Differentially Private Network Data Release via Structural Inference

Qian Xiao
NGS
National University of Singapore, Singapore
xiaoqian@nus.edu.sg

Rui Chen
Dept. of Computer Science
Hong Kong Baptist University, Hong Kong
ruichen@hkbu.edu.hk

Kian-Lee Tan
NGS and School of Computing
National University of Singapore, Singapore
tankl@comp.nus.edu.sg

ABSTRACT

Information networks, such as social media and email networks, often contain sensitive information. Releasing such network data could seriously jeopardize individual privacy. Therefore, we need to sanitize network data before the release. In this paper, we present a novel data sanitization solution that infers a network's structure in a differentially private manner. We observe that, by estimating the connection probabilities between vertices instead of considering the observed edges directly, the noise scale enforced by differential privacy can be greatly reduced. Our proposed method infers the network structure by using a statistical *hierarchical random graph* (HRG) model. The guarantee of differential privacy is achieved by sampling possible HRG structures in the model space via Markov chain Monte Carlo (MCMC). We theoretically prove that the sensitivity of such inference is only $O(\log n)$, where n is the number of vertices in a network. This bound implies less noise to be injected than those of existing works. We experimentally evaluate our approach on four real-life network datasets and show that our solution effectively preserves essential network structural properties like degree distribution, shortest path length distribution and influential nodes.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Network data; differential privacy; structural inference

1. INTRODUCTION

Information networks are invaluable assets for exploratory data analysis in a wide range of real-life applications. For instance, online social networks (e.g., Facebook and Twitter) are studied by sociologists to understand human social relationships; co-author networks are explored to analyze the degree and patterns of collaboration between researchers; voting and election networks are used to expose different views in the community; trust networks like Epinions are great resources for personalized recommendations. However, many of such networks contain highly sensitive personal information, such as social contacts, personal opinions and private communication records. To respect the privacy of individual participants in the networks, network data cannot be released for public access and scientific studies without proper "sanitization".

Previously, a great deal of work has investigated *anonymization* techniques [27, 16, 8, 13, 28, 5] to ensure network data privacy. However, it has been shown that anonymization is susceptible to several newly discovered privacy attacks and might lead to further privacy breaches. Recently, *differential privacy* [9] has been proposed to solve such vulnerability. In this paper, we study the problem of releasing network data under this emerging privacy standard. Given a network dataset, our goal is to release its sanitized differentially private version to hide each participant's connections to others while preserving essential structural information to support data analysis.

To ensure differential privacy, the standard technique is to add Laplace noise to query answers. However, network data can be very sensitive to relatively small changes in the network structure. Direct perturbation in the data domain (e.g., adding noise to a subgraph counting query in order to obscure the presence or absence of an edge) normally incurs excessive noise, which makes it impossible to conduct any effective data mining on the sanitized data. An alternative solution is to first project the data to other domains (e.g., the graph spectral domain [25], which is analogous to the classical frequency domain, or some parametric model space that describes the observed network, such as dK-2 series [21, 24]). While this idea is appealing, the resultant data utility of the existing works in this line is still undesirable for many graph mining algorithms. For example, Wang et al. [25] propose to perturb the eigenvalues and eigenvectors of the corresponding adjacency matrix. This approach requires to impose noise of magnitude proportional to $O(\sqrt{n})$, where n is the number of vertices in the input network, and therefore massive noise has to be injected in large real-life network datasets. As another example, the works [21, 24] consider to approximate the original network by the dK-series. To achieve ϵ-differential privacy, the global sensitivity of this scheme is $O(n)$ even for dK-2 series, which also demands excessive noise to be added.

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623642.

In this paper, we advocate a different approach that can offer better data utility. Broadly, we propose to encode a network's structural information in terms of *connection probabilities* between vertices, rather than the presence or absence of the observed edges. The fundamental advantage of adopting such a perspective is that we can capture the generally understandable and statistically meaningful properties of the network while "diluting" the impact of a single edge. In the context of differential privacy, this means that we can significantly lower the magnitude of noise added to mask the change of a single edge.

In essence, connection probabilities can be estimated by a set of edge-counting queries (i.e., a query that counts the number of edges between two given sets of vertices). Therefore, our problem can be converted to find a strategy that identifies a good set of edge-counting queries in order to truthfully represent a network's structure. This can be done in many possible ways. In particular, in this paper, we use a statistical *hierarchical random graph* (HRG) model [7] for this purpose. This HRG model carefully maps all participants of a network into a hierarchical structure (called a *dendrogram*) and records connection probabilities between any pair of vertices in the network. This allows us to draw a sample model from the model's space, which essentially consists of a set of good edge-counting queries. Moreover, the model itself is paired with a likelihood score, which makes it possible to observe the quality of released data.

