary, 40 newly added edges in G_8 are not significantly different from randomly added edges.

Since the number of nodes are unchanged across all monthly graphs, we are also interested in how the subset of individuals identified as top non-random nodes (e.g., top 30) is varied dynamically. We used Jaccard's index measuring the similarity between two subsets. Formally, we define

$$J_t = \frac{\mid S_{t-1} \cap S_t \mid}{\mid S_{t-1} \cup S_t \mid}$$

where S_t denotes the subset of non-randomness nodes from data G_t. We can observe from Table 6.5 that those non-random nodes do change along the time. Hence, our node non-randomness measure $R(u)$ can be applied in practice to monitor the change of individual's roles in terms of its randomness in the social network.

6.5 Adjacency Cut via Line Fitting

In this section, we present a novel graph partition algorithm, *AdjCut*, which utilizes the line orthogonality pattern in the spectral space of the adjacency matrix. Our idea is to fit node spectral coordinates with the k orthogonal lines in the k-dimensional spectral space. Our algorithm is different from traditional spectral partition algorithms [17, 27, 43, 50, 76, 78, 86] that utilize the cluster pattern in the spectral space of Laplacian or normal matrix.

6.5.1 Problem Formalization

For a fixed k, let the unit row vector $\boldsymbol{l}_i = (l_{i1}, \ldots, l_{ik})$, $i = 1, \ldots, k$, denote the k orthogonal fitted lines. Each line corresponds to a community in the graph. If $\boldsymbol{l}_i$ is well fitted, the spectral coordinate $\boldsymbol{\alpha}_u$ should be close to the line corresponding to the community the node belongs to. When we project each node to its closest line, the length of the projection vector $\widehat{\boldsymbol{\alpha}}_u$ should be large as shown in Figure 6.11. Hence, we can estimate the k orthogonal lines by maximizing the total sum of projection

lengths: $J = \sum_{u=1}^{n} \|\widehat{\boldsymbol{\alpha}}_u\|_2$.

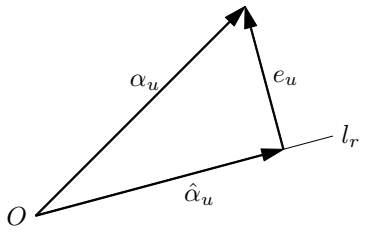


Figure 6.11: Vectors: $\boldsymbol{l}_r$, $\boldsymbol{\alpha}_u$, $\hat{\boldsymbol{\alpha}}_u$ and $\boldsymbol{e}_u$

Suppose $\boldsymbol{l}_r$ is the line closest to a certain point $\boldsymbol{\alpha}_u$. Then, the projection vector is $\widehat{\boldsymbol{\alpha}}_u = (\boldsymbol{l}_r \boldsymbol{\alpha}_u^T)\boldsymbol{l}_r$ and $|\boldsymbol{l}_i \boldsymbol{\alpha}_u^T|$ is the projection length. We denote $\boldsymbol{e}_u$ the residual vector $\boldsymbol{e}_u = \boldsymbol{\alpha}_u - \widehat{\boldsymbol{\alpha}}_u$. Moreover, $\|\widehat{\boldsymbol{\alpha}}_u\|_2$ should be the largest among all $|\boldsymbol{l}_i \boldsymbol{\alpha}_u^T|$, i.e, $r = \arg\max_i |\boldsymbol{l}_i \boldsymbol{\alpha}_u^T|$. Then the objective function can be written as follows:

$$J = \sum_{u=1}^{n} \|\widehat{\boldsymbol{\alpha}}_u\|_2 = \sum_{u=1}^{n} \sum_{i=1}^{k} h_{ui} \boldsymbol{l}_i \boldsymbol{\alpha}^T,$$

where $h_{ui} = \text{sign}(\boldsymbol{l}_i \boldsymbol{\alpha}_u^T)$ if $i = r$ and 0 otherwise.

Let L be the matrix of k lines: the i-th row of L is $\boldsymbol{l}_i$, $Y_k = (\boldsymbol{y}_1 | \boldsymbol{y}_2 | \cdots | \boldsymbol{y}_k)$, and let $H = (h_{ui})_{n \times k}$. Our optimization problem can be formalized as

$$\begin{aligned} &\max_{H,L} J = \text{trace}(L Y_k^T H) \\ &\text{s.t. } h_{ij} \in \{-1, 0, 1\},\ L^T L = I_k. \end{aligned} \tag{6.25}$$

The constraint $L^T L = I_k$ guarantees that the k fitted lines are orthogonal to each other. H naturally indicates the community partition: node u is assigned to C_i if h_{ui} is non-zero.

Once the lines are properly fitted, we can determine bridging nodes by examining their locations with respect to the k fitted lines. Bridging nodes are those ones connecting to multiple communities. They usually lie on the boundaries of communities, bridging gaps between otherwise disconnected groups. In the spectral space, they are neither close to the origin, nor close to any orthogonal fitted line corresponding to a

certain community. Therefore, we can mark a node as a bridging node if $\|\boldsymbol{\alpha}_u\|_2 \geq \tau_1$ and $\cos(\boldsymbol{\alpha}_u, \widehat{\boldsymbol{\alpha}}_u) = \|\widehat{\boldsymbol{\alpha}}_u\|_2 / \|\boldsymbol{\alpha}_u\|_2 \leq \tau_2$, where τ_1 and τ_2 are some thresholds.

6.5.2 Fitting k Orthogonal Lines

For the optimization problem shown in (6.25), it is difficult to obtain the optimal H and L simultaneously. Here we present an iterative algorithm to solve this optimization problem by fixing H or L and solving for the other matrix in each step, as outlined in Algorithm 6.

Algorithm 6 *AdjCut*: Fitting k lines

1: $t = 0$, Initiate $L^{(0)}$;
2: **while** not converge **do**
3: **for** $u = 1, 2, \ldots, n$ **do**
4: $r = \arg\max_i |\boldsymbol{l}_i^{(t)} \boldsymbol{\alpha}_u^T|$;
5: $h_{ur}^{(t)} = \text{sign}\left(\boldsymbol{l}_r^{(t)} \boldsymbol{\alpha}_u^T\right)$, and $h_{ui}^{(t)} = 0$ for $i \neq r$;
6: **end for**
7: $USV^T = Y_k^T H$, and $L^{(t+1)} = VU^T$;
8: $t = t + 1$;
9: **end while**

When L is fixed, the loop from Line 3 to 6 determines the optimal H. Given k lines determined at step t, it assigns each node to the line closest to it. When H is fixed, we have

$$J = \text{trace}(LY_k^T H) = \text{trace}(LM),$$

where we write $M = Y_k^T H$. Let $M = USV^T$ be the SVD of matrix M, where $S = \text{diag}(\sigma_1, \ldots, \sigma_k)$ and σ_i is the singular value of M. Then we have

$$J = \text{trace}(LM) = \text{trace}(LUSV^T) = \text{trace}(SV^T LU) \leq \sum_{i=1}^{k} \sigma_i, \tag{6.26}$$

and when $L = VU^T$, J reaches the maximum value in (6.26) (Line 7). During the iterations, the objective function $J^{(t)} = \text{trace}(L^{(t)} Y_k^T H^{(t)})$ is non-decreasing, and the process hence converges to a local maximal.

Calculation of the eigenvectors of an $n \times n$ matrix takes in general a number of

operations $O(n^3)$, which is almost inapplicable for large networks. However, in our framework, we only need to calculate the first k eigen-pairs. Furthermore, adjacency matrices in our context are usually sparse. The Arnoldi/Lanczos algorithm [38] generally needs $O(n)$ rather than $O(n^2)$ floating point operations at each iteration. The cost of our k orthogonal line fitting algorithm is $O(k^3 n)$.

Initiate L. To have a proper initial L, we can apply the greedy search in the spectral space, as shown in Procedure 2. We start with the searching subspace equal to the full k-dimensional space. At iteration j, we pick up the vector with the largest length in the searching space, normalize its length, and get $\boldsymbol{l}_j$ (Line 2). The new searching space is then the subspace orthogonal to $\boldsymbol{l}_j$ and all vectors are projected to the new searching space via the Graham-Schmidt process (Line 3). Apparently, the solution L from the greedy search is an orthogonal matrix. When the spectral coordinates form clear k quasi-orthogonal lines in the spectral space, the k fitted lines found by the greedy algorithm are already close to the optimal solution, and the convergence can then be very fast.

Procedure 2 Greedy Search for L

1: **for** $j = 1, \ldots, k$ **do**
2: $\quad s = \arg\max_i \|\boldsymbol{\alpha}_i\|_2$, and $\boldsymbol{l}_j = \boldsymbol{\alpha}_s^T / \|\boldsymbol{\alpha}_s\|_2$;
3: $\quad \boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i - (\boldsymbol{\alpha}_i \boldsymbol{l}_j)\boldsymbol{l}_j^T$ for $i = 1, \ldots, n$;
4: **end for**

Determining Proper k. The objective function J is not appropriate in comparing the goodness of fit when k varies. This is because the error length generally increases as the dimension k increases. Next we propose the use of the normalized error to determine proper k. Formally, we define the measure as

$$\rho = \frac{\sum_{u=1}^{n} \|\boldsymbol{e}_u\|_2^2}{(k-1)\sum_{u=1}^{n} \|\boldsymbol{\alpha}_u\|_2^2}. \tag{6.27}$$

The measure is normalized by $k-1$ since $\boldsymbol{e}_u$ has the degree of freedom $k-1$. Once we fit the data by k orthogonal lines, we can calculate the statistic ρ. When $k = 1$,

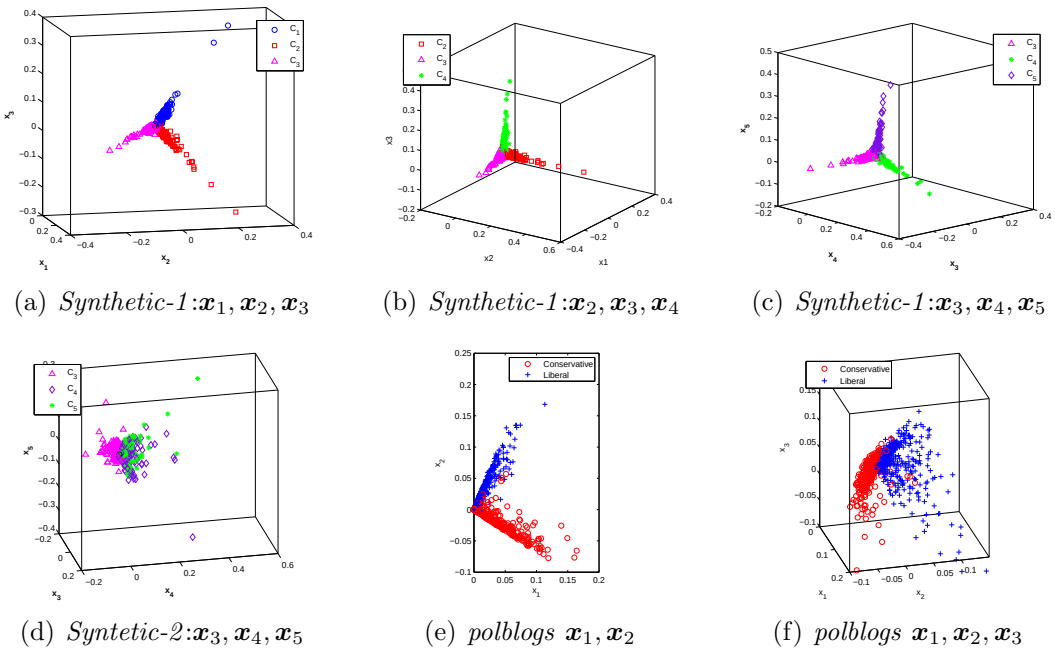
ρ is always 0 since we do the projection in a 1-dimensional subspace. For, $k \geq 2$, we have $0 \leq \rho \leq 1$. When the graph contains k clear communities, spectral coordinates form k quasi-orthogonal lines in the k-dimensional subspace. We can find the k orthogonal lines that well fit the data. In this case ρ should be close to 0. However, in the subspace spanned by less or more eigenvectors, the coordinates scatter from those lines, and we will not obtain a very good fit of the data. So our strategy is to determine a k value that incurs a low ρ value.

6.5.3 Evaluation of Adjacency Cut Algorithm

Data Sets. We use several real network data sets in our evaluation: *polbooks*, *polblogs*, *Enron* and *Facebook*. We also generate two synthetic graphs: *Synthetic-1* and *Synthetic-2*. The *Synthetic-1* has 5 communities with the number of nodes 200, 180, 170, 150, and 140 respectively, and each community is generated separately with a power law degree distribution with the parameter 2.3 [3]. We add cross community edges randomly and keep the ratio between inter-community edges and inner-community edges as 20% in *Synthetic-1*. *Synthetic-2* is the same as the *Synthetic-1* except that we increase the number of links between community C_4 and C_5 to 80%. As a result, the *Synthetic-2* has four communities. Table 6.6 shows the statistics of these data sets.

Line Orthogonality Property. We first check how line orthogonality property holds in various networks. We can clearly observe from Figures 6.12(a), 6.12(b), and 6.12(c) that there exist five orthogonal lines in the spectral space spanned by $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_5$ and nodes from the same community (denoted by different colors) lie on the same line for *Synthetic-1*. For *Synthetic-2*, we are particularly interested in the subspace spanned by $\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$. As shown in Figure 6.12(d), we cannot observe any clear line orthogonality pattern, which demonstrates our theoretical results since there

[3] Real social network data with one major component usually follow a power law degree distribution with the parameter between 2 to 3.48 according to [5].



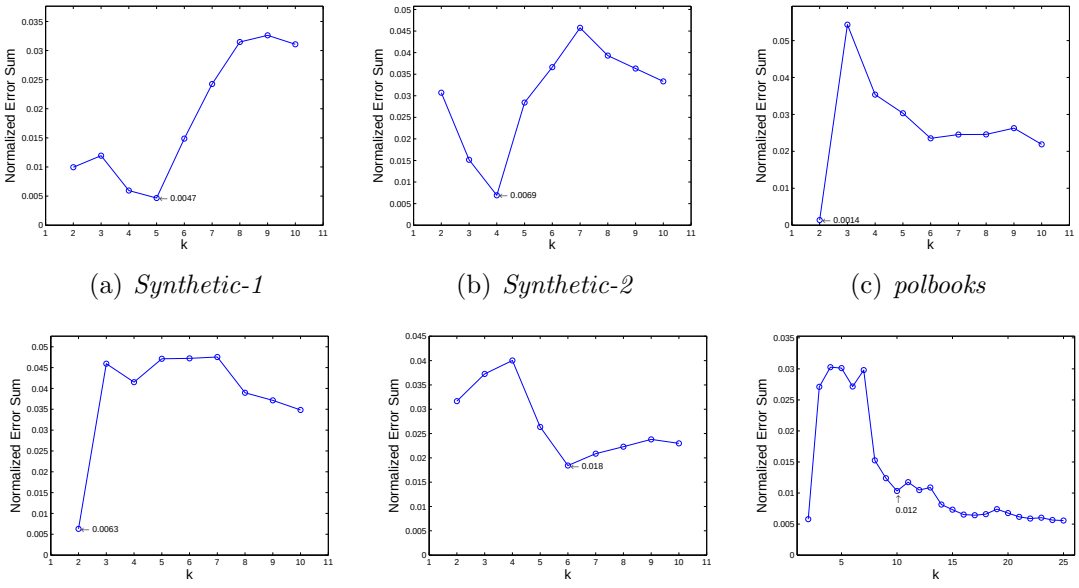
(a) *Synthetic-1*:$\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ (b) *Synthetic-1*:$\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ (c) *Synthetic-1*:$\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$

(d) *Syntetic-2*:$\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$ (e) *polblogs* $\boldsymbol{x}_1, \boldsymbol{x}_2$ (f) *polblogs* $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$

Figure 6.12: The plots of spectral coordinates for various networks

are actually four communities in *Synthetic-2*. We also show the 2-D and 3-D spectral plots of *polblogs*. As we know, there are two communities in this data. Hence, we can observe that spectral coordinates form two orthogonal lines in the subspace spanned by $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as shown in Figure 6.12(e). However, we cannot observe any clear line orthogonality pattern when we introduce the additional eigenvector $\boldsymbol{x}_3$, as shown in Figure 6.12(f).

