Graph Generation with Prescribed Feature Constraints

Xiaowei Ying, Xintao Wu
Department of Software and Information Systems
Univ. of North Carolina at Charlotte
{xying,xwu}@uncc.edu

Abstract

In this paper, we study the problem of how to generate synthetic graphs matching various properties of a real social network with two applications, privacy preserving social network publishing and significance testing of network analysis results. We present a simple switching based graph generation approach to generate graphs preserving features of a real graph. We then investigate potential disclosures of sensitive links due to the preserved features. Our algorithms on graph generation with feature range and feature distribution constraints are based on the Metropolis-Hastings sampling. This is of importance for significance testing of network analysis results.

1 Introduction

The management and analysis of social networks has attracted increasing interest 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 [6, 14].

Many applications of networks such as anonymous Web browsing require relationship anonymity due to the sensitive, stigmatizing, or confidential nature of relationship. It has been shown in [1,12] that the simple technique of anonymizing graphs by replacing the identifying information of the nodes with random ids before publishing the actual graph does not guarantee privacy since the identification of the vertices can be seriously jeopardized by applying subgraph queries. As a result, link randomization was suggested. However, link randomization may significantly affect the utility of the released randomized graph. To preserve utility, we expect certain aggregate characteristics (a.k.a., feature) of the original graph should remain basically unchanged.

In the first part of this paper, we study the problem of how to generate a synthetic graph matching various properties of a real social network. Previous work, which applied switching algorithms [22, 23] or matching algorithms [2, 15, 24, 29] to generate graphs satisfying only a given degree sequence, cannot guarantee that the generated graphs preserve various topological features of the real graph since many structural properties are not purely determined by the degree sequence. We present a switching based algorithm for generating synthetic graphs to preserve various features of the original graph. We then formally study how various features preserved in the released graph can be exploited by attackers to breach link privacy.

In the second part of this paper, we study the problem of how to generate a group of graphs for the purpose of significance testing of network analysis results. When assessing the significance of graph analysis results, we need the randomization to be controlled in such a way that some feature of the generated graphs follow a certain prescribed distribution, which may be quite different from the natural distribution of all graphs in the ensemble. We present our algorithm to serve this purpose. Our algorithms on uniform graph generation with feature range constraints or feature distribution constraints are based on the Metropolis-Hastings sampling [11].

The rest of this paper is organized as follows. In Section 2 we discuss the notations used in this paper and present preliminaries of Markov chain. In Section 3 we focus on generating a graph for privacy preserving social network publishing and in Section 4 we investigate potential disclosures of sensitive links due to preserved features. In Section 5 we further investigate how to generate graphs for the task of significance testing of network analysis results. We discuss related work in Section 6. Finally we offer our concluding remarks and discuss future work in Section 7.

2 Preliminaries

A network or graph G is a set of n nodes connected by a set of m links. The network considered here is binary, symmetric, and without self-loops. Let $A = (a_{ij})_{n\times n}$ be its adjacency matrix, $a_{ij} = 1$ if node i and j are connected and $a_{ij} = 0$ otherwise. Associated with A is the degree distribution $D_{n\times n}$, a diagonal matrix with row-sums of A along the diagonal, and 0's elsewhere. Table 1 summarizes the notation used in this paper.

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 [6]. In this paper, we consider the following two real space features and two spectrum features.

- λ_1, the eigenvalues of the adjacency matrix A.
- μ_2, the second eigenvalue of the Laplacian matrix

Table 1: Notation

Symbol	Definition
$G(n,m)$	a graph with n nodes and m edges
$\widetilde{G}$	the released graph by the data owner
$\widetilde{G}_s$	the graph samples generated by the attacker
G^t	the graph at time t in a Markov chain
$\mathcal{G}_{\boldsymbol{d}}$	set of the graphs with degree sequence $\boldsymbol{d}$
$\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$	set of the graphs satisfying constraint $\boldsymbol{S}$ in $\mathcal{G}_{\boldsymbol{d}}$
$q(G)$	$\mathcal{G}_{\boldsymbol{d}} \to [0,1]$, the target stationary probability of graph G in a graph generator
$\psi(x)$	$\mathbb{R} \to \mathbb{R}$, a function used in relaxed generator
$S, S(G)$	$\mathcal{G}_{\boldsymbol{d}} \to \mathbb{R}$, a feature of graph G
$f(x)$	$\mathbb{R} \to \mathbb{R}$, the natural distribution of S over $\mathcal{G}_{\boldsymbol{d}}$
$g(x)$	$\mathbb{R} \to \mathbb{R}$, the target stationary distribution of S in the graph generator
$d(\widetilde{G}_s, G)$	the proportion of different edges in $\widetilde{G}_s$

defined as $L = D - A$.