Technically, we make the following contributions. Unlike existing studies, we propose to infer a network's structure via connection probabilities. We further identify that the HRG model can be used to encode a network in terms of a set of such connection probabilities. Generating a good HRG under differential privacy requires careful design. We do not directly perturb the best-fitting HRG of the input network (i.e., the HRG generated by the non-private algorithm), but rather, we infer the HRG by learning in the entire HRG model space and sampling an HRG by a Markov chain Monte Carlo (MCMC) method while satisfying differential privacy. Given a sampled HRG, we propose a carefully designed thresholding strategy coupled with the Erdős-Rényi model to calculate the noisy connection probabilities.

We adopt such a methodology for two reasons. First, relying on the best-fitting HRG itself will incur a high sensitivity. Changing even one edge in the network may result in a great number of changes in both the dendrogram's structure and the set of its associated connection probabilities. This is undesirable since it may alter many of the HRG's parameters in the worst case. In contrast, we design an MCMC method to iteratively learn a reasonably good HRG from the entire HRG space. By construction, with a single edge difference, only one probability in the HRG would be influenced. Second, it is non-trivial to sample a good HRG in our setting because it is computationally challenging to compute the scores of all possible HRGs even for a small network. It can be seen that there are a total of $(2n-3)!! \approx \sqrt{2}(2n)^{n-1}e^{-n}$ possible dendrograms for a network with n vertices. Hence, it is computationally infeasible to directly apply the exponential mechanism. We side-step this problem by using an MCMC method, which is in a similar spirit to the idea in [23]. However, our problem and challenges are quite different from those in [23]. Our goal is to publish the entire graph, *not* frequent subgraphs. A direct consequence is that we have to harness the large sensitivity in our problem, while it is always 1 in [23].

From the perspective of utility, we rigorously prove that the sensitivity of our proposed approach is $O(\log n)$ for fitting the dendrogram structure, which reaps the benefit of preserving good data utility in theory. We conduct extensive experiments on four real-life datasets to evaluate the effectiveness of our solution. We demonstrate that our approach significantly outperforms the state-of-the-art competitors [24, 25].

2. PRELIMINARIES

In this section, we briefly introduce the hierarchical random graph (HRG) model and differential privacy.

2.1 Hierarchical Random Graph

In this study, we follow the convention to model an input network dataset as a simple undirected graph $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. Let $A \in \{0,1\}^{n\times n}$ be the adjacency matrix that represents a graph G, where $A_{i,j} = 1$ if there is an edge between vertices i and j in G and $A_{i,j} = 0$, otherwise.

The HRG model is based on the intuition that the connection probability between two vertices depends on their degree of relatedness, which can be modelled mathematically via statistical inference. Specifically, the HRG model represents a graph G in terms of its hierarchical structure and a set of connection probabilities [6, 7]. The hierarchical structure of G in an HRG is captured by a *dendrogram* T, which is a rooted binary tree with n leaf nodes corresponding to the n vertices of G. Each internal node r of T is associated with a probability p_r. For any two vertices i, j of G, their probability of being connected $p_{i,j} = p_r$, where r is their lowest common ancestor in T. Formally, an HRG is defined by a pair $(T, \{p_r\})$.

Let L_r and R_r be the left and right subtrees of r respectively, and n_{Lr} and n_{Rr} be the numbers of leaves in L_r and R_r respectively. Let e_r be the number of edges in G whose endpoints are leaves of each of the two subtrees of r in T. The *likelihood* of an HRG for a given graph G measures how plausible this HRG is to represent G, which can be calculated, by Bayes' theorem, as follows:

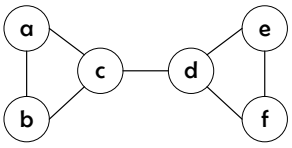
$$\mathcal{L}(T, \{p_r\}) = \prod_{r\in T} p_r^{e_r}(1-p_r)^{n_{Lr}n_{Rr}-e_r} \tag{1}$$

For a fixed dendrogram T, the maximum likelihood estimator of $p_r = \frac{e_r}{n_{Lr}\cdot n_{Rr}}$, which is the fraction of potential edges between the leaves of L_r and R_r that actually exist in G. In our scheme, we work with the logarithm of the likelihood (referred to as *log-likelihood* in the sequel):

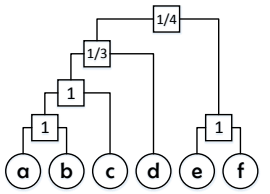
$$\log \mathcal{L}(T, \{p_r\}) = -\sum_{r\in T} n_{Lr}n_{Rr}h(p_r) \tag{2}$$

where $h(p_r) = -p_r \log p_r - (1-p_r)\log(1-p_r)$ is the Gibbs-Shannon entropy function. Essentially, a dendrogram paired with a higher likelihood is a better representation of the network's structure than those with lower likelihoods. We denote $\log \mathcal{L}(T, \{p_r\})$ by $\log \mathcal{L}(T)$ from now on when no confusion arises.