Quality of Community Partition. Our strategy of determining k is to choose the one with the small normalized error ρ. Figure 6.13 shows how ρ changes when we vary k for various networks. We also circle out the k value we choose for each social network in Figure 6.13. For *Synthetic-1*, we can observe from Figure 6.13(a) that ρ reaches the minimum (0.0047) when $k = 5$ whereas for *Synthetic-2*, ρ reaches the minimum (0.0069) when $k = 4$ for *Synthetic-2* from Figure 6.13(a). This phenomenon matches the community numbers we used to generate data. For *polbooks*, *polblogs*, and *Enron*, we choose k that incurs the minimum ρ value. For *Facebook*, we can see there are several good choices of k from Figure 6.13(f). Although $k = 2$ incurs the



(a) *Synthetic-1* (b) *Synthetic-2* (c) *polbooks*

(d) *polblogs* (e) *Enron* (f) *Facebook*

Figure 6.13: Normalized error ρ vs. varied k

minimum ρ value, we instead choose $k = 10$ in our evaluation, which incurs small ρ and matches our manual examination.

Since the original data descriptions of *polbooks* and *polblogs* (and Synthetic) provide node-community relations, we evaluate the accuracy of our partition algorithm using $\frac{\sum_{i=1}^{k} |C_i \cap \hat{C}_i|}{n}$ where $\hat{C}_i$ denotes the i-th community produced by our algorithm. The last column of Table 6.6 shows our results. We can see our algorithm achieves high accuracy values. We also calculate the modularity Q of our partition algorithm on all networks.

Table 6.6: Statistics of networks and partition quality

	n	m	k	Q	Accuracy(%)
Synth-1	840	4917	5	0.38	90.83
Synth-2	840	5743	4	0.39	89.17
polbooks	105	441	2	0.45	96.7
polblogs	1222	16714	2	0.42	94.7
Enron	148	869	6	0.48	/
Facebook	63392	816886	10	0.52	/

Comparison with Normalized Cut. Researchers have developed several different

versions of normalized spectral clustering algorithms. In this paper, we use the normalized cut algorithm in [86] for comparison. The quality of the partitioning produced by our algorithm is comparable or better than that produced by the normalized cut for a wide range of graphs in terms of accuracy and modularity measures, as shown in Figure 6.14.

One well known problem of the normalized cut algorithm is that it tends to produce some small-sized partitions. Table 6.7 shows some statistics of the partition results on *Facebook* data. All the last four communities produced by the normalized cut contain less than 100 nodes, and have very few links to the rest of the graph. Take the community of size 10 as example, the community has 24 edges within itself and only 1 link connecting to other communities. These small communities do not make significant contributions to network topology although they are good candidates to minimize the number of cut edges. Clusters should be reasonably large groups of nodes. Our *AdjCut* algorithm produce more balanced partitions.

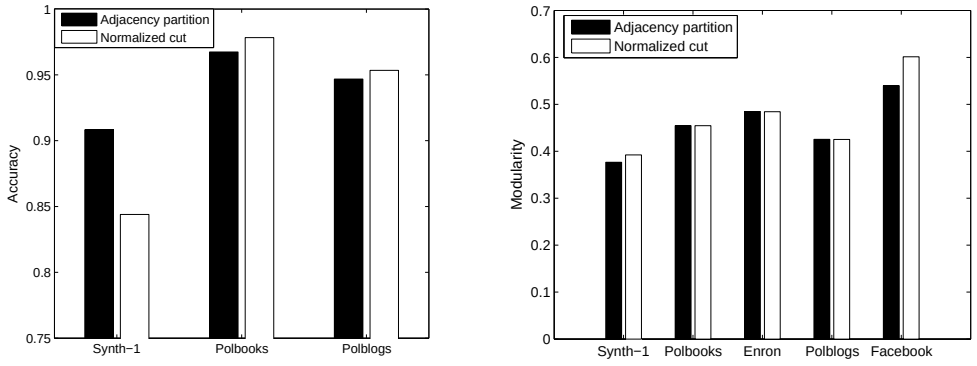
Table 6.7: Partition statistics on the *Facebook* network ($k = 10$)

Algorithm	Sizes of C_i	Q
AdjCut	29492, 7574, 7420, 6791, 3107, 2862, 1728, 1579, 1496, 1341	0.5389
Normalized cut	28709, 13325, 9236, 4548, 4394, 3092, 65, 10, 7, 6	0.6014

For the normalized cut and its variants, node u tends to be assigned to C_i if node u has more links to C_i than to other communities. Our *AdjCut* algorithm assigns u to its nearest line in the spectral space. Nodes are deviated from $\boldsymbol{r}_i$ due to its direct connections to other communities. Note that $\boldsymbol{r}_i$ is the closest line, and for any $j \neq i$ we have the following inequality:

$$x_{iu} = \sum_{\substack{v \in C_i \\ v \sim u}} \frac{x_{iv}}{\lambda_i} > \sum_{v \in \Gamma_u^j} \frac{x_{jv}}{\lambda_j}.$$

Note that x_{iv} or x_{jv} indicates the "belongings" of node v in community C_i or C_j. When the neighbors of node u in C_i have the largest total "belongings" (scaled by



(a) Accuracy (b) Modularity

Figure 6.14: Comparison of community partition results

λ_i), *AdjCut* assigns node u to community C_i even though node u may have more links to C_j than to C_i. The *AdjCut* may be more suitable for large and complex social networks, because it takes the association to the communities into consideration: a user is more likely to belong to the community when her friends in that community are core members than other community even with more connections to unimportant nodes.

6.6 Summary

In this chapter, we first discover the line orthogonality pattern in the spectral space of the adjacency matrix. Based on this pattern, we then present a framework which can quantify graph non-randomness at the edge, node, subgraph, and overall graph levels. We show that all graph non-randomness measures can be obtained mathematically from the spectrum of the adjacency matrix of the network. We also present a relative non-randomness measure of the overall graph, which allows quantitative comparisons between various social networks with different sizes and densities or between different snapshots of a dynamic social network.

We explore whether other graph spectra (such as Laplacian spectrum and normal spectrum) could also be used to derive a framework of non-randomness measures. Our theoretical results show that they are unlikely, if not impossible, to have a consistent

framework to evaluate randomness accurately at all granularity levels.

Utilizing the line orthogonality pattern in the adjacency spectral space, we develop a novel algorithm *AdjCut* to partition graph communities. We also discuss how to choose a proper k and conduct empirical comparison with the normal cut algorithm.

Wu et al. explore the spectral patterns of singed graphs [100]. In our future work, we would like to extend our framework of graph non-randomness to signed graphs, weighted graphs, directed graphs, and other types of rich graphs.

CHAPTER 7: SPECTRUM BASED NETWORK FRAUD DETECTION

In the preceding chapters, we mainly focus on the privacy-preserving graph randomization. Usually this type of randomization procedure is applied by the data owner. In this chapter, we consider another special type of randomization: the randomization caused by attackers. In large-scale and dynamic networks, each participant is vulnerable to various attacks including spam, denial of service, Sybil attacks, etc. The attackers can also join the social network and create links among themselves or to legitimate users, which constructs fraudulent nodes, links and subgraphs in the original social network. One type of such attack models is the *active attack*. Generally speaking, the subgraphs constructed by the attackers contain some patterns different from that of the original graph to some extent. One questions is that how we detect such subgraphs.

Specifically, we develop a fraud detection algorithm based on the non-randomness spectral framework to identify various attacks. Our approach, which exploits the spectral space of the underlying interaction structure of the network, is different from traditional topological analysis approaches [19, 77, 88]. Traditional topology based detection methods explore the graph topology directly and discover abnormal connectivity patterns caused by attacks. Our approach is based on graph spectral analysis that deals with the analysis of the spectra (eigenvalues and eigenvector components) of the adjacency matrix. We study how to identify attackers by characterizing their distributions in the spectral space.

Attacks in Social Networks. Social networks have always been vulnerable to various attacks including spam emails, annoying telemarketing calls, viral market-

ing, and individual re-identification in anonymized social network publishing. *Spam email* has been one of the most effective attacks. A majority of such spam emails are sent to victim email addresses that are generated randomly or chosen from existing mailing lists. Victims are unrelated generally and lack the variety of mutual personal, professional, institutional ties among individuals. Various techniques against spam [13, 14, 37, 42, 57, 83] have been developed. The key idea has been to build classification models using machine learning and data mining algorithms.

A *Viral Marketing* attack aims to leverage the power of social networks to produce rapid increase in brand awareness. Viral marketing is based on the fact that users are more receptive to a product or service recommended by their friends. In viral marketing, the marketer or attacker can create a set of seemingly innocent profiles, use them to make friendship links with a large set of seed users, then send advertisements to those seed users. It is expected that some seed users will recommend advertisements to their friends.

In an auction network[1], reputation systems have been used extensively by auction sites to prevent auction fraud [20, 77, 81]. However, it is difficult to truly assess the trustworthiness and show faithful representation of users' reputation. Many fraudsters can be detected by identifying relatively small and densely connected subgraphs since they usually interact in small *cliques* of their own (in order to mutually boost their credibility). The authors [19, 77] uncovered a different modus operandi for fraudsters in auction networks, which leads to the formation of near *bipartite cores*. Fraudsters make use of accomplices, who behave like honest users, except that they interact heavily with a small set of fraudsters in order to boost their reputation. The fraud identities are the ones used eventually to carry out the actual fraud, while the accomplices exist only to help the fraudsters carry out their job by boosting their feedback rating.

[1]Transactions among users are modeled as a graph, with a node for each user and an edge for one or more transactions between two users

Recently, the authors in [88] provided a general abstraction, called the *Random Link Attack* (RLA), which identifies the collaborative nature of these attacks to evade detection. In an RLA, the malicious user creates a set of false identities and uses them to connect with a large set of victim nodes. To evade detection, the malicious user also creates various interactions among false identities, which make the subgraph formed by false identities similar to that formed by regular users. This property makes the discovery of the attack and the responsible entities a difficult task.

The rest of this chapter is organized as follows. In Section 7.1, we first give our result on the change of eigenvectors during graph perturbation. In Section 7.2, we present a theoretical framework for detecting collaborative attacks based on spectral coordinates. In Section 7.3, we focus on RLAs. We derive distributions of spectral coordinates of attacking nodes and present our algorithm to filter attacking groups using their spectral characteristics. In Section 7.4, we conduct empirical evaluations and compare with topology based approaches. We extend RLAs to other attacking scenarios and offer our concluding remarks and discuss future work in Section 7.5. Some results in this chapter are also reported in [109, 110].

7.1 Graph Spectral Analysis

Our observed graph $\tilde{G}$ with adjacency matrix $\tilde{A}$ contains some fake links and nodes generated by the attackers. We use E to denote the difference matrix: $\tilde{A} = A + E$ (if new nodes are added, we simply extend A to the same dimension of $\tilde{A}$ by adding all-zero rows and columns). Let $\lambda_j(\tilde{\lambda}_j)$ denote the j-th largest eigenvalue of $A(\tilde{A})$ with eigenvector $\boldsymbol{x}_j(\tilde{\boldsymbol{x}}_j)$. We are interested in how the graph spectra (eigenvalue and eigenvector) are affected by perturbation.

The relationship between $\tilde{\boldsymbol{x}}_j$ and $\boldsymbol{x}_j$ has also been well studied. In [90] (refer to Theorem 2.7 and 2.8), it was shown that $\tilde{\boldsymbol{x}}_j$ can be approximated by a function of all original eigenvectors and the perturbation matrix E. However, it is difficult to apply them to separate fraud nodes from regular ones in the perturbed spectral space. In

this paper, we apply the Power Iteration method [38] to derive the following result.

Result 7.1: Suppose eigenvalue λ_j satisfies $|\lambda_j| \gg \|E\|_2$. For small integer t, the eigenvector of λ_j, $\tilde{\boldsymbol{x}}_j$, can be approximated by:

$$\tilde{\boldsymbol{x}}_j \approx \boldsymbol{x}_j + \sum_{l=1}^{t} \frac{1}{\lambda_j^l}(A+E)^{l-1}E\boldsymbol{x}_j. \tag{7.1}$$

Proof. To derive the relationship between $\tilde{\boldsymbol{x}}_j$ and $\boldsymbol{x}_j$, we first introduce the Power Iteration method [38].

Let A be a $n \times n$ symmetric matrix, and $\boldsymbol{e}^{(0)}$ be an nonzero $n \times 1$ vector, assuming $\boldsymbol{x}_1^T\boldsymbol{e}^{(0)} \neq 0$. Then, series $\boldsymbol{e}^{(s+1)} = \frac{A\boldsymbol{e}^{(s)}}{\|A\boldsymbol{e}^{(s)}\|}$ converges to the eigenvector corresponding to the largest eigenvector of A. To compute $\boldsymbol{x}_i$, starting with $\boldsymbol{e}^{(0)}$ that satisfies $\boldsymbol{x}_i^T\boldsymbol{e}^{(0)} \neq 0$, series $\boldsymbol{e}^{(s+1)} = \frac{A\boldsymbol{e}^{(s)}}{\|A\boldsymbol{e}^{(s)}\|}\perp\boldsymbol{x}_j$, $j = 1, \ldots, i-1$, converges to $\boldsymbol{x}_i$, where $\boldsymbol{v}\perp\boldsymbol{x}_j$ means orthogonalizing vector $\boldsymbol{v}$ with previous eigenvectors.