- h, the harmonic mean of the shortest distance [17].
- C, the transitivity measure [6]. The transitivity measure is one type of clustering coefficient which measure and characterizes the presence of local loops near a vertex.

It has been shown that the spectrum have close relation with the many graph characteristics and can provide global measures for many network properties [25]. For example, the maximum degree, chromatic number, clique number, and extend of branching in a connected graph are all related to λ_1. μ_2 is an important eigenvalue of the Laplacian matrix and can be used to show how good the communities separate, with smaller values corresponding to better community structures. It is important to point out that our algorithms we present in this paper are general enough to work with any other chosen features defined on graphs.

Throughout this paper, we conduct our empirical evaluation on four real networks: dolphins, Karate, polbooks, and Enron. The first three networks are from network benchmark datasets (`http://www-personal.umich.edu/~mejn/netdata/`). The *Enron* 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 [26]. This dataset contains 252,759 emails from 151 Enron employees, mainly senior managers, and we regard there is an edge between node i and j if there is at least 5 emails between them.

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

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

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

LEMMA 2.1. *[21] 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:*

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

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

3 Graph generation for privacy preserving social network publishing

In this section, we first revisit previous switching based method (shown in Algorithm 1) on generating graphs without feature constraints. We then extend this method to generate graphs with feature range constraints.

Algorithm 1 Uniform graph generator [27]

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;

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: $\quad$ Randomly pick up two edges (a,b) and (c,d) in G^t;
4: $\quad$ **if** edge (a,b) and (c,d) are switchable **then**
5: $\quad\quad G^{t+1} \leftarrow$switch (a,b) and (c,d) in G^t;
6: $\quad$ **end if**
7: **end if**

3.1 Graph generation without feature constraints 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 1. The algorithm uses a Markov chain to generate a random graph. 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 1. 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 1 can generate graphs uniformly from the ensemble of all graphs that have the given degree sequence from the original graph.

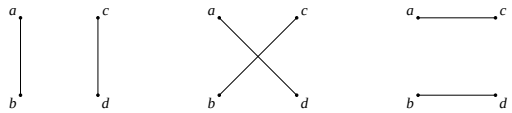


Figure 1: Switch edges

THEOREM 1. *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 1 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 [27], and with Lemma 2.1, 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 $\mathcal{G}_{\boldsymbol{d}}$, $p_{ij} := P[G^{t+1} = G_j | G^t = G_i] = \frac{1}{2m(m-1)}$ 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.

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 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 [22], and in [28] 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 [9].

Estimate the feature distribution over $\mathcal{G}_{\boldsymbol{d}}$ Since graphs obtained by Algorithm 1 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 1, then the unbiased estimator of $E[S(G)]$ and $Var[S(G)]$ over $\mathcal{G}_{\boldsymbol{d}}$ are given by:

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

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:

$$\hat{f}_h(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left[\frac{x - S(G_i)}{h}\right] \tag{3.1}$$

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

3.2 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[1]. 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 1 cannot preserve the utility of the original graph in general. Table 2 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.

For those samples generated from polbooks, Figure 2 plots the sample distribution of four features over $\mathcal{G}_{\boldsymbol{d}}$. We can observe that all features except μ_2 approximately follow normal distributions while μ_2 has a skewed distribution. This observation matches previous theoretical studies [8].

[1] 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.

Table 2: 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

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 2 outlines this algorithm[2].

Algorithm 2 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 2 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 2).

For those synthetic graphs, we also compute the means and standard deviations of other three uncontrolled features. Table 3 shows the means and standard deviations of the

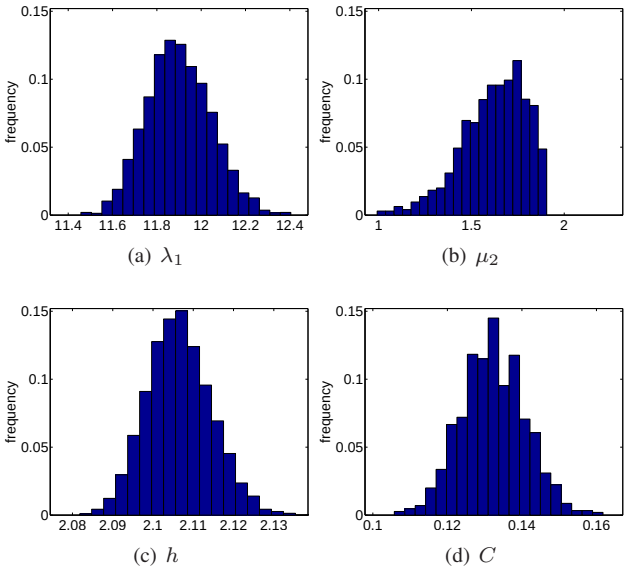


Figure 2: Feature Distributions over $\mathcal{G}_{\boldsymbol{d}}$ for U.S. politics books network

feature values of the generated graphs for four networks. By comparing with Table 2, 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.

4 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 [3] to breach link privacy. In Section 4.1, we present one attacking method and empirically show its effectiveness in breaching link privacy. In Section 4.2, we conduct theoretical analysis.

[2]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 2

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

Table 3: Feature means and standard deviations of synthetic graphs with feature constraints

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

4.1 Attacking method Let G and $\widetilde{G}$ denote the original graph and the released graph respectively. To simplify the notation, we also use G and $\widetilde{G}$ 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 edge (i,j), and hence

$$P[G(i,j)=1|\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}] = \frac{1}{|\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}|}\sum_{G_s\in\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}} G_s(i,j). \quad (4.2)$$

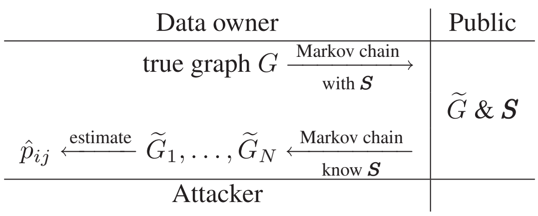


Figure 3: Graph publishing and attacking process

The attacking method works as follows. Starting with the released graph $\widetilde{G}$, attackers apply the same randomization strategy to generate N samples $\widetilde{G}_s$ $(s=1,2,\ldots,N)$. Then attackers calculate the posterior probability of existence of a link for all node pairs as $\hat{p}_{ij} = \frac{1}{N}\sum_{s=1}^{N}\widetilde{G}_s(i,j)$ and choose top t as predicted links. Figure 3 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 $\widetilde{G}$, 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 2, 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 4 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 $\mathcal{G}_{\boldsymbol{d}}$.

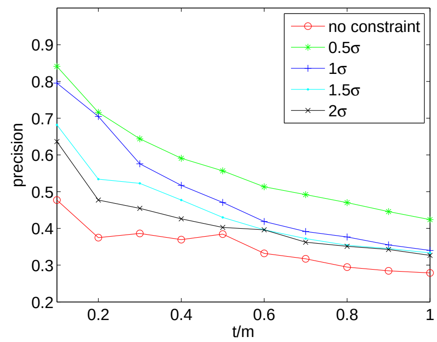


Figure 4: Precision of Top t predictions with μ_2 confined within different ranges for polbooks.

Figure 5 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.

We can also observe in Figure 5 that, when λ_1 or transi-

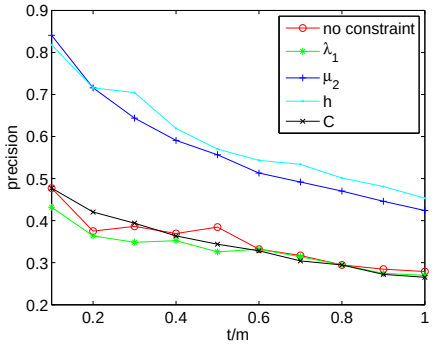


Figure 5: Precision of top t predictions for polbooks

tivity (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 Section 4.2.

4.2 Features vs. privacy From Figure 5, 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 6(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.