Consider computing $\tilde{\boldsymbol{x}}_1$ and $\tilde{\lambda}_1$ using the power iteration method. Since the noise is moderate, the original eigenvector $\boldsymbol{x}_1$ is a good initial vector, and hence $\frac{\tilde{A}^t\boldsymbol{x}_1}{\|\tilde{A}^t\boldsymbol{x}_1\|_2}$ is a good approximation of $\tilde{\boldsymbol{x}}_1$. To make the later proofs concise, we normalize $\tilde{A}^t\boldsymbol{x}_1$ by λ_1^t instead of it exact vector length $\|\tilde{A}^t\boldsymbol{x}_1\|_2$. Let $\tilde{\boldsymbol{x}}_1^{(t)} = \tilde{A}^t\boldsymbol{x}_1/\lambda_1^t$, then we have

$$\tilde{\boldsymbol{x}}_1^{(t)} = \frac{(A+E)^t\boldsymbol{x}_1}{\lambda_1^t} = \boldsymbol{x}_1 + \sum_{l=1}^{t} \frac{1}{\lambda_1^l}(A+E)^{l-1}E\boldsymbol{x}_1. \tag{7.2}$$

When t is not too large, $\|\tilde{\boldsymbol{x}}_1^{(t)}\|_2$ is close to 1. This is because E is formed by adding edges, and hence $\lambda_1^t \leq \|\tilde{A}^t\boldsymbol{x}_1\|_2 \leq \tilde{\lambda}_1^t \leq (\lambda_1 + \|E\|_2)^t$. When $\lambda_1 \gg \|E\|_2$ and t is not too large, we have $1/\|\tilde{A}^t\boldsymbol{x}_1\|_2 \approx 1/\lambda_1^t$. Altogether, we have

$$\tilde{\boldsymbol{x}}_1 \approx \frac{\tilde{A}^t\boldsymbol{x}_1}{\|\tilde{A}^t\boldsymbol{x}_1\|_2} \approx \frac{\tilde{A}^t\boldsymbol{x}_1}{\lambda_1^t} = \tilde{\boldsymbol{x}}_1^{(t)}. \tag{7.3}$$

Note that vector $\tilde{\boldsymbol{x}}_1^{(t)}$ and $\frac{\tilde{A}^t\boldsymbol{x}_1}{\|\tilde{A}^t\boldsymbol{x}_1\|_2}$ have the same direction but slightly different vector lengths, and $\tilde{\boldsymbol{x}}_1^{(t)}$ thus converges to $\tilde{\boldsymbol{x}}_1$ in direction. Let θ_t denote the angle between $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_1^{(t)}$. With the power iteration, we have $\cos\theta_t = 1 - O(|\frac{\tilde{\lambda}_2}{\tilde{\lambda}_1}|^t)$, i.e., $\cos\theta_t$ approaches to 1 geometrically with ratio $|\frac{\tilde{\lambda}_2}{\tilde{\lambda}_1}|$ [38].

Similarly, we can show (7.1) stands for $\tilde{\boldsymbol{x}}_j$ $(j \neq 1)$ when $\lambda_j \gg \|E\|_2$. When it comes to $\tilde{\boldsymbol{x}}_j$ $(j \neq 1)$, we need to ensure the orthogonality of the eigenvectors. The orthogonality approximately stands since $\|E\boldsymbol{x}_j\|_2 \leq \|E\|_2 \ll |\lambda_j|$.

Remarks: The approximation in (7.3) stands when t is small, in which case $\|\tilde{A}^t\boldsymbol{x}_1\|_2 \approx \lambda_1^t$. How to determine t and what is the exact form of the error term are beyond the scope of this work. We would like to give some illustration here. Eigenvectors $\tilde{\boldsymbol{x}}_i$ $(i = 1, \ldots, n)$ form a basis in $\mathbb{R}^n$. Let $\boldsymbol{x}_1 = \sum_{i=1}^n c_i\tilde{\boldsymbol{x}}_i$, where c_i is the coefficient, $c_i = \boldsymbol{x}_1^T\tilde{\boldsymbol{x}}_i$. When the noise is moderate, $\tilde{\boldsymbol{x}}_1$ and $\boldsymbol{x}_1$ are close in direction, i.e., c_1 is close to 1 while c_i $(i \neq 1)$ is close to 0 (note that $\sum_i c_i^2 = 1$). Then $\tilde{\boldsymbol{x}}_1^{(t)} = \frac{\tilde{A}^t\boldsymbol{x}_1}{\lambda_1^k} = \sum_{i=1}^n c_i(\frac{\tilde{\lambda}_i}{\lambda_1})^t\tilde{\boldsymbol{x}}_i$. Hence the error term is

$$\begin{aligned}
\|\tilde{\boldsymbol{x}}_1^{(t)} - \tilde{\boldsymbol{x}}_1\|_2 &= \left\| [c_1(\frac{\tilde{\lambda}_1}{\lambda_1})^t - 1]\tilde{\boldsymbol{x}}_1 + \sum_{i=2}^n c_i(\frac{\tilde{\lambda}_i}{\lambda_1})^t\tilde{\boldsymbol{x}}_i \right\|_2 \\
&= \left([c_1(\frac{\tilde{\lambda}_1}{\lambda_1})^t - 1]^2 + \sum_{i=2}^n c_i^2(\frac{\tilde{\lambda}_i}{\lambda_1})^{2t} \right)^{\frac{1}{2}} \\
&\approx |1 - c_1(\frac{\tilde{\lambda}_1}{\lambda_1})^t|. \qquad (c_i \approx 0, \text{ for } i \neq 1)
\end{aligned}$$

Similarly, we can have

$$\|\boldsymbol{x}_1 - \tilde{\boldsymbol{x}}_1\|_2 = \left[(c_1 - 1)^2 + \sum_{i=2}^n c_i^2 \right]^{\frac{1}{2}} \approx |1 - c_1|.$$

Note that c_1 is a constant less than 1, and $\frac{\tilde{\lambda}_1}{\lambda_1} > 1$. When t is small, $c_1(\frac{\tilde{\lambda}_1}{\lambda_1})^t$ is closer to 1 than c_1, and hence $\|\tilde{\boldsymbol{x}}_1^{(t)} - \tilde{\boldsymbol{x}}_1\|_2 \leq \|\boldsymbol{x}_1 - \tilde{\boldsymbol{x}}_1\|_2$. As t increases, $c_1(\frac{\tilde{\lambda}_1}{\lambda_1})^t$ goes to infinity, and hence the error increases. This is because, when t is large, $\|\tilde{A}^t\boldsymbol{x}_1\|_2$ can be much greater than λ_1^t, and more accurate normalization in vector length is desired. In our work, we only use $t = 1, 2$. We find the above approximations are very stable in most practical cases. □

We can see that in our approximation $\tilde{\boldsymbol{x}}_j$ is expressed as a function of only $\boldsymbol{x}_j$ and E for those leading eigenvectors ($|\lambda_j| \gg \|E\|_2$). The approximation in (7.3) stands

when t is small. How to determine the optimal t is beyond the scope of this work. Please refer to our proof and remarks in Appendix 1 for the discussion of the error term. In this paper, we use $t = 1, 2$ and find the above approximations are very stable in most practical cases.

7.2 A Spectrum Based Framework for Detecting Attacks

In this section, we present a spectrum based fraud detection framework, which is different from the traditional topology based fraud detection. In our framework, we exploit the spectral space and characterize the difference between the spectral coordinates of regular users and that of attackers, rather than exploring the graph topology directly.

In a collaborative attack, the malicious user has complete control over the attacking nodes and uses them to attack (e.g., send emails) a large set of victim nodes. Assume there are c ($c \ll n$) attacking nodes and they form a subgraph with adjacency matrix $C = \{c_{ij}\}_{c\times c}$. The outgoing links from attacking nodes to regular nodes form the subgraph with adjacency matrix $B = (b_{ij})_{n\times c}$: $b_{ij} = 1$ if the j-th attacking node has a link to the i-th regular node, and $b_{ij} = 0$ otherwise. The graph after attacks $\tilde{G}$ has $N = n + c$ nodes, and we can arrange the nodes in the graph so that node 1 to c are attacking nodes and node $c + 1$ to N are regular ones. We have:

$$A = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_n \end{pmatrix}, \tilde{A} = \begin{pmatrix} C & B^T \\ B & A_n \end{pmatrix}, E = \begin{pmatrix} C & B^T \\ B & \mathbf{0} \end{pmatrix} \tag{7.4}$$

The degree of node i, d_i, is the number of links connecting to node i in $\tilde{G}$, including the attacking links: $d_i = \sum_{j=1}^{N} \tilde{a}_{ij}$.

Let $\boldsymbol{z}_j$ be the eigenvector of A associated to λ_j, and $\tilde{\boldsymbol{z}}_j$ be the eigenvector of $\tilde{A}$ associated to eigenvalue $\tilde{\lambda}_j$. Then, $\boldsymbol{z}_j$ and $\tilde{\boldsymbol{z}}_j$ can be partitioned as follows:

$$\boldsymbol{z}_j = \begin{pmatrix} \mathbf{0}_{c\times 1} \\ \boldsymbol{x}_j \end{pmatrix}, \quad \tilde{\boldsymbol{z}}_j = \begin{pmatrix} \tilde{\boldsymbol{y}}_j \\ \tilde{\boldsymbol{x}}_j \end{pmatrix},$$

where $\tilde{\boldsymbol{y}}_j = (\tilde{y}_{j1}, \ldots, \tilde{y}_{jc})^T$ denotes the entries corresponding to the attackers in $\tilde{\boldsymbol{z}}_j$ and $\tilde{\boldsymbol{x}}_j = (\tilde{x}_{j1}, \ldots, \tilde{x}_{jn})^T$ denotes the entries corresponding to those regular nodes in $\tilde{\boldsymbol{z}}_j$. Since A is expanded by adding 0's into A_n, $\boldsymbol{x}_j$ is then the eigenvector of A_n along with the eigenvalue λ_j. Let $\bar{x}_j$ be the mean value of entries in $\boldsymbol{x}_j$: $\bar{x}_j = \frac{1}{n}\mathbf{1}_n^T \boldsymbol{x}_j$. To make the deduction simple, we choose the sign of $\boldsymbol{x}_j$ so that $\bar{x}_j \geq 0$.

We utilize the leading k eigenvalues and eigenvectors to detect attacks in network data. In the following sections, we denote $\boldsymbol{\alpha}_u = (x_{1u}, x_{2u}, \cdots, x_{ku})$ as the spectral coordinate of regular node u in the original spectral space. It is the u-th row in the bottom part of the matrix shown in (7.5). Denote $\boldsymbol{\beta}_i = (y_{1i}, y_{2i}, \cdots, y_{ki})$ as the spectral coordinate of attacking node i (the i-th row in the upper part of the matrix in (7.5)). Since we assume there is no attack in the original graph, $\boldsymbol{\beta}_i$ is actually a zero vector. Similarly, we denote $\tilde{\boldsymbol{\alpha}}_u = (\tilde{x}_{1u}, \tilde{x}_{2u}, \cdots, \tilde{x}_{ku})$ and $\tilde{\boldsymbol{\beta}}_i = (\tilde{y}_{1i}, \tilde{y}_{2i}, \cdots, \tilde{y}_{ki})$ as the spectral coordinate of regular node u and attacking node i in the perturbed spectral space respectively, as shown in (7.6).

$$\left(\begin{array}{ccc} \mathbf{0} & \cdots & \mathbf{0} \\ \hline \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_k \end{array}\right) = \left(\begin{array}{ccccc} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \\ \hline x_{11} & \cdots & x_{j1} & \cdots & x_{k1} \\ \vdots & & \vdots & & \vdots \\ x_{1n} & \cdots & x_{jn} & \cdots & x_{kn} \end{array}\right) \begin{array}{l} \\ \\ \leftarrow \boldsymbol{\beta}_i \\ \\ \\ \leftarrow \boldsymbol{\alpha}_u \\ \\ \end{array} \tag{7.5}$$

$$\left(\begin{array}{ccc} \tilde{\boldsymbol{y}}_1 & \cdots & \tilde{\boldsymbol{y}}_k \\ \hline \tilde{\boldsymbol{x}}_1 & \cdots & \tilde{\boldsymbol{x}}_k \end{array}\right) = \left(\begin{array}{ccccc} \tilde{y}_{11} & \cdots & \tilde{y}_{j1} & \cdots & \tilde{y}_{k1} \\ \vdots & & \vdots & & \vdots \\ \tilde{y}_{1c} & \cdots & \tilde{y}_{jc} & \cdots & \tilde{y}_{kc} \\ \hline \tilde{x}_{11} & \cdots & \tilde{x}_{j1} & \cdots & \tilde{x}_{k1} \\ \vdots & & \vdots & & \vdots \\ \tilde{x}_{1n} & \cdots & \tilde{x}_{jn} & \cdots & \tilde{x}_{kn} \end{array}\right) \begin{array}{l} \\ \\ \leftarrow \tilde{\boldsymbol{\beta}}_i \\ \\ \\ \leftarrow \tilde{\boldsymbol{\alpha}}_u \\ \\ \end{array} \tag{7.6}$$

Result 7.2: In a graph $\tilde{G}$ under collaborative attacks, for attacking node i, the eigenvector entry $\tilde{y}_{ji}$ $(1 \leq i \leq c, 1 \leq j \leq k)$ can be approximated by:

$$\tilde{y}_{ji} \approx \frac{1}{\lambda_j} \sum_{u \in \Omega_i} x_{ju} + \frac{1}{\lambda_j^2} \sum_{r=1}^{c} \left(c_{ir} \sum_{u \in \Omega_r} x_{ju} \right), \tag{7.7}$$

where Ω_r denotes the victim set of attacking node r. For any regular node u, $1 \leq u \leq n$, $\tilde{x}_{ju}$ is approximately unchanged: $\tilde{x}_{ju} \approx x_{ju}$.

Proof. Substituting the corresponding matrices as shown in (7.4) to Result 7.1, we can approximate the eigenvectors after attacks $\tilde{\boldsymbol{z}}_j$ by $\boldsymbol{z}_j$ and λ_j:

$$\begin{pmatrix} \tilde{\boldsymbol{y}}_j \\ \tilde{\boldsymbol{x}}_j \end{pmatrix} \approx \begin{pmatrix} \mathbf{0} \\ \boldsymbol{x}_j \end{pmatrix} + \frac{1}{\lambda_j} \begin{pmatrix} B^T \boldsymbol{x}_j \\ \mathbf{0} \end{pmatrix} + \frac{1}{\lambda_j^2} \begin{pmatrix} CB^T \boldsymbol{x}_j \\ BB^T \boldsymbol{x}_j \end{pmatrix}.$$

With the above expression, we can write the i-th entry in $\tilde{\boldsymbol{y}}_j$ as

$$\begin{aligned} \tilde{y}_{ji} &\approx \frac{1}{\lambda_j} \boldsymbol{b}_i^T \boldsymbol{x}_j + \frac{1}{\lambda_j^2} \sum_{r=1}^{c} c_{ir} \boldsymbol{b}_r^T \boldsymbol{x}_j \\ &= \frac{1}{\lambda_j} \sum_{u=1}^{n} b_{ui} x_{ju} + \frac{1}{\lambda_j^2} \sum_{r=1}^{c} \left(c_{ir} \sum_{u=1}^{n} b_{ur} x_{ju} \right). \end{aligned} \tag{7.8}$$

Note that $\boldsymbol{b}_r$ is the index vector for the victims attacked by node r, then $\sum_{u=1}^{n} b_{ur} x_{ju} = \sum_{u \in \Omega_r} x_{ju}$, and we get (7.7).