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 $\widetilde{G}_s$ and G have the same number of edges, it is easy to check that $\frac{1}{4}\|\widetilde{G}_s - G\|_F^2$ is the number of different edges, and we can then define the relative distance measure between the original graph and the synthetic graph:

$$d(\widetilde{G}_s, G) = \frac{\|\widetilde{G}_s - G\|_F^2}{2\|G\|_F^2} = \frac{\|\widetilde{G}_s - G\|_F^2}{4m}. \tag{4.3}$$

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

Table 4 lists the means and standard deviations of $d(\widetilde{G}_s, G)$ 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 $d(\widetilde{G}_s, G)$ is not much different from the case without constraints. However, when μ_2 or h is preserved, the mean of $d(\widetilde{G}_s, G)$ 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 $d(\widetilde{G}_s, G)$ 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 1. *Let $\bar{d}$ denote the expectation of $d(\widetilde{G}_s, G)$ over $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$:*

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

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

$$\sum_{i<j, ij \in G} \hat{p}_{ij} \to m(1 - \bar{d}). \tag{4.4}$$

Please refer to appendix for the proof.

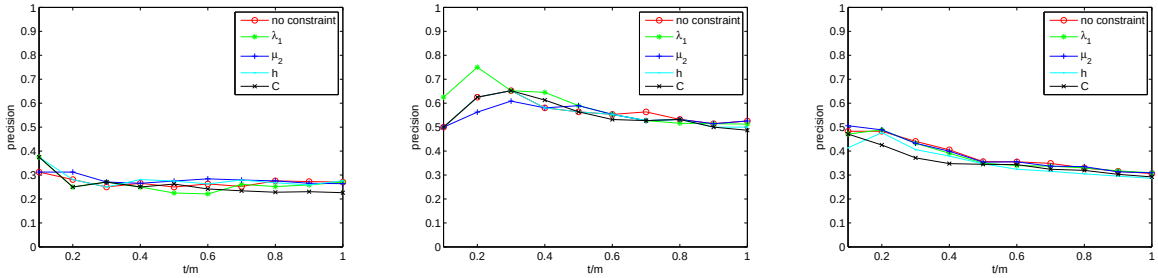
From Equation (4.4), we can see that if the constraint $\boldsymbol{S}$ specifies a graph space which has smaller average distance to the true graph (smaller $\bar{d}$), the true edges must have higher estimated posterior probability $\hat{p}_{ij}$. On the other hand, since

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

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.

5 Graph generation for significance testing of graph analysis results

In the following, 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 [11] is one of the standard methods of converting a Markov chain with one stationary distribution to another Markov chain with a different stationary distribution.



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

Figure 6: Precisions of top t predictions for different networks

Table 4: Means and standard deviations of $d(\widetilde{G}_s, G)$ 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

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

$$\alpha_{ij} = \min\left(1, \frac{q_j p_{ji}}{q_i p_{ij}}\right), \tag{5.5}$$

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

$$\alpha_{ij} = \min\left(1, q_j/q_i\right). \tag{5.6}$$

5.1 Relaxed graph generation with feature range constraints

Generally speaking, the graph generator with feature range constraint shown in Algorithm 2 may not access all the graphs that satisfies the constraint. To overcome this problem, we propose a relaxed algorithm in this section. The relaxed algorithm, shown in Algorithm 3, can access all the graphs in $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$ and achieve approximate uniformity.

Algorithm 3 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 1 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

$$q(G) = \frac{\psi[S(G)]}{K}, \tag{5.7}$$

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 3 only depends on the ratio of two probabilities, we can simply set

$$q(G) \leftarrow \psi[S(G)]. \tag{5.8}$$

The transition matrix in Algorithm 1 is symmetric, and we can thus set the acceptance ratio $q(G^t)/q(G^{t-1})$ as Equation (5.6). The connectivity of the Markov chain in Algorithm 3 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:

$$\psi(s) = \begin{cases} \frac{1}{\sigma_1\sqrt{2\pi}} \exp\left[-\frac{(s-s_0)^2}{2\sigma_1^2}\right], & \text{if } s \geq s_0 \\ \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left[-\frac{(s-s_0)^2}{2\sigma_2^2}\right], & \text{if } s < s_0 \end{cases} \tag{5.9}$$

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 Equation (5.9). When we set $\psi(\cdot)$ as

$$\psi(s) = \begin{cases} 1 & \text{if } s \in [s_-, s_+] \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

we get Algorithm 2. We can see that Algorithm 2 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 1. *Suppose that graph G is generated by the relaxed graph generator with feature range constraint (Algorithm 3) whose $q(\cdot)$ is set as Equation* (5.7)*, 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. $f(\cdot)$.*

See appendix for the proof.