For the the regular nodes, it is unlikely that a regular node u is attacked by many RLA attackers, and hence the terms of λ_j^{-2} can be further neglected, and we have $\tilde{x}_{ju} \approx x_{ju}$. We prove the result. □

Result 7.2 shows that the spectral coordinate of an attacking node can be approximated by the spectral coordinates of its victims. Figure 7.1 shows a collaborative attack example. Attacking nodes (black ones in the dashed region) form an inner subgraph and each attacking node links to some victims (gray ones). For example, the victim set of attacking node p, Ω_p, includes victims u, v, w. For example, the spectral coordinate of attacking node p is mainly determined by the sum of spectral

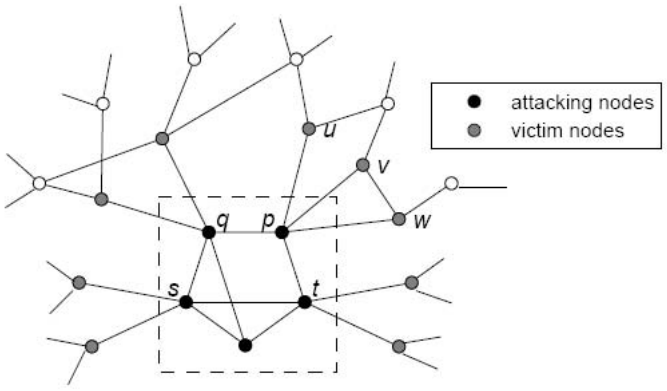


Figure 7.1: A collaborative attack example

coordinates of its victim set Ω_p scaled by $1/\lambda_j$ (the first term of the right hand side of (7.7)). The second term captures the effect of all its neighbor attacking nodes' victims, Ω_q and Ω_t (scaled by $1/\lambda_j^2$).

When attackers do not collaborate with each other ($C = \mathbf{0}_{c\times c}$), the second term of the right hand side of (7.7) disappears. We simply have $\tilde{y}_{ji} \approx \frac{1}{\lambda_j}\sum_{u\in\Omega_i} x_{ju}$, which indicates the attacker's spectral coordinate is fully determined by that of its victims. From (7.7) we can also observe that the inner structure C among the attackers only affects $\tilde{y}_{ji}$ in the order of λ_j^{-2}. When λ_j is large, the second term of the right hand side of (7.7) is already negligible, which means that the inner subgraph structure has little impact on the distribution of attackers in the spectral space.

The above result is mathematically elegant. We show that the spectral coordinate of an attacker is mainly determined by that of its victims and the inner structure among collaborative attackers has negligible impact on attackers' spectral coordinate distributions in the spectral space. Hence the efforts by the collaborative attackers of resembling the rest of the network do not help much in hiding their spectral characteristics in the spectral space.

In practice users have no knowledge about which nodes are attackers (or victims). In other words, users do not know the true spectral coordinates of victim nodes (i.e., x_{ji}) as well as the eigenvalues of the original graph (i.e., λ_j). As a result, we

cannot derive the exact spectral coordinates of attacking nodes (i.e., y_{ji}). In the next section, we will present our results on the random link attack. We will show that the distribution of attackers' spectral coordinates under random link attack are determined by $\tilde{\boldsymbol{z}}_j$ and $\tilde{\lambda}_j$, which can be calculated directly from the observed graph $\tilde{A}$.

7.3 Detecting Random Link Attack

A Random Link Attack (RLA) is a special type of collaborative attacks. The malicious user has complete control over the attacking nodes and uses them to attack (e.g., send emails) a large randomly chosen set of victim nodes. To masquerade as regular users, the attackers usually form a dense subgraph by increasing the number of edges among themselves such that the attacker's neighborhood is structurally similar to that of a regular user. One assumption here is that the attacker's victim set is selected randomly. In other words, each regular node in the graph has an equal probability to be attacked, independent of other victims. Also note that for a successful attack, the size of the victim set is typically large as compared to the size of the attacker set. If this is not true, then the scope of the RLA attack is severely constrained. The nodes in the neighborhood of a regular user typically contains a set of communities, or a group of nodes that also have edges between themselves. On the contrary, the randomly chosen victim nodes in the neighborhood of the attackers have a different structure with fewer edges between themselves.

Definition 7.1: Random Link Attack (RLA) In a RLA, the malicious user creates $c(\ll n)$ false identities (attacking nodes) and uses them to connect with a large set of victims. Attacking node i randomly attack v_i victims and each regular node has the same probability to be a victim. The total number of victims is $v = \sum_{i=1}^{c} v_i$. To evade detection, the malicious user also creates m_c links among attacking nodes, which may make the subgraph formed by attacking nodes similar to that formed by regular users.

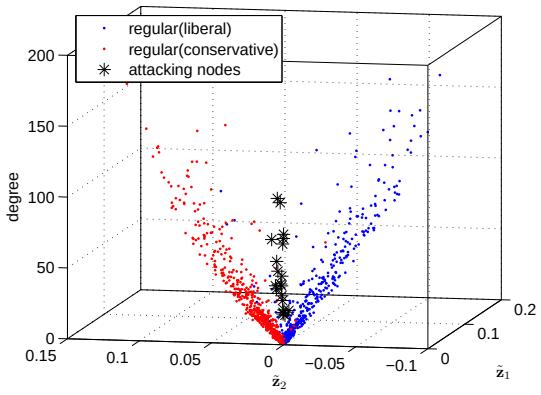


Figure 7.2: Spectral coordinates of political blogsphere data under a degree attack with 20 attackers.

Throughout this section, we use *polblogs* as an example to illustrate our theoretical results. Figure 7.2 plots the node spectral coordinates under a degree attack[2] with 20 attackers. We also show node degrees in the z-axis. We can observe from the figure that the majority of nodes projected in the 2-D spectral space distribute along two straight and quasi-orthogonal lines. This indicates that there exist two communities with sparse edges connecting them. We also observe that attacking nodes (denoted as black) locate between the two quasi-orthogonal lines in the spectral projection space.

7.3.1 Identifying Suspects in Spectral Space

In this section, we investigate how attackers distribute in the spectral space. By identifying the distribution of attackers' spectral coordinates, we expect to separate attacking nodes from regular ones in the spectral space.

Result 7.3: Let $\Lambda_k = \text{diag}(\lambda_1, \ldots, \lambda_k)$ and $X_k = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$. When all attackers satisfy $v_i \leq \lambda_k$ and $d_i \leq \frac{n}{2}$, the spectral coordinate $\tilde{\boldsymbol{\beta}}_i(\tilde{y}_{1i}, \tilde{y}_{2i}, \ldots, \tilde{y}_{ki})$ asymptotically follows the multivariate normal distribution whose mean and covariance satisfy the

[2]Attacking nodes have the same degree distribution as the regular nodes. For attacking node i, it attacks $\frac{2d_i}{3}$ victims.

following two inequalities[3]:

$$\mathbf{E}(\tilde{\boldsymbol{\beta}}_i) \leq d_i \bar{X}_k \Lambda_k^{-1} = d_i \left(\frac{\bar{x}_1}{\lambda_1}, \ldots, \frac{\bar{x}_k}{\lambda_k} \right), \tag{7.9}$$

$$\mathrm{Cov}(\tilde{\boldsymbol{\beta}}_i) \leq \frac{d_i}{n} \left(1 - \frac{d_i}{n} \right) \Lambda_k^{-2}. \tag{7.10}$$

Furthermore, $\tilde{y}_{si}$ and $\tilde{y}_{ti}$ $(s \neq t)$ are independent. When attackers do not collaborate with each other ($C = \mathbf{0}_{c \times c}$), the expectation and variance reach their upper bounds.

Proof. Since each regular node has the same probability to be attacked, and entries of $\boldsymbol{b}_i$ are i.i.d. bernoulli random variables with parameter $p_i = \frac{v_i}{n}$. We thus have

$$\mathbf{E}(\boldsymbol{b_i}) = \frac{v_i}{n} \mathbf{1}_{n \times 1}, \ \mathrm{Cov}(\boldsymbol{b_i}) = \frac{v_i}{n}(1 - \frac{v_i}{n}) I_{n \times n}.$$

With (7.8), $\tilde{y}_{ji}$ is a linear function of $\boldsymbol{b}_i$ and $\boldsymbol{b}_r$. Since entries of $\boldsymbol{b}_i$ (or $\boldsymbol{b}_r$) are i.i.d. bernoulli random variables, when n is large, $\tilde{y}_{ji}$ asymptotically follows the multivariate normal distribution. Taking the expectation of (7.8), we have

$$\begin{aligned} \mathbf{E}(\tilde{y}_{ji}) &= \frac{1}{\lambda_j} \mathbf{E}(\boldsymbol{b}_i)^T \boldsymbol{x}_j + \frac{1}{\lambda_j^2} \sum_{r=1}^{c} c_{ir} \, \mathbf{E}(\boldsymbol{b}_r)^T \boldsymbol{x}_j \\ &= \frac{v_i \bar{x}_j}{\lambda_j} + \frac{1}{\lambda_j^2} \sum_{r=1}^{c} c_{ir} v_r \bar{x}_j \end{aligned} \tag{7.11}$$

With $v_r \leq \lambda_j$, we have

$$\begin{aligned} \mathbf{E}(\tilde{y}_{ji}) &\leq \frac{v_i \bar{x}_j}{\lambda_j} + \frac{1}{\lambda_j} \sum_{r=1}^{c} c_{ir} \bar{x}_j \\ &= \frac{\bar{x}_j}{\lambda_j} \left(v_i + \sum_{r=1}^{c} c_{ir} \right) = \frac{d_i \bar{x}_j}{\lambda_j}. \end{aligned}$$

We have proved the upper bound for the expectation shown in (7.9). Note that when $C = \mathbf{0}$, $\mathbf{E}(\tilde{y}_{ji})$ is reduced to

$$\mathbf{E}(\tilde{y}_{ji}) = \frac{1}{\lambda_j} \mathbf{E}(\boldsymbol{b}_i)^T \boldsymbol{x}_j = \frac{v_i \bar{x}_j}{\lambda_j} = \frac{d_i \bar{x}_j}{\lambda_j}.$$

[3]By using "$\leq$" between two vectors or matrices, we mean entry-wise less or equal.

The last equality holds because there is no link among attackers($v_i = d_i$). In this case, the expectation reaches the upper bound.

With (7.8), we can write $\tilde{\boldsymbol{\beta}}_i = (\tilde{y}_{1i}, \ldots, \tilde{y}_{ki})$ in matrix form:

$$\tilde{\boldsymbol{\beta}}_i = \boldsymbol{b}_i^T X_k \Lambda_k^{-1} + \sum_{r=1}^{c} c_{ir} \boldsymbol{b}_r^T X_k \Lambda_k^{-2}.$$

When the graph is large, we simply regard that $\boldsymbol{b}_i$ and $\boldsymbol{b}_r$ are independent. Also notice that $c_{ii} \equiv 1$, and we have

$$\begin{aligned}
&\operatorname{Cov}(\tilde{\boldsymbol{\beta}}_i) = \Lambda_k^{-1} X_k^T \operatorname{Cov}(\boldsymbol{b}_i) X_k \Lambda_k^{-1} \\
&+ \sum_{r=1}^{c} c_{ir} \Lambda_k^{-2} X_k^T \operatorname{Cov}(\boldsymbol{b}_r) X_k \Lambda_k^{-2} \\
&= \frac{v_i}{n}(1 - \frac{v_i}{n}) \Lambda_k^{-2} + \Lambda_k^{-4} \sum_{r=1}^{c} c_{ir} \frac{v_r}{n}(1 - \frac{v_r}{n}).
\end{aligned} \tag{7.12}$$

When $v_i \le d_i \le \frac{n}{2}$, $\frac{v_i}{n}(1 - \frac{v_i}{n}) < \frac{d_i}{n}(1 - \frac{d_i}{n})$. We can thus enlarge the first term of (7.12) to $\frac{d_i}{n}(1 - \frac{d_i}{n})\Lambda_k^{-2}$, and the term of Λ_k^{-4} is then negligible. In summary, we have

$$\operatorname{Cov}(\tilde{\boldsymbol{\beta}}_i) \le \frac{d_i}{n}\left(1 - \frac{d_i}{n}\right) \Lambda_k^{-2}.$$

Notice that the covariance matrix in (7.12) is an diagonal matrix. For multivariate normal distribution, we know that two entries $\tilde{y}_{si}$ and $\tilde{y}_{ti}$ $(s \neq t)$ are then independent. When $C = \mathbf{0}$, the second term of (7.12) is 0, and $\operatorname{Cov}(\tilde{\boldsymbol{\beta}}_i)$ then reaches the upper bound. □

Specifically, $\tilde{y}_{ji}$ follows the normal distribution whose mean and variance satisfy the following two inequalities:

$$\mathbf{E}(\tilde{y}_{ji}) \le \frac{d_i \bar{x}_j}{\lambda_j}, \quad \mathbf{V}(\tilde{y}_{ji}) \le \frac{d_i}{n}\left(1 - \frac{d_i}{n}\right) \frac{1}{\lambda_j^2}. \tag{7.13}$$

When attackers do not collaborate with each other, we know the exact values of

expectation and variance of $\tilde{y}_{ji}$.

$$\mathbf{E}(\tilde{y}_{ji}) = \frac{d_i \bar{x}_j}{\lambda_j}, \quad \mathbf{V}(\tilde{y}_{ji}) = \frac{d_i}{n}\left(1 - \frac{d_i}{n}\right)\frac{1}{\lambda_j^2}. \tag{7.14}$$

Lemma 7.1: In the setting of RLA, let $\bar{\tilde{z}}_j$ denote the mean of $\tilde{\boldsymbol{z}}_j$: $\bar{\tilde{z}}_j = \frac{1}{n+c}\mathbf{1}_{n+c}^T \tilde{\boldsymbol{z}}_j$. When $\lambda_j \gg \|E\|_2$, we have $\bar{\tilde{z}}_j \to \bar{x}_j$, as $n \to \infty$.

Proof. When G is attacked by RLA (either with or without collaboration), $\tilde{\boldsymbol{z}}_j$ can be approximated to the first order as $\tilde{\boldsymbol{z}}_j \approx \begin{pmatrix} \frac{1}{\lambda_j} B^T \boldsymbol{x}_j \\ \boldsymbol{x}_j \end{pmatrix}$. Then, we have

$$\begin{aligned}
\bar{\tilde{z}}_j &= \frac{1}{n+c}\mathbf{1}_{n+c}^T \tilde{\boldsymbol{z}}_j = \frac{1}{n+c}(\mathbf{1}_c^T, \mathbf{1}_n^T)\begin{pmatrix} \frac{1}{\lambda_j} B^T \boldsymbol{x}_j \\ \boldsymbol{x}_j \end{pmatrix} \\
&= \frac{1}{n+c}\left(\frac{1}{\lambda_j}\mathbf{1}_c^T B^T \boldsymbol{x}_j + n\bar{x}_j\right) \\
&\to \frac{1}{n+c}\left(\frac{1}{\lambda_j}\mathbf{1}_c^T\, \mathbf{E}(B^T \boldsymbol{x}_j) + n\bar{x}_j\right), \text{ as } n \to \infty.
\end{aligned} \tag{7.15}$$

Note that $\mathbf{E}(B^T \boldsymbol{x}_j) = \mathrm{diag}(v_1 \bar{x}_j, v_2 \bar{x}_j, \ldots, v_c \bar{x}_j)$, we have

$$\frac{1}{\lambda_j}\mathbf{1}_c^T\, \mathbf{E}(B^T \boldsymbol{x}_j) = \frac{\bar{x}_j}{\lambda_j}\sum_{i=1}^{c} v_i = \frac{v\bar{x}_j}{\lambda_j}.$$

Continue with (7.15), and we have

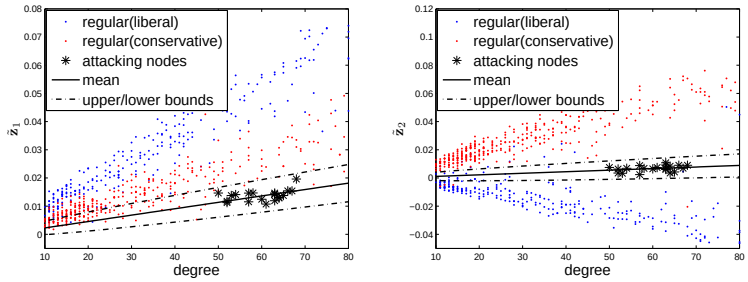
$$\bar{\tilde{z}}_j \to \frac{v}{(n+c)\lambda_j}\bar{x}_j + \frac{n}{n+c}\bar{x}_j. \tag{7.16}$$

Since $v \le n$ and $\lambda_j \gg 1$, the first term in (7.16) is negligible. The second term in (7.16) approaches 1 since $c \ll n$, and we get the result $\bar{\tilde{z}}_j \to \bar{x}_j$ as $n \to \infty$. □

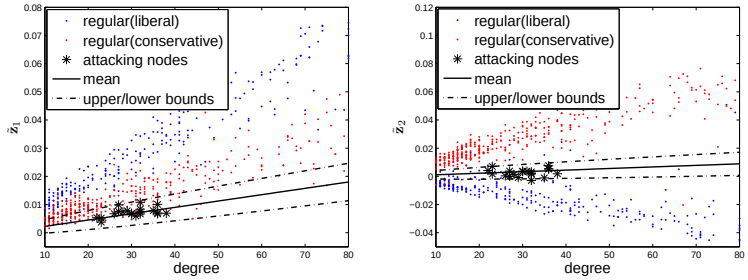
One problem here is that users do not know the values of $\bar{x}_j$. Lemma 7.1 shows that in the setting of RLA, $\bar{x}_j$ can be approximated by $\bar{\tilde{z}}_j$, which can be directly calculated from the observed $\tilde{A}$. Hence, we can simply use $\bar{\tilde{z}}_j$ to replace $\bar{x}_j$ in (7.9),(7.10), (7.13), and (7.14) . We can see that both expectation and variance are functions of node degree.

We can regard node i as a suspect if the corresponding entry $\tilde{y}_{ji}$ is within the

confidence interval $\mathbf{E}(\tilde{y}_{ji}) \pm \epsilon\sqrt{\mathbf{V}(\tilde{y}_{ji})}$ where $\epsilon > 0$ denotes the $\frac{1+p}{2}$ quantile of the standard normal distribution (i.e., interval $[-\epsilon, \epsilon]$ covers probability p). In our work, we choose $\epsilon = 2$ if not otherwise noted, and the confidence interval covers more than probability 0.954.



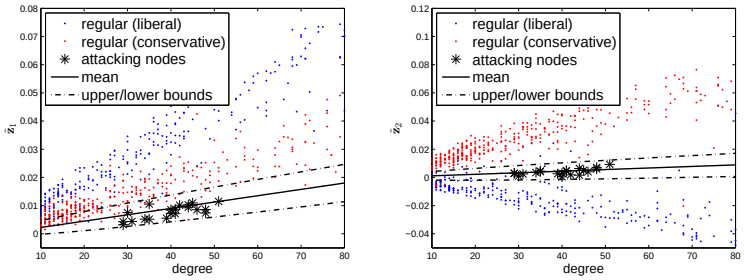
(a) $\tilde{\boldsymbol{z}}_1$ $(m_c = 0, \bar{v}_i = 60)$ (b) $\tilde{\boldsymbol{z}}_2$ $(m_c = 0, \bar{v}_i = 60)$



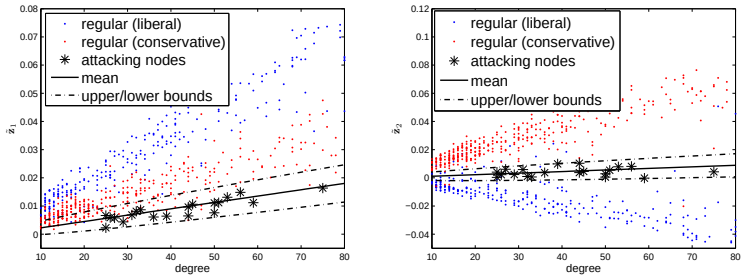
(c) $\tilde{\boldsymbol{z}}_1$ $(m_c = 0, \bar{v}_i = 30)$ (d) $\tilde{\boldsymbol{z}}_2$ $(m_c = 0, \bar{v}_i = 30)$

Figure 7.3: Spectral plot of RLAs in *polblogs* network, 20 attacking nodes. Attackers do not establish any connections among themselves.

Figure 7.3 and Figure 7.4 plot $\tilde{\boldsymbol{z}}_1$ vs. degree (or $\tilde{\boldsymbol{z}}_2$ vs. degree) of both attacking nodes and regular nodes under various RLA attacking schemes (with and without collaboration). Figure 7.3(c) and Figure 7.3(d) correspond to one independent RLA with 30 attacking nodes and each having 30 outgoing links on average ($\bar{v}_i = 30$), whereas Figure 7.3(a) and Figure 7.3(b) corresponds the RLA with each attacking node having 60 outgoing links on average ($\bar{v}_i = 60$). The number of victims of attacking node i is uniformly chosen from the interval $[\bar{v}_i - 10, \bar{v}_i + 10]$. We plot the mean value (the black line) and the upper and lower bounds (the dashed lines). We can observe that a majority of attacking nodes (denoted as black) form a region



(a) $\tilde{\boldsymbol{x}}_1$ ($p_{\text{in}} = .5, \bar{v}_i = 30$) (b) $\tilde{\boldsymbol{x}}_2$ ($p_{\text{in}} = .5, \bar{v}_i = 30$)



(c) $\tilde{\boldsymbol{x}}_1$, degree attacks (d) $\tilde{\boldsymbol{x}}_2$, degree attacks

Figure 7.4: Spectral plot of RLAs in *polblogs* network, 20 attacking nodes. Attackers establish some connections among themselves.

which locates within 2 standard deviations from the mean values while the majority of regular nodes locate in different regions. Hence we can regard nodes within two dashed lines as suspects. We can also observe that the more outgoing links (victims) of attacking nodes, the farther their spectral coordinates are away from those regular ones.

In the next attacking schemes, we introduce various inner link structures in the attacking subgraph. In Figure 7.4(a) and Figure 7.4(b), connections among attacking nodes follow the ER-model with probability 0.5 In Figure 7.4(c) and Figure 7.4(d), we masquerade attacking nodes as good users by specifying a degree distribution for connections among attacking nodes with the same parameters as the regular nodes (degree attack). Our result shows that we can successfully separate attacking nodes from regular ones no matter how the malicious creates links among attacking nodes. In other words, our method is robust with the subgraph C formed by attacking nodes.

When attackers do collaborate with each other, $\tilde{y}_{ji}$ still follows the normal distribution. However, we only have the upper bound of its expectation and variance as shown in (7.13). We shown the formula of the derived confidence interval in our next result.

Result 7.4: Given probability $p \in [0,1]$, let $\epsilon > 0$ denote the $\frac{1+p}{2}$ quantile of the standard normal distribution (i.e. interval $[-\epsilon, \epsilon]$ covers probability p). RLAs. For an attacker i with observed degree $d_i \leq \frac{N}{2}$, $\tilde{y}_{ji}$ has probability more than p to fall into the interval $[\tau_{\text{lw}}, \tau_{\text{up}}]$, where

$$\tau_{\text{up}} = \frac{d_i \bar{\tilde{z}}_j}{\tilde{\lambda}_j} + \frac{\epsilon}{\tilde{\lambda}_j}\left[\frac{d_i}{N}\left(1 - \frac{d_i}{N}\right)\right]^{\frac{1}{2}}, \tag{7.17}$$

$$\tau_{\text{lw}} = \frac{v^* \bar{\tilde{z}}_j}{\tilde{\lambda}_j} - \frac{\epsilon}{\tilde{\lambda}_j}\left[\frac{v^*}{N}\left(1 - \frac{v^*}{N}\right)\right]^{\frac{1}{2}}, \tag{7.18}$$

and $v^* = \min\{d_i, \frac{N}{2}[1 - \frac{N\bar{x}_j}{(\epsilon^2 + N^2\bar{x}_j^2)^{1/2}}]\}$.

Proof. If node i perform the RLA, $\tilde{y}_{ji}$ is normally distributed, and has probability p to fall into interval $[\tau_1, \tau_2] = \mathbf{E}(\tilde{y}_{ji}) \pm \epsilon\sqrt{\mathbf{V}(\tilde{y}_{ji})}$. When $v_i \leq d_i \leq \frac{n}{2}$, with (7.13), it is easy to get that the upper bound of τ_2:

$$\tau_2 \leq \frac{d_i \bar{x}_j}{\lambda_j} + \frac{\epsilon}{\lambda_j}\left[\frac{d_i}{n}\left(1 - \frac{d_i}{n}\right)\right]^{\frac{1}{2}}. \tag{7.19}$$

Note that the upper bound is obtained when attacker i does not collaborate with other attackers.

Next, we obtain an lower bound for τ_1. Neglecting terms of λ_j^{-2} and higher in (7.11) and (7.12), we have that τ_1 can be expressed as a function of v_i:

$$\tau_1(v_i) = \mathbf{E}(\tilde{y}_{ji}) - \epsilon\sqrt{\mathbf{V}(\tilde{y}_{ji})} = \frac{v_i \bar{x}_j}{\lambda_j} - \frac{\epsilon}{\lambda_j}\left[\frac{v_i}{n}\left(1 - \frac{v_i}{n}\right)\right]^{\frac{1}{2}}. \tag{7.20}$$

By taking the derivative and setting it to be 0, we have that $\tau_1(v_i)$ reaches the minimum when

$$v_i = v_{\min} := \frac{n}{2}\left[1 - \frac{n\bar{x}_j}{(\epsilon^2 + n^2\bar{x}_j^2)^{\frac{1}{2}}}\right]. \tag{7.21}$$

Moreover, $\tau_1(v_i)$ is a decreasing function when $0 \leq v_i \leq v_{\min}$, and is an increasing function when $v_i > v_{\min}$. Notice that $v_i \in [0, d_i]$, and we thus have $\tau_1 \leq \tau_1(\min\{d_i, v^*\})$, and the lower bound is obtained when attacker i attacks $\min\{d_i, v^*\}$ victims.

When the graph is large and the number of attackers c is much smaller than the graph size N, $\bar{x}_j \approx \bar{\tilde{z}}_j$ (with Lemma 7.1), $\tilde{\lambda}_j \approx \lambda_j$ (with the Weyl's Theorem) and $N \approx n$. By substituting the unknown values ($\bar{x}_j$, λ_j and n) with those values obtainable from $\tilde{A}$ ($\bar{\tilde{z}}_j$, $\tilde{\lambda}_j$ and N) in (7.19), (7.20) and (7.21), we get the result. □

Although we can filter out attackers by checking whether their spectral coordinate values locate within the confidence interval at each dimension, it is very tedious in practice. Users would prefer a single metric to quantify each node's likelihood of being an attacker. Next, we adopt a combined metric, the node non-randomness, to identify suspects.

In Section 6.2 we presented a framework that can quantify non-randomness at all granularity levels from edge, node, subgraph, to the overall graph. All graph non-randomness measures can be obtained mathematically from the spectra of the adjacency matrix of the network. The node non-randomness can be calculated as $R_u = \sum_{j=1}^{k} \lambda_j x_{ju}^2 = \boldsymbol{\alpha}_u \Lambda_k \boldsymbol{\alpha}_u^T$, where $\Lambda_k = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$, which means the non-randomness of node u is the length of its spectral vector with eigenvalue weighted on corresponding dimensions. As pointed by [90], eigenvalues and eigenvectors with large eigen-gaps are generally more stable under perturbation. In our work we choose k so that $\lambda_k - \lambda_{k+1}$ is maximized.

Result 7.5: For attacking node i, its node non-randomness is defined as

$$R_i = \tilde{\boldsymbol{\beta}}_i \tilde{\Lambda}_k \tilde{\boldsymbol{\beta}}_i^T = \sum_{j=1}^{k} \tilde{\lambda}_j \tilde{y}_{ji}^2. \tag{7.22}$$

Then, the expectation and variance of R_i have the following upper bounds respec-

tively: $\mathbf{E}(R_i) \leq U_i^{\mathbf{E}}$, and $\mathbf{V}(R_i) \leq U_i^{\mathbf{V}}$, where

$$U_i^{\mathbf{E}} = d_i^2 \sum_{j=1}^{k} \frac{\bar{x}_j^2}{\lambda_j} + \frac{d_i}{n}(1 - \frac{d_i}{n}) \sum_{j=1}^{k} \frac{1}{\lambda_j}; \tag{7.23}$$

$$U_i^{\mathbf{V}} = \frac{4d_i^3}{n}(1 - \frac{d_i}{n}) \sum_{j=1}^{k} \frac{\bar{x}_j^2}{\lambda_j^2} + \frac{2d_i^2}{n^2}(1 - \frac{d_i}{n})^2 \sum_{j=1}^{k} \frac{1}{\lambda_j^2}. \tag{7.24}$$

$\mathbf{E}(R_i)$ and $\mathbf{V}(R_i)$ reach the upper bounds when the attackers do not collaborate ($m_c = 0$).

Proof. For attacking node i, $R_i = \sum_{j=1}^{k} \lambda_j \tilde{y}_{ji}^2$. Since $\tilde{y}_{ji}$ is normally distributed with mean μ_{ji} and variance σ_{ji}^2, $\tilde{y}_{ji}^2/\sigma_{ji}^2$ follows the noncentral χ^2-distribution with degree of freedom 1 and parameter μ_{ji}^2/σ_{ji}^2, and hence we have

$$\mathbf{E}\left(\frac{\tilde{y}_{ji}^2}{\sigma_{ji}^2}\right) = 1 + \frac{\mu_{ji}^2}{\sigma_{ji}^2}, \quad \mathbf{V}\left(\frac{\tilde{y}_{ji}^2}{\sigma_{ji}^2}\right) = 2 + \frac{4\mu_{ji}^2}{\sigma_{ji}^2}.$$

Then, $\mathbf{E}(\tilde{y}_{ji}^2) = \mu_{ji}^2 + \sigma_{ji}^2$, and we have

$$\mathbf{E}(R_i) = \sum_{j=1}^{k} \lambda_j \, \mathbf{E}(\tilde{y}_{ji}^2) = \sum_{j=1}^{k} \lambda_j (\mu_{ji}^2 + \sigma_{ji}^2).$$

Substitute μ_{ji} and σ_{ji}^2 with their upper bounds shown in (7.13), we get the upper bound of $\mathbf{E}(R_i)$ shown in (7.23).