From Property 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 3. 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 1, if graph G is from the relaxed generator, we have

$$P(G \in \mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}) = \frac{1}{E_f[\psi(s)]} \int_{s_-}^{s_+} \psi(s)f(s)ds. \tag{5.11}$$

We can see that low value of $f(x)$ over $[s_-, s_+]$ reduces the probability in Equation (5.11). 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 Equation (5.11). When we choose $\psi(\cdot)$ as Equation (5.10), we have that the relaxed generator will then always on a graph within the range, for the probability in Equation (5.11) is always equal to 1.

Figure 7 illustrates two choices of $\psi(\cdot)$. $\psi(\cdot)$ is the p.d.f. of a normal distribution as shown in Equation (5.9). 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 $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$.

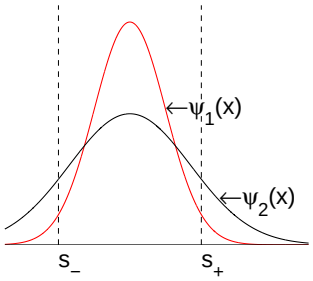


Figure 7: Choice of $\psi(\cdot)$

5.2 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 4 outlines the graph generator with feature distribution constraint.

Algorithm 4 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: $\quad G^t \leftarrow SingleSwitch(G^{t-1})$;
3: $\quad$ **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: $\quad\quad G^t \leftarrow G^{t-1}$
5: $\quad$ **end if**
6: **end for**
7: **return** G^k;

From Property 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 Equation (5.11), 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 Equation (5.11) we have

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

and then the p.d.f. of S value is equal to $g(x)$. Hence, by setting $q(\cdot)$ in Equation (5.6) as

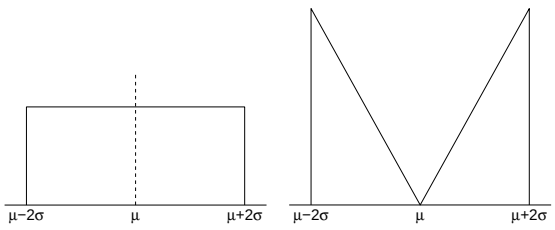
$$q(G) \leftarrow g[S(G)]/f[S(G)], \tag{5.12}$$

we can achieve the target distribution in Algorithm 4.

Empirical evaluation We apply Algorithm 4 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 2. The second distribution is a double-triangle-shaped distribution:

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

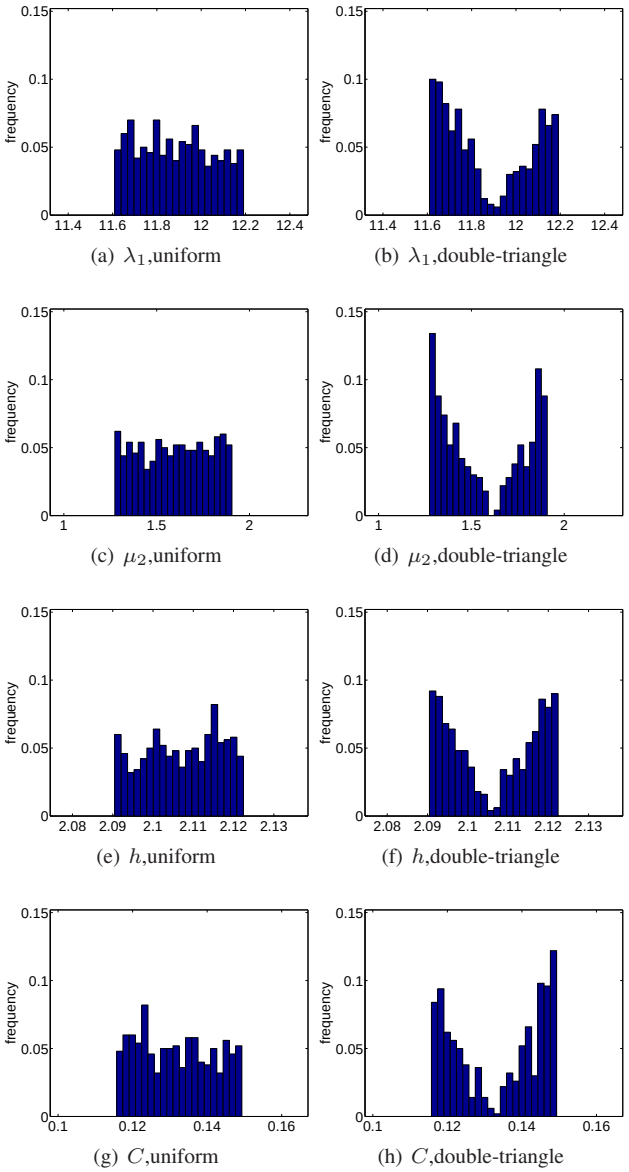
The shapes of the two target distributions are shown in Figure 8. Both of them are very different from the features' natural distributions $f(x)$. When applying Algorithm 4, we need to know the natural distribution of those features $f(x)$, and we use the kernel density estimator shown in Equation (3.1) to estimate $f(x)$ from the 3000 uniformly generated samples. Figure 9 shows the distributions of the four features of the 500 generated samples ($k = 6000$) using Algorithm 4. We can observe from Figure 9 that all the four features of generated samples match well the target distributions (shown in Figure 8).