Similarly, $\mathbf{V}(\tilde{y}_{ji}^2) = 2\sigma_{ji}^4 + 4\mu_{ji}^2\sigma_{ji}^2$, and hence

$$\mathbf{V}(R_i) = \sum_{j=1}^{k} \lambda_j^2 \, \mathbf{V}(\tilde{y}_{ji}^2) = \sum_{j=1}^{k} \lambda_j^2 (2\sigma_{ji}^4 + 4\mu_{ji}^2\sigma_{ji}^2).$$

Substitute μ_{ji} and σ_{ji}^2 with their upper bounds shown in (7.13), we get the upper bound of $\mathbf{V}(R_i)$ shown in (7.24). □

R_i is non-negative and it is naturally lower bounded by zero. R_i is a linear combination of noncentral χ^2 random variables. The distribution of such type of random variable was given in [79]. We can then choose a proper ϵ such that, for attacking node i, R_i has a high probability to fall into the interval $[0, \mu + \epsilon\sigma]$. Substituting the

mean and standard deviation with their upper bounds, we can regard node i as a suspect if

$$R_i \leq U_i^{\mathbf{E}} + \epsilon (U_i^{\mathbf{V}})^{\frac{1}{2}}. \tag{7.25}$$

Similarly, we can replace the unknown variables ($\bar{x}_j$, λ_j and n) in (7.23) and (7.24) by those values obtainable from $\tilde{A}$ ($\bar{\tilde{z}}_j$, $\tilde{\lambda}_j$ and N). The non-randomness test to identify suspects is shown in Procedure 3.

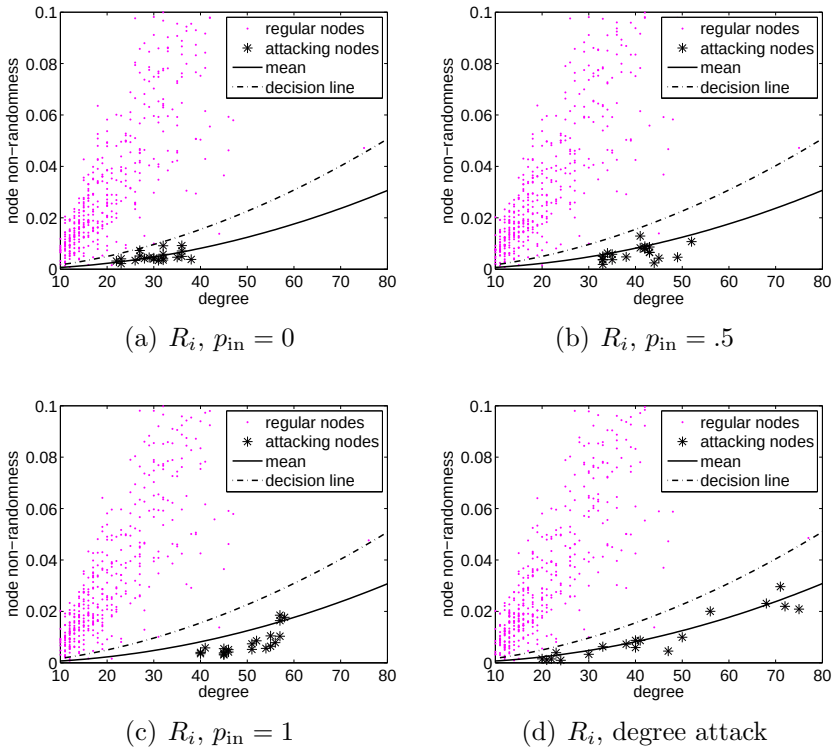
Procedure 3 Node non-randomness test

Input: $\tilde{A}$, **Output:** suspect set V_{susp}

1: Calculate $\tilde{\lambda}_j$, $\tilde{\boldsymbol{z}}_j$ and $\bar{\tilde{z}}_j$ from $\tilde{A}$, $j = 1, \ldots, k$;
2: **for** $i \leftarrow 1$ to N **do**
3: Calculate node non-randomness R_i by (7.22);
4: Calculate $U_i^{\mathbf{E}}$, $U_i^{\mathbf{V}}$ by (7.23) and (7.24);
5: **if** $R_i \leq U_i^{\mathbf{E}} + \epsilon (U_i^{\mathbf{V}})^{\frac{1}{2}}$ **then**
6: $V_{\text{susp}} \leftarrow V_{\text{susp}} \cup \{i\}$;
7: **end if**
8: **end for**

Figure 7.5 plots R_i vs. degree under four RLA attacking scenarios. The inner structure of the attacking groups is formed by Erdos-Renyi model [30] with parameter p_{in}: any two attacking nodes are connected with probability p_{in}. The black line shows the expected value of attacking nodes. The dashed pink line is the decision line corresponding to the upper bound as shown in (7.25). We regard nodes under the decision line as suspects. We can observe that for all four attacking scenarios non-randomness values of those fraud nodes are well below the decision lines, which indicates that our node non-randomness test is robust with any inner structure of the subgraph formed by attackers.

Comparison with Topology Based Testing. The authors in [88] first formalized the RLA property. Basically, an attack set A is called a RLA iff it satisfies 1) $|A| \leq k$; 2)the size of the victim set is larger than a constant (α) factor of the attack set; and 3) the number of distinct external triangles (formed by attackers and with the rest of the graph) $\triangle_A \leq \theta$. They proposed two tests, the clustering test and the neighborhood



(a) R_i, $p_{\text{in}} = 0$ (b) R_i, $p_{\text{in}} = .5$

(c) R_i, $p_{\text{in}} = 1$ (d) R_i, degree attack

Figure 7.5: Node randomness measure of various RLAs (20 attackers). (a)-(c): *ER* attacks with parameter p_{in} and $\bar{v}_i = 30$; (d): degree attacks.

independence test, to identify suspects. Identification of the suspect set is the first step towards finding an RLA, i.e., the identification of a set of suspect nodes that are potentially part of the attacking group.

A node in the graph is marked as a suspect if it fails either the clustering property or the neighborhood independence property. A node i satisfies the clustering property iff $\triangle_i \geq \rho d_i(d_i - 1)/2$ where $\triangle_i$ denotes the number of triangles of node i. A node i satisfies the neighborhood independence property iff the size of the maximum independent set in the neighborhood of node i[4] is less than $\alpha - \frac{\theta}{k}$ (more accurately, see inequality (5) in [88]). There are several limitations of this topology based test approach. First, the testing procedures contain too many parameters. In practice, it is hard to determine those parameters properly for a given network. Hence, the

[4]In the independence test, an independent set is defined as a set of nodes such that no two nodes share an edge.

identified suspect set contains a large number of false positives (good nodes marked as suspects). Second, the neighborhood independence test involves huge computational cost since it needs to find the maximum independent set for each node.

Table 7.1: Testing results for *polblogs* network, 20 attackers in all cases ($\rho = 0.0865$ in clustering test). ssp: number of suspects found by the testing; atk: true attackers in the suspects.

	indpdt.	*ER* attacks, $\bar{v}_i = 30$			degree
	$\bar{v}_i = 60$	$p_{\text{in}} = 0$	$p_{\text{in}} = .5$	$p_{\text{in}} = 1$	attack
	ssp atk	ssp atk	ssp atk	ssp atk	ssp atk
	Node nonrandomness test				
	22 20	22 20	22 20	23 20	21 19
θ	Cluster and Neighborhood independence test				
100	63 20	211 20	210 20	212 20	282 16
150	68 20	240 20	236 20	236 20	321 18
200	74 20	293 20	291 20	290 20	386 20
250	76 20	324 20	326 20	318 20	440 20
300	84 20	390 20	398 20	396 20	496 20

Table 7.1 shows the comparison between the topology based test approach and our spectrum based test approach for *polblogs* under five attacking scenarios. For each attack, we report the number of suspect nodes identified by each approach and the number of true attacking nodes among the identified suspect nodes. We follow their strategy by setting $\rho = .0865$ so that 95% of the nodes in the original graph satisfy the clustering property. We vary the θ with different values in the neighborhood independence test. We can see that for all five attacking scenarios (especially degree attack) the topology based test generates a large number of false positives, which will affect both accuracy and efficiency of catching the true attacking groups in the next step. In contrast, our spectrum based test results in much fewer false positives across all attacking scenarios.

7.3.2 Spectrum Based RLA Detection Algorithm

In this section, we present our spectrum based detection algorithm called SPCTRA to catch RLAs. The algorithm consists of three steps, as shown in Algorithm 7. In

Algorithm 7 SPCTRA: Spectrum based RLA Detection

Input: $\tilde{A}$, **Output:** attacking groups

1: $V_{\text{susp}} \leftarrow$node nonrandomness test on $\tilde{A}$ by Procedure 3;
2: $G_{\text{susp}} \leftarrow$ subgraph formed by V_{susp};
3: Find dense subgraphs H_s, $s = 1 \ldots, l$, in G_{susp} via the algorithm in [18];
4: **for** each dense subgraph H_s **do**
5: **if** H_s pass node nonrandomness test **then**
6: **return** H_s as an attacking group
7: **end if**
8: **end for**

the first step, we conduct node non-randomness test to identify suspects as shown in Procedure 3.

The subgraph G_{susp} formed by suspects expect to contain most attacking nodes as well as some regular ones. It is unlikely that regular nodes in the suspect set form dense subgraphs in G_{susp}. In the second step, from G_{susp} we identify suspect groups as candidates of RLA groups. In our work, we implemented the greedy algorithm [18] to approximately find dense subgraphs in $O(n_{\text{susp}})$ time. The algorithm starts with the full graph (G_{susp} in our setting). At each iteration, the node with minimum degree is delete, and the density of the remaining subgraph is calculated and recorded. The subgraph with the maximum density is finally returned.

In the third step, we test whether each dense subgraph is a true RLA group. As a result, we can filter out dense subgraphs accidentally formed by regular nodes. We regard nodes in a suspect group H_s as one single super-node regardless its inner structure. If H_s is mainly formed by a true RLA, the super-node expects to behave as an independent RLA node. We then check the super-node using the node non-randomness test. This requires the calculation of the eigenvector entries of the super-node. With Result 7.2, we can approximate the corresponding eigenvector entries. Let Ω_{H_s} denote the set of nodes connected to H_s. Then, its corresponding entry in

the j-the eigenvalue and the node non-randomness measure can be approximated by:

$$\tilde{y}_{j,H_s} = \frac{1}{\tilde{\lambda}_j} \sum_{u \in \Omega_{H_s}} \tilde{x}_{ju}, \text{ and } R_{H_s} = \sum_{j=1}^{k} \tilde{\lambda}_j \tilde{y}^2_{j,H_s}.$$

Similarly, $U^{\mathbf{E}}_{H_s}$ and $U^{\mathbf{V}}_{H_s}$ can be calculated by replacing d_i with $d_{H_s} = |\Omega_{H_s}|$ in (7.23) and (7.24). If $R_{H_s} \leq U^{\mathbf{E}}_{H_s} + \epsilon (U^{\mathbf{V}}_{H_s})^{\frac{1}{2}}$, we consider H_s as a RLA group.

Complexity. Our algorithm involves the calculation of the first k eigenvectors of a graph in Line 1 of Procedure 3. In general, eigen-decomposition of an $n \times n$ matrix takes a number of operations $O(n^3)$. In our framework, we only need calculate the first k largest eigenvalues and their eigenvectors. Furthermore, adjacency matrices in our context are usually sparse and well structured. We implemented the Arnoldi/Lanczos algorithm [38] which generally needs $O(n)$ rather than $O(n^2)$ floating point operations at each iteration. The storage cost is reduced to $nO(k)+O(k^2)$. The greedy algorithm to find dense subgraphs from the suspect set, in Line 3 of Algorithm 7, takes $O(n_{\text{susp}})$.

The authors in [88] developed the Greedy algorithm to catch RLAs from the suspect set. The Greedy algorithm is to mine subgraphs satisfying the RLA-property, starting from the suspect nodes identified by two tests. It grows a potential attack cluster by iteratively adding nodes with a high degree of connectivity with the cluster. However, the Greedy algorithm does not scale well for large graphs since it makes many calls to the expensive Filter procedure (to find the maximum independent set). The time complexity of the neighborhood independence test is $O(\sum_i d_i^2) = O(m^2)$ (the approximation algorithm to find the maximum independent set around node i needs $O(d_i^2)$ time). To catch attacking groups among n_{susp} suspects, the Greedy needs $O(m^2_{\text{susp}})$ time. The authors also developed the second technique, triangle random walk (Trwalk), which performs a randomized graph traversal starting at each suspect. They reported the Trwalk is 8 to 10 times more efficient than the Greedy but Trwalk is less accurate than the Greedy.

7.4 Experimental Results

One challenging research issue is how to effectively separate fraud nodes under a set of mixed attacks from regular users in large networks. Since different attacks can form different topological patterns, we evaluate how effectively our SPCTRA algorithm can characterize them in large-scale networks. We also compare our algorithm with the topology based detection approach [88].

Data Set and Setting. We conducted experiments on the Web Spam Challenge 2007 data [6], which contains over 105 million pages in 114,529 hosts in the .UK domain. The number of links among these hosts is 1,836,228. It also contains a small labeled training data set (6,382 hosts) where 341 hosts were labeled as spam. Our node non-randomness test identified 1,127 hosts as suspects (i.e., their node non-randomness values are below the decision line), among which 54 hosts were labeled as spam in the training data set. These 54 nodes randomly attack 14,283 victims in total. We analyzed the inner subgraph formed by these 54 nodes is very sparse and there is no evidence to show they are from a collaborative random link attack.

To evaluate the efficacy of our algorithm, we generated a mixed instance of attacks and added them together to the original network. We then ran our algorithm to detect the attack set on this modified network. We implemented our spectrum based detection algorithm and the topology based detection algorithm [88] (including two testing procedures, *clustering test* and *neighborhood independence test*, and the GREEDY algorithm) in Matlab. We evaluated both run-time and the accuracy of catching RLAs. Our experiments were carried out on a Windows XP64 workstation with a 3.0 GHz Pentium-IV CPU and 2GB RAM.

Accuracy of Detecting RLAs. We generated 8 RLAs with varied sizes and connection patterns (links between attackers and victims and internal links among the attackers), as shown in Table 7.2. The total number of attacking nodes is 650 and the size of victims is 56,144. Our goal is to test whether algorithms (SPCTRA and

Table 7.2: Evaluation results on Web Spam data set, 8 RLA attacking groups, 650 total attackers, and 56144 total victims.

	setting		SPCTRA		GREEDY	
RLA	size	$\bar{v}_i$ p_{in}	ssp	atck	ssp	atck
1	50	100 .3	50	50	49	47
2	50	100 .6	50	50	0	0
3	50	100 1	50	50	50	50
4	50	200 .3	50	50	79	47
5	100	100 .3	100	100	3	3
6	50	degree	49	49	20	20
7	100	degree	97	97	6	6
8	200	degree	188	188	27	27
final results (total)			634	634	4534	200

GREEDY) can catch them and how accurate they achieve. Each algorithm output a set of suspect groups (i.e., RLA candidates).

Our SPCTRA algorithm (with $k = 3$) successfully identified all 8 RLAs and no false suspect group was reported. For the first five RLAs (generated by ER model), our SPCTRA algorithm achieved 100% accuracy and no false positive or false negative node was introduced, as shown in Table 7.2. Even for those degree based RLAs (6, 7, 8), our SPCTRA algorithm achieved more than 94% accuracy.

In contrast, the GREEDY algorithm (with the best chosen parameters: the maximum size of attacking groups is 200, $\rho = 0.008$, $\theta = 300$, $\alpha = 50$) output 4,534 suspect nodes. It successfully identified RLA 3, in which attackers form a clique among themselves. However, it missed most attacking nodes in other RLAs (e.g., 2,5,7). The number of true attacking nodes among the output suspect groups was only 200. Table 7.2 shows our detailed comparisons.

Table 7.3: Execution time (in seconds) of for different data sets

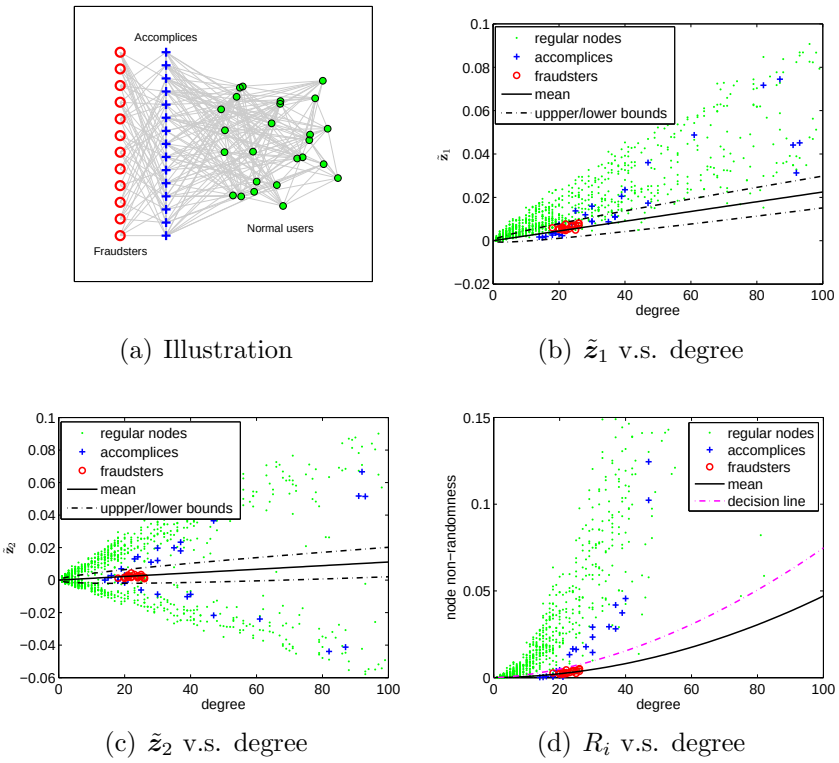
Data set	Alg.	Testing	Grouping	Total
polblogs (1222, 16714)	SPCTRA	0.037	0.041	0.078
	GREEDY	16.20	6.047	22.24
Web Spam (33%) (37562, 199406)	SPCTRA	0.702	0.239	0.941
	GREEDY	577.2	515.4	1093
Web Spam (114529, 1836228)	SPCTRA	4.017	29.68	33.69
	GREEDY	12728	83314	96043

Running Time. In this experiment, we compare the running times of the SPCTRA and GREEDY algorithms using three data sets, *polblogs*, Web Spam, and a sample of Web Spam data. In Table 7.3, we report the running time for both SPCTRA and GREEDY including the testing step (catching suspects) and the grouping step (catching RLAs). We can see that the time taken by GREEDY is 285, 1161, and 2851 times more than our SPCTRA algorithm. For example, the GREEDY takes more than 23 hours for the Web Spam Data while our SPCTRA takes only 34 seconds. Although the TRWALK improves the efficiency to some extent, it is still not scalable for large social networks since it uses the same expensive testing procedures.

7.5 Summary and Future Work

There are other types of collaborative attacks that can be considered as RLA variations.

Bipartite Core Attack. Pandit et. al. [19, 77] uncovered a new type of attack called the *bipartite cores* attack in auction networks. As illustrated in Figure 7.6(a), there are two types of identities: fraudsters (denoted as red cycle) and accomplices (denoted as blue +). The attacker creates some fake identities (denoted as fraudsters). The attacker is also capable of controlling a small set of regular identities (denoted as accomplices). Accomplices behave like honest users, except that they interact heavily with the set of fraudsters in order to boost their reputation. The fraudster identities are the ones used eventually to carry out the actual fraud. To avoid detection, the fraudsters or accomplices have few links to nodes of the same type, and they hence form a (nearly) bipartite subgraph in the network. In the bipartite core attacks, we assume that each regular node has the same probability to be controlled by the attacker. Then, the links between the accomplices and the fraudsters follow the random pattern, and the bipartite core attack can be regard as a special type of RLA. Therefore, the node non-randomness R_i for the frausters should also satisfy Result 7.5.



(a) Illustration (b) $\tilde{z}_1$ v.s. degree

(c) $\tilde{z}_2$ v.s. degree (d) R_i v.s. degree

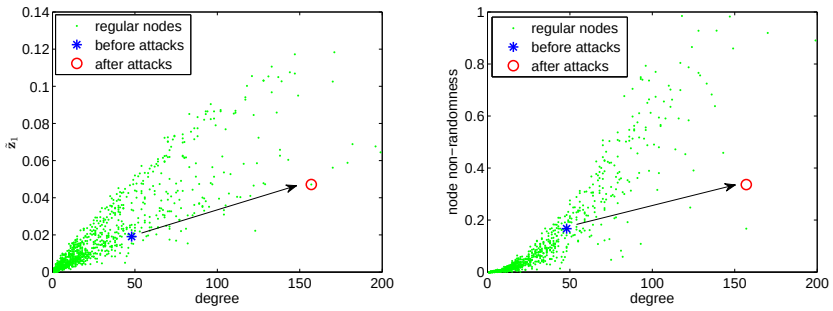
Figure 7.6: The spectral patterns for *polblogs*: bipartite core attacks with 20 fraudsters and 30 accomplices

Figure 7.6 shows the spectral patterns of the bipartite core attacks for *polblogs* network. The dashed pink line in Figure 7.6(d) is the decision line under which nodes are regarded as suspects of fraudsters. We can easily observe that the fraudsters nodes locate in the region different from regular nodes, and the distribution of fraudsters is also characterized by the means and variances shown in Result 7.3 and 7.5. The accomplices which are from the regular nodes are not distinct from other regular nodes. However, once the majority of fraudsters are captured, we can further detect related accomplices by searching the links.

DDoS Attacks. In the distributed denial of service attack (DDoS), the attacker randomly control a large number of regular nodes and create links from the controlled nodes to a target node, and the links between the target node and the controlled nodes follow the random pattern. Meanwhile, the target node also have its regular links,

but the proportion of regular links is small compared with the attacking links.

Figure 7.7 shows one DDoS attack on the *polblogs* network. In this attack, we randomly selected 10% nodes as attacking nodes and used them to attack one victim node. The spectral positions of the victim node before (blue star) and after (red circle) attacks are shown in the figure. By comparing spectral projects before and after attacks, we can easily locate the victim node and then identify attacking nodes through links connected to the victim. We can see from Figure 7.7 that the victim's non-randomness value remains almost unchanged while its degree increases significantly. This is very different from the change pattern of regular nodes' non-randomness values during graph evolution.



(a) $\tilde{\boldsymbol{z}}_1$ vs. degree (b) R_i vs. degree

Figure 7.7: DDoS attacks on *polblogs* network.

In summary, we have presented a novel framework that exploits the spectral space of underlying network topology to identify frauds or attacks. Our theoretical results showed that attackers locate in a different region of the spectral space from regular users. By identifying fraud patterns in graph spectral spaces, we can detect various collaborative attacks that are hard to be identified from original topological structures. Focusing on RLAs, we presented an efficient algorithm, SPCTRA, and compared with the topology based detection approach [88]. Empirical evaluations show that our approach significantly improves both effectiveness and efficiency especially when a mix of RLAs are introduced.

In our future work, we will explore various other attacking scenarios in both social networks and communication networks. Specifically, we will study how our spectrum based detection works when attackers choose victims purposely (rather than randomly) or only attack very few victims when they launch their collaborative attacks. For example, in the *passive and active attacks* [8], an adversary may only identify targeted individuals or derive sensitive relationships between targeted individuals from published node-anonymized social networks. The adversary constructs a highly distinguishable subgraph with edges to a set of targeted nodes, and then re-identifies the subgraph and consequently the targets in the released anonymized network. We will explore the effectiveness and efficiency of using spectral characteristics to detect the attacking subgraph. Another example is the *Sybil attack*. In a direct Sybil attack [75], a physical device controlled by adversaries may demonstrate multiple identities to its neighbors. A Sybil attack usually forms a group that has very few edges with the victims. We will empirically study whether we can still use the spectral characteristics to identify them. In practice, networks are constantly changing as both legitimate and fraud nodes (edges) can be added as networks evolve. Since the detection of some interested events depends on observing anomaly changes of network parameters over a time duration, we will consider time an important dimension in data representation and attack detection. We will extend our approach to use the temporal information in these evolving networks to identify and catch potential attacks. We will also explore matrix visualization and organization approaches that enable interactive navigation between network topology and its spectral spaces.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

In our work, we investigated the application of graph randomization techniques on social network data to preserve both data privacy and data utility. We conducted theoretical studies and empirical evaluations on the tradeoff between utility and privacy of various graph randomization techniques as well as investigation of some potential attacking methods from adversaries.

We studied various adjacency spectral properties of many real-world networks as well as some random ones. To quantify the data utility, we developed a consistent framework of non-randomness measures and applied it to community partition and fraud detection in social network settings. Extensive theoretical analysis and empirical evaluations were conducted to demonstrate the efficacy of our developed techniques.

8.1 Privacy Analysis of Social Network Randomization

Privacy Disclosure Risks. We first investigated the link disclosure risk of a randomized graph. To quantify the link disclosure risk, we considered the three probabilities on a sensitive link (i, j): the prior probability, the posterior probability given the existing (or non-existing) link (i, j) in the randomized graph, and the enhanced posterior probability utilizing the observed link as well as the some proximity measure between the two nodes.

For *Rand Add/Del* and *Rand Switch* randomization procedure, we established the relationship between the posterior probabilities and the magnitude of randomization. Not surprisingly, the posterior probabilities decrease as the randomization magnitude increases. We then calculated the minimal randomization magnitude needed to

protect the link privacy to a tolerable level.

The enhanced posterior probability based on proximity measures was derived for *Rand Add/Del* procedure. We studied the attacking model in which the attacker can learn the correlation between the proximity value and the existence of a link in the randomized graph, estimate the correlation in the original graph, and improve his prediction of sensitive links. Our empirical results showed that the enhanced posterior probability can greatly increase the attacker's ability of inferring the presence of a link and the prediction accuracy for those links between nodes with high similarity values.

We also proved one important property of the prior probability, posterior probability, and the enhanced posterior probability: the summations of the three probabilities are all (or approximately) equal to the total number of edges m. This equality indicates that, the more information the attacker utilizes, the higher accuracy he can achieve in predicting sensitive links.

In addition to the link disclosure risk, we investigated the identity disclosure risk of randomization. Our studies were based on one most widely used type of background knowledge, node degree. Although the nodes' degrees in the randomized graph are generally different from those in the original graph, our theoretical studies showed that the attacker could still estimate the degree sequence of the original graph based on the degree sequence of the randomized graph. We also derived the minimal randomization magnitude needed to protect the identity privacy given tolerable level.

Feature Preserving Randomization. Our empirical evaluations showed that pure randomization can significantly incur the loss of graph utility. Therefore, we developed two randomization procedures to better preserve graph characteristics without sacrificing much privacy protection during randomization.

The spectrum based randomization procedures, *Spctr Add/Del* and *Spctr Switch*, preserve the data utility by preserving some certain eigenvalues of the graph matrices.

We derived the conditions that the edge modification will increase or decrease the randomized eigenvalues. Then, our algorithms carefully choose the edges to be modified during the randomization, so that the eigenvalues of the randomized graph are close to the original values. Our evaluations showed that, compared with *Rand Add/Del* and *Rand Switch*, *Spctr Add/Del* and *Spctr Switch* procedures can not only better preserve the targeted eigenvalues but also many topological features of networks.

One limitation of the spectrum based randomization procedures is that they only consider some certain eigenvalues. Hence to what extent they can preserve general topological features is not guaranteed. We further developed the Markov chain based randomization procedure that could preserve any graph feature specified by users. The users or analysts usually require some feature of the randomized graph is close to the original value, resulting in a feature constraint on the randomized graph. When a feature constraint is placed, we established the switch-based Markov chain that can access any graph with the original degree sequence and satisfying the feature constraint. Using Metropolis-Hastings sampling, a standard method for generating a Markov chain with a target distribution, our graph generator can output any graph in the graph space with equal probability. Note that this graph generator (possibly with some minor adjustment) can also be used in testing the significance of data mining results.

We further investigated the potential disclosure of sensitive links due to the preserved features. We studied the attacking model in which the attacker can estimate the probability of a true link by uniformly sampling the graph space and thus breach the link privacy. One interesting finding by our evaluation is that, for some features, the constraints do not increase the attacker's posterior belief on true links (compared with the randomization without constraints). This is very important to data miners, because if the graph feature they focus on is one of such features, the data owner can safely place the feature constraint on the randomized graph to preserve the data

utility without increasing privacy disclosure risks. The extent to which the feature constraint can increase the privacy disclosure risk depends on the characteristics of the graph space specified by the feature constraint, which we would like to further investigate as our future work.

8.2 Spectral Analysis of Social Network Randomization

We discovered that the adjacency spectral spaces of many real-world graphs have clear patterns different from random ones. If the graph has k disconnected communities, the spectral coordinates of the nodes lie only on the axes in the space spanned by the leading k eigenvectors of the adjacency matrix, forming k strictly orthogonal lines. Nodes from the same community all lie on one axis with the central nodes far from the origin and the noisy ones close to the origin. When the communities are loosely connected, the nodes form k quasi-orthogonal lines that are rotated away from the axes for a certain degree. Those nodes with links outwards their own communities would deviate away from the lines. This pattern is different from those displayed in the spectral space of normal or Laplacian matrix. In the normal or Laplacian matrix, the communities form some clusters, and hence it is usually difficult to distinguish the central nodes from the noisy ones.