(a) uniform (b) double-triangle

Figure 8: Target distributions $g(x)$

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.



(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 9: Feature distributions of generated graphs with feature distribution constraints shown in Figure 8 for polbooks

6 Related Work

6.1 State of the Art of Graph Generation Generally there are two approaches for the generation of graphs: matching approach [2, 7, 15, 18, 24, 29] and switching approach [22, 23].

The first matching based graph generator is the random graph model [7], which assumes every pair of nodes has identical, independent probability of being joined by an edge. To simulate important properties of real world graphs, various realistic graph generators using matching approach have been proposed in the past, such as the preferential at-

tachment [2, 15], the copying model, the small-world model [29], and the forest fire model. See [4] for a detailed survey. For example, the preferential attachment model can generate heavy-tailed degree distributions by attaching new nodes to high-degree old nodes. More recently, Leskovec and Faloutsos in [18,19] proposed the Kronecker model (based on Kronecker matrix multiplication) to generate graphs that obey multiple properties of real world graphs. Although the generated graphs satisfy power-law or some other properties (e.g., low diameters, similar spectrum), none of the above matching models can generate a uniform sample of possible graphs. Furthermore, there is no guarantee how well the generated synthetic graph mimics the structural features of a given real graph.

Switching approach applies a Markov chain to generate a synthetic graph by switching edges from the original graph. It has been shown in [22] that switching itself cannot generate uniformly sampled directed graphs. A "Go with the winners" algorithm based on a non-Markov chain Monte Carlo method was proposed to generate uniformly sampled directed graphs. However, previous switching based generators cannot guarantee the generated graph still preserves some useful features. As shown in this paper, many important topological features are lost in the generated graph.

Randomization techniques for testing the significance of discovered patterns have attracted much attention in data mining [10]. To conduct significance testing of network analysis results, it is essential to generate a group of synthetic graphs with features satisfying some distributions. In this paper, we presented algorithms based on Markov chain to generate synthetic graphs with feature range and distribution constraints.

6.2 State of the Art of Privacy Preservation in Social Networks Social network analysis has increasing interest in the database, data mining, and theory communities. The current state of the art is that there has been little work dedicated to privacy preserving social network analysis with the exception of some very recent work [1,3,5,12,13,20,30–33].

In [1], Backstrom et al. described a family of attacks such that an adversary can learn whether edges exist or not between specific targeted pairs of nodes from node-anonymized social networks. In this scenario, a social network owner releases the underlying graph structure after removing all node annotations. The goal of an attacker is to map the nodes in this anonymized graph to real work entities. The adversary can 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. Similarly in [12], Hay et al. further observed that the structure of the graph itself (e.g., the degree of the nodes or the degree of the node's neighbors) determines the extent to which an individual in the network can be distinguished. In [5], the authors considered settings where releasing the unlabeled graph is permitted and proposed an approach that masks the mapping from entities to nodes of the bipartite graph. The approach ensures that the underlying bipartite graph structure is not affected and identity privacy is preserved by perturbing the mapping from entities to nodes. However, this approach is not secure against subgraph attacks.

Link randomization or generalization has been shown a necessity in addition to node anonymization to preserve privacy in the released graph [1, 12]. Various anonymization schemes have been proposed to prevent the re-identification of individuals by the adversary with a priori knowledge of the social relationship of certain individuals. The idea is to modify a graph via a set of edge addition (or deletion) operations in order to construct a new k-anonymous graph, in which every node is indistinguishable with at least $k-1$ other nodes. In [20], Liu and Terzi investigated how to modify a graph via a set of edge addition (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. This property prevents the re-identification of individuals by the attackers with a-priori knowledge of the social relationships of certain people. In [33], Zhou and Pei anonymized the graph by generalizing node labels and inserting edges until each neighborhood is indistinguishable to at least $k-1$ others. In [3, 32], authors applied a structural anonymization approach called *edge generalization* that consists of collapsing clusters together with their component nodes' structure, rather than add or delete edges from the social network dataset. Although the above proposed approaches would preserve privacy, however, it is not clear how useful the anonymized graph is since many topological features may be lost.

Randomization methods [12, 30] based on link perturbation can be considered as one approach of generating a synthetic graph. In [30], we studied two natural edge-based graph perturbation strategies: *Rand Add/Del*(randomly adding one edge followed by deleting another edge and repeating this process for a fixed times.) and *Rand Switch*(randomly switching a pair of existing edges and repeating it for a fixed times) and showed that various structural properties can be significantly lost due to randomization. How to preserve utility (in terms of various structural features) and link privacy in the released graph is an important issue in privacy preserving social network analysis. In [30], a spectrum preserving graph randomization approach, which chooses switching edges via examining eigenvector values of corresponding nodes in order to better preserve network spectrum (i.e., eigenvalues of network matrices), was presented since the spectrum of a network is intimately connected to many important topological features. In

this paper, we presented an approach of directly generating graphs satisfying various feature constraints. The problem on how attackers may exploit the topological features of the released graph to breach link privacy was also recently studied in [31]. However, the attacking strategy in [31] was to exploit the relationship between existence of a link and the similarity measure values of node pairs in one released randomized graph. In this paper, the attacking model is based on the probability of existence of a link across all possible graphs in the graph space. It is interesting to compare these two attacking strategies and explore other potential attacking strategies on released perturbed social networks.

One loosely-related work is [16] that considered a particular threat in which an attacker subverts user accounts to gain information about local neighborhoods in the network and pieces them together in order to build a global information about the social graph. It considered the case where no underlying graph is released, and, in fact, the owner of the network would like to keep the entire structure of the graph hidden from any one individual. The goal of the attacker is, rather than to de-anonymize particular individuals from that graph, to compromise the link privacy of as many individuals as possible by determining the link structure of the graph based on the local neighborhood views of the graph from the perspective of several non-anonymous users.

7 Conclusion and Future Work

In this paper, we have presented a framework for generating synthetic graphs from the original one for two important applications, privacy preserving social network publishing and significance testing of network analysis results. We presented a simple switching based graph generator to generate graphs preserving features of a real graph. We then investigated the potential disclosure of sensitive links due to the preserved features. Our algorithm on graph generation with feature distribution constraints is based on the Metropolis-Hastings sampling, a standard method for generating a Markov chain with a target distribution. By configuring the transition probabilities in the switch process, we are able to generate graphs satisfying given feature constraints. This is of great importance for significance testing of network analysis results.

Our algorithms can be straightforwardly extended to generate graphs with constraints of multiple features. In our future work, we will explore the relationships of various features for larger real-world graphs and investigate how to efficiently generate graphs when a large number of constraints of multiple features are given as well as the impacts on link privacy. We are also interested in studying whether the attacker can improve their predication by exploring those non-random graphs in the graph space $\mathcal{G}_{\boldsymbol{d},\boldsymbol{S}}$, since the real-world graph usually contains less randomness than the synthetic graphs.

Acknowledgments

This work was supported in part by U.S. National Science Foundation IIS-0546027 and CNS-0831204.