Based on this phenomenon, we then developed a consistent framework to quantify the graph non-randomness at edge, node, subgraph and the overall graph levels. We showed that all graph non-randomness measures can be obtained mathematically from the spectra of the adjacency matrix of the network. A relative non-randomness measure of the overall graph was also presented. It allows quantitative comparisons between various social networks with different sizes and densities or between different snapshots of a dynamic social network. We proved that the relative non-randomness measure of any graph is lower (upper) bounded by the regular (complete) graph. We also analyzed the distributions of both edge and node non-randomness for real-world social networks and random graphs. Our results showed that edge non-randomness

and node non-randomness of real-world social networks usually display some high skewed distributions, obeying either a power law or an exponential law. On the contrary, random graphs display approximate normal distributions.

We further studied two applications of the non-randomness framework: community partition and fraud detection. Utilizing the property of line orthogonality, we developed the algorithm, *AdjCut*, to partition the communities by fitting orthogonal lines in the spectral space. Our algorithm achieves comparable accuracy with the normal spectrum based cutting algorithm and better than the Laplacian spectrum based algorithm. Our evaluations showed that the communities produced by our *AdjCut* algorithm have more balanced sizes. One difference between our *AdjCut* algorithm and the normal spectrum based cutting algorithm is that the normal spectrum based algorithm partitions the communities by cutting as few edges as possible, whereas our *AdjCut* algorithm does the partition by assigning a node to the community to which it has the strongest association.

The second application of the non-randomness measures we studied is the fraud detection in social network settings. We developed a novel framework that exploited the spectral space of underlying network topology to identify frauds or attacks. Our theoretical results showed that attackers locate in a region different legitimate users in the spectral space. Specifically, the spectral coordinate of an attacker is mainly determined by that of its victims. The inner structure among collaborative attackers has limited impact on attackers' distributions in the spectral space. By identifying fraud patterns in graph spectral spaces, we can detect various collaborative attacks that are hard to be identified via graph topology. For random link attacks, we proved that the spectral coordinates of the fraudulent nodes should follow the multivariate normal distribution in the adjacency spectral space. Therefore, the node non-randomness values of the fraudulent nodes follow the χ^2-distribution and are significantly lower than the values of the legitimate nodes. Based on this statistical property, we de-

veloped an efficient algorithm, SPCTRA, to filter the fraudulent nodes. Empirical evaluations showed that our approach significantly improved both effectiveness and efficiency especially when a mix of RLAs were introduced.

8.3 Future Work

In our future work, we would like to further study the privacy disclosure risks for the feature-preserving randomization procedures. As we discussed earlier, preserving some features would jeopardize the privacy while preserving some others would not. We are interested in what aspects of characteristics of the features make this difference. We will investigate how to efficiently randomize graphs to preserve multiple features, and study its impacts on privacy disclosure risks.

We will conduct comprehensive comparisons among the randomization, K-anonymity, and generalization based privacy-preserving techniques. Based on the background knowledge of nodes' degrees, some preliminary comparison results between the randomization and K-degree techniques have been reported in Section 4.2. More comparisons merit further study, especially when adversaries are able to exploit various complex background knowledge in their attacks.

We will also study the scalability issue of graph randomization techniques and conduct empirical evaluations on large social networks. The computational cost of some graph features can be very expensive, which raises an challenge to preserve such features for large social networks. For example, we may preserve the feature with high computational cost by preserving a highly correlated feature but with low computational cost, and approximate the feature rather than calculate it precisely during the randomization.

For spectral analysis of social network randomization, we have shown that the graph non-randomness measures are determined by the spectral space spanned by the leading k adjacency eigenvalues, where k indicates the number of communities in the graph. We will also investigate the full relationship between our proposed

non-randomness measures (especially the graph non-randomness) and the traditional topology based measures.

For the adjacency spectrum based fraud detection, we will explore more attacking scenarios in both social networks and communication networks. Specifically, we will study how our spectrum based detection works when attackers choose victims purposely (rather than randomly) or only attack very few victims when they launch their collaborative attacks. For example, in the *active* and *passive attacks*, an adversary may only attack a small number of targeted individuals and derive sensitive relationships among them. We will empirically study whether we can still use the spectral characteristics to identify the attacking nodes. In practice, networks are constantly changing as both legitimate and fraudulent nodes (edges) can be added as networks evolve. We will extend our approach to use the temporal information in these evolving networks to identify and catch potential attacks. We will also explore matrix visualization and organization approaches that enable interactive navigation between network topology and its spectral spaces.

We will investigate differentially private algorithms for social network data. It involves deriving accurate sensitivity values of various social network features and graph mining algorithms. The differential privacy mechanism is very different from those non-interactive social network release mechanisms which can be used to answer an unlimited number of queries. In our future work, we would like to compare the differential privacy and non-interactive social network release mechanisms for the social network data.

REFERENCES

[1] L. Adamic and N. Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

[2] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[3] D. Agrawal and C. Agrawal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th Symposium on Principles of Database Systems*, 2001.

[4] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 439–450. Dallas, Texas, May 2000.

[5] W. Aiello, F. Chung, and L. Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001.

[6] AIRWeb. Web spam challenge. In *http://webspam.lip6.fr/wiki/pmwiki.php*, 2007.

[7] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, 2006.

[8] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 181–190, 2007.

[9] J. Baumes, M. K. Goldberg, M. Magdon-Ismail, and W. A. Wallace. Discovering hidden groups in communication networks. In *ISI*, pages 378–389, 2004.

[10] T. Y. Berger-Wolf and J. Saia. A framework for analysis of dynamic social networks. In *KDD*, pages 523–528, 2006.

[11] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Class-based graph anaonymization for social network data. In *Proc. of 35th International Conference on Very Large Data Base*, 2009.

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

[13] P. Boykin and V. Roychowdhury. Leveraging social networks to fight spam. *COMPUTER*, pages 61–68, 2005.

[14] A. Bratko, B. Filipič, G. Cormack, T. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *The Journal of Machine Learning Research*, 7:2673–2698, 2006.

[15] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *PinKDD*, 2008.

[16] D. Chakrabarti and C. Faloutsos. Graph mining: laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2, 2006.

[17] P. Chan, M. Schlag, and J. Zien. Spectral k-way ratio-cut partitioning and clustering. In *Proceedings of the 30th international Design Automation Conference*, pages 749–754. ACM, 1993.

[18] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX '00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95, 2000.

[19] D. H. Chau, S. Pandit, and C. Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. In *PKDD*, pages 103–114, 2006.

[20] C. E. H. Chua and J. E. Wareham. Fighting internet auction fraud: An assessment and proposal. *COMPUTER*, pages 31–37, 2004.

[21] F. Chung and R. Graham. Sparse quasi-random graphs. *Combinatorica*, 22 (2): 217–244, 2002.

[22] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. In *Proc. of VLDB08*, pages 833–844, 2008.

[23] L. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Advances In Physics*, 56:167, 2007.

[24] D. Cvetkovic and P. Rowlinson. The largest eigenvalue of a graph: A survey. *Linear and multilinear algebra*, 28:3–33, 1990.

[25] D. Cvetkovic, P. Rowlinson, and S. Simic. *Eigenspaces of Graphs*. Cambridge University Press, 1997.

[26] S. Das, Ömer Egecioglu, and A. E. Abbadi. Anonymizing edge-weighted social network graphs. Technical report, UCSB CS, March 2009.

[27] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, 2001.

[28] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM (to appear)*, 2011.

[29] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography*, pages 265–284, 2006.

[30] P. Erdos and A. Renyi. On random graphs i. *Publicationes Mathematicae*, 6: 290–297, 1959.

[31] E. Estrada and J. A. Rodrguez-Velzquez. Subgraph centrality in complex networks. *Physical Review E*, 71(056103), 2005.

[32] I. J. Farkas, I. Derenyi, A.-L. Barabasi, and T. Vicsek. Spectra of "real-world" graphs: Beyond the semi-circle law. *Physical Review E*, 64:1, 2001.

[33] A. Fast, D. Jensen, and B. N. Levine. Creating social networks to improve peer-to-peer networking. In *KDD*, pages 568–573, 2005.

[34] Z. Furedi and J. Komlos. The eigenvalues of random symmetric matrices. *Combinatorica*, 1 (3):233–41, 1981.

[35] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman & Hall/CRC, 1996.

[36] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821–7826, June 2002.

[37] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam*, pages 30–31, 2004.

[38] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[39] L. Guo, S. Guo, and X. Wu. On addressing accuracy concerns in privacy preserving association rule mining. In *PAKDD*, pages 124–135, 2008.

[40] L. Guo, X. Ying, and X. Wu. On attribute disclosure in randomization based privacy preserving data publishing. In *IEEE International Workshop on Privacy Aspects of Data Mining (PADM10), in conjunction with ICDM10*, 2010.

[41] S. Guo, X. Wu, and Y. Li. Determining error bounds for spectral filtering based reconstruction methods in privacy preserving data mining. *Knowl. Inf. Syst.*, 17(2):217–240, 2008.

[42] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004.

[43] L. Hagen and B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on computer-aided design*, 11(9), 1992.

[44] S. Hanhijarvi, G. C. Garriga, and K. Puolamaki. Randomization techniques for graphs. In *Proc. of the 9th SIAM Conference on Data Mining*, 2009.

[45] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometria*, 57-1:97, 1970.

[46] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *University of Massachusetts Technical Report*, 07-19, 2007.

[47] M. Hay, G. Miklau, D. Jensen, D. Towsely, and P. Weis. Resisting structural re-identification in anonymized social networks. In *VLDB*, 2008.

[48] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *IEEE International Conference on Data Mining*, pages 169–178, 2009.

[49] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge, 1985.

[50] L. Huang, D. Yan, M. Jordan, and N. Taft. Spectral clustering with perturbed data. *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[51] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2005.

[52] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proc. of the 3rd Int'l Conf. on Data Mining*, pages 99–106, 2003.

[53] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[54] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[55] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[56] J. M. Kleinberg. Challenges in mining social network data: processes, privacy, and paradoxes. In *KDD*, pages 4–5, 2007.

[57] I. Koprinska, J. Poon, J. Clark, and J. Chan. Learning to classify e-mail. *Inf. Sci.*, 177(10):2167–2187, 2007.

[58] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity in networks. In *KDD*, pages 245–255, 2006.

[59] V. Krebs. http://www.orgnet.com/. 2006.

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

[61] V. Latora and M. Marchiori. Efficient behavior of small-world networks. *Physics Review Letters*, 87, 2001.

[62] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, 2003.

[63] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD Conference*, Vancouver, Canada, 2008. ACM Press.

[64] K. Liu, K. Das, T. Grandison, and H. Kargupta. Privacy-preserving data analysis on graphs and social networks, 2008.

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

[66] L. Lovasz. Random walks on graphs. *Combinatorics*, 2:1–46, 1993.

[67] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54:396–405, 2003.

[68] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. *l*-diversity: privacy beyond *k*-anonymity. In *Proceedings of the IEEE ICDE Conference*, 2006.

[69] S. P. Meyn and R. Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, London, 1993.

[70] R. Milo, N. Kashtan, S. Itzkovitz, M. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences, 2003. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0312028`.

[71] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Security & Privacy '09*, 2009.

[72] M. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter*, 38(2):321–330, March 2004.

[73] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[74] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, 2006.

[75] J. Newsome, E. Shi, D. X. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *IPSN*, pages 259–268, 2004.

[76] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference*, pages 849–856, 2001.

[77] S. Pandit, D. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210, 2007.

[78] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11:430, 1990.

[79] S. J. Press. Linear combinations of non-central chi-square variates. *The Annals of Mathematical Statistics*, 37-2:480–487, 1966.

[80] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: Output perturbation for queries with joins. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 107–116. ACM, 2009.

[81] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[82] S. Russell and P. Norvig. *Artifical Intelligence: A Modern Approach*. Pearson Education, 2003.

[83] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, volume 460, 1998.

[84] A. Seary and W. Richards. Spectral methods for analyzing and visualizing networks: an introduction. *National Research Council, Dynamic Social Network Modelling and Analysis: Workshop Summary and Papers*, pages 209–228, 2003.

[85] J. Shetty and J. Adibi. The enron email dataset database schema and brief statistical report. *Information Sciences Institute Technical Report, University of Southern California*, 2004.

[86] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[87] M. Shiga, I. Takigawa, and H. Mamitsuka. A spectral clustering approach to optimally combining numericalvectors with a modular network. In *KDD*, pages 647–656, 2007.

[88] N. Shrivastava, A. Majumder, and R. Rastogi. Mining (social) network graphs to detect random link attacks. In *ICDE*, pages 486–495, 2008.

[89] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *KDD*, pages 678–684, 2005.

[90] G. W. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[91] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *KDD*, pages 717–726, 2007.

[92] R. Taylor. Contrained switchings in graphs. *Combinatorial Mathematics VIII, Proceedings of the 8th Australian Conference on Combinatorial Mathematics*, pages 314–336, 1981.

[93] Z. Teng and W. Du. Comparisons of k-anonymization and randomization schemes under linking attacks. In *Proceedings of The IEEE International Conference on Data Mining (ICDM)*, 2006.

[94] F. Viger and M. Latapy. Fast generation of random connected graphs with prescribed degrees. In *Proc. 11th International Computing and Combinatorics Conference*, 2005.

[95] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, 2009.

[96] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. *Proceedings of the 22nd International Symposium on Reliable Distributed Systems*, 2003.

[97] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *IEEE International Conference on Computer Vision*, pages 975–982, 1999.

[98] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *KDD*, pages 266–275, 2003.

[99] L. Wu, X. Ying, and X. Wu. Reconstruction from randomized graph via low rank approximation. In *Proc. of the 10th SIAM Conference on Data Mining*, 2010.

[100] L. Wu, X. Ying, X. Wu, A. Lu, and Z.-H. Zhou. Spectral analysis of k-balanced signed graphs, 2011.

[101] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang. k-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, 2010.

[102] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 139–150, September 2006.

[103] X. Ying and X. Wu. On link privacy in randomizing social networks. In *PAKDD*, 2009.

[104] X. Ying and X. Wu. Graph generation with prescribed feature constraints. In *Proc. of the 9th SIAM Conference on Data Mining*, 2009.

[105] X. Ying and X. Wu. On randomness measures for social networks. In *SDM*, 2009.

[106] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *Proc. of the 8th SIAM Conference on Data Mining*, 2008.

[107] X. Ying, K. Pan, X. Wu, and L. Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In *SNA-KDD '09: Proceedings of the 3rd SIGKDD Workshop on Social Network Mining and Analysis*, 2009.

[108] X. Ying, L. Wu, and X. Wu. A spectrum-based framework for quantifying randomness of social networks. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2010.

[109] X. Ying, X. Wu, and D. Barbará. Spectrum based fraud detection in social networks. In *ACM Conference on Computer and Communications Security*, pages 747–749, 2010.

[110] X. Ying, X. Wu, and D. Barbará. Spectrum based fraud detection in social networks. In *IEEE International Conference on Data Engineering*, 2011.

[111] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

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

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

[114] B. Zhou, J. Pei, and W.-S. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations*, 10(2), 2009.

[115] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. In *VLDB*, 2009.