References

[1] L. Backstrom, C. Dwork, and Jon 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, New York, NY, USA, 2007. ACM Press.

[2] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[3] A. Campan and T. Truta. A clustering approach for data and structural anonymity in social networks. In *PinKDD*, 2008.

[4] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2, 2006.

[5] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. *Proceedings of the VLDB*, 1(1):833–844, 2008.

[6] L. Costa, F. Rodrigues, G. Travieso, and P. Boas. Characterization of complex networks: A survey of measurements. *Advances In Physics*, 56:167, 2007.

[7] P. Erdos and A. Renyi. On random graphs i. *Publicationes Mathematicae*, 6:290–297, 1959.

[8] Z. Furedi and J. Komlos. The eigenvalues of random symmetric matrices. *Combinatorica*, 1 (3):233241, 1981.

[9] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman & Hall/CRC, 1996.

[10] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 167–176, 2006.

[11] W. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometria*, 57-1:97, 1970.

[12] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *University of Massachusetts Technical Report*, 07-19, 2007.

[13] M. Hay, G. Miklau, D. Jensen, D. Towsely, and P. Weis. Resisting structural re-identification in anonymized social networks. In *VLDB*, 2008.

[14] J. Kleinberg. Challenges in mining social network data: processes, privacy, and paradoxes. In *KDD*, pages 4–5, 2007.

[15] J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–17, 1999.

[16] A. Korolova, R. Motwani, S. Nabar, and Y. Xu. Link Privacy in Social Networks. Technical report, Technical Report.

[17] V. Latora and M. Marchiori. Efficient behavior of small-world networks. *Physics Review Letters*, 87, 2001.

[18] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. *Conference on Principles and Practice of Knowledge Discovery in Databases. Springer, Berlin, Germany*, 2005.

[19] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using Kronecker multiplication. *Proceedings of the 24th international conference on Machine learning*, pages 497–504, 2007.
[20] K. Liu and E Terzi. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD Conference*, Vancouver, Canada, 2008. ACM Press.
[21] S. Meyn and R. Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, London, 1993.
[22] R. Milo, N. Kashtan, S. Itzkovitz, M. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences, 2003.
[23] M. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89(20):208701, 2002.
[24] M. Newman, S. Strogatz, and D. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):26118, 2001.
[25] 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.
[26] 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.
[27] R. Taylor. Contrained switchings in graphs. *Combinatorial Mathematics VIII, Proceedings of the 8th Australian Conference on Combinatorial Mathematics*, pages 314–336, 1981.
[28] F. Viger and M. Latapy. Fast generation of random connected graphs with prescribed degrees. In *Proc. 11th International Computing and Combinatorics Conference*, 2005.
[29] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.
[30] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *Proc. of the 8th SIAM Conference on Data Mining*, April 2008.
[31] X. Ying and X. Wu. On link privacy in randomizing social networks. In *PAKDD*, 2009.
[32] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, pages 153–171, 2007.
[33] B. Zhou and J. Pei. Preserving Privacy in Social Networks Against Neighborhood Attacks. *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 506–515, 2008.

A Proof of Property 1

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

$$P[s(G) = s] = \frac{\psi(s)}{K} f(s)|\mathcal{G}_{\boldsymbol{d}}|, \tag{1.13}$$

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

$$P[a \leq S(G) \leq b] = \frac{|\mathcal{G}_{\boldsymbol{d}}|}{K} \int_a^b \psi(s) f(s) ds. \tag{1.14}$$

Let the range be the whole real axis, then

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

and we have $K = |\mathcal{G}_{\boldsymbol{d}}| E_f[\psi(s)]$. Combining this with Equation (1.14), we have the property proved. □

B Proof of Result 1

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

$$\begin{aligned} \frac{1}{N} \sum_{s=1}^{N} \|\widetilde{G}_s - G\|_F^2 &= \frac{1}{N} \left| \sum_{s=1}^{N} (\widetilde{G}_s - G)^{\cdot 2} \right| \\ &= \left| \frac{1}{N} \left(\sum_s \widetilde{G}_s^{\cdot 2} - 2G \otimes \sum_s \widetilde{G}_s + N G^{\cdot 2} \right) \right|, \end{aligned} \tag{2.15}$$

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 Equation (2.15), 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 (4.4).