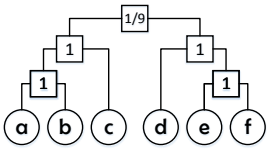
Example 1. *Figure 1(b) and (c) give an example of two possible dendrograms, T_1 and T_2, for an original graph in Figure 1(a). We first calculate the set of $\{p_r\}$ for each*



(a) Origin graph



(b) Dendrogram T_1



(c) Dendrogram T_2

Figure 1: An example of the HRG model in [7]

dendrogram. For example, to compute p_{r_2} associated with the root r_2 of T_2, we first obtain the two groups of leaf nodes in r_2's left and right subtrees, that is, $\{a, b, c\}$ and $\{d, e, f\}$. Since there is only one edge between these two sets of leaf nodes (i.e., the edge $\{c, d\}$), we have $e_{r_2} = 1$ and $p_{r_2} = 1/(3 * 3) = 1/9$. Similarly, we can calculate all $\{p_r\}$ and compute the likelihoods of the dendrogram T_1 and T_2. Specifically, $\mathcal{L}(T_1) = (1/3)(2/3)^2(1/4)^2(3/4)^6 \approx 0.00165$, and $\mathcal{L}(T_2) = (1/9)(8/9)^8 \approx 0.0433$. Since $\mathcal{L}(T_2)$ is much larger than $\mathcal{L}(T_1)$, T_2 is a more plausible hierarchy to describe the original graph. ■

2.2 Differential Privacy

Differential privacy [9] has emerged as a prevalent privacy model to quantify the notion of "indistinguishability" of *neighboring* databases. The privacy guarantee of differential privacy in the context of network data depends on the interpretation of *neighboring* graphs. In this paper, we define two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ to be *neighbors* if $V_1 = V_2$, $E_1 \subset E_2$ and $|E_1|+1 = |E_2|$. Formally, ϵ-differential privacy for network data is defined below.

Definition 1 (ϵ-Differential privacy). *A randomized algorithm $\mathcal{A}$ is ϵ-differentially private if for any two neighboring graphs G_1 and G_2, and for any output $O \in Range(\mathcal{A})$,*

$$\Pr[\mathcal{A}(G_1) \in O] \leq e^{\epsilon} \times \Pr[\mathcal{A}(G_2) \in O]$$ ■

Our definition of differential privacy is also known as *edge differential privacy* [12]. Intuitively, it hides the existence of *any* single edge from an adversary. The smaller ϵ is, the better the privacy protection is. Normally, ϵ is a small value (e.g., $\epsilon \leq 1$).

Differential privacy can be achieved by two standard mechanisms, the *Laplace mechanism* [9] and the *exponential mechanism* [18]. Both mechanisms are based on the concept of *global sensitivity* of a function f. For any two neighboring graphs G_1 and G_2, the global sensitivity of a function $f : G \to \mathbb{R}^d$ is defined as $\Delta f = \max_{G_1,G_2} \|f(G_1) - f(G_2)\|_1$.

The Laplace mechanism is mainly used for queries which return real values. It adds properly calibrated noise to the true answer to a query. More precisely, given a function f and the privacy parameter ϵ, the noise is drawn from a Laplace distribution with the probability density function $p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$, where $\lambda = \Delta f/\epsilon$.

Theorem 1 (Laplace mechanism [9]). *For any function $f : G \to \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(G) = f(G) + \langle \mathtt{Lap}_1(\frac{\Delta f}{\epsilon}), \ldots, \mathtt{Lap}_d(\frac{\Delta f}{\epsilon}) \rangle$$

gives ϵ-differential privacy, where $\mathtt{Lap}_i(\frac{\Delta f}{\epsilon})$ are i.i.d Laplace variables with scale parameter $\frac{\Delta f}{\epsilon}$. ■

The exponential mechanism is mainly used for functions whose outputs are not real numbers. Its general idea is to sample an output O from the output space $\mathcal{O}$ according to a utility function u. It assigns exponentially greater probabilities of being selected to outputs of higher scores so that the final output would be close to the optimum with respect to u. Let the global sensitivity of u be $\Delta u = \max_{O,G_1,G_2} |u(G_1, O) - u(G_2, O)|$.

Theorem 2 (Exponential mechanism [18]). *Given a utility function $u : (G \times \mathcal{O}) \to \mathbb{R}$ for a graph G, the mechanism $\mathcal{A}$ that samples an output O with probability proportional to $\exp(\frac{\epsilon \cdot u(G,O)}{2\Delta u})$ satisfies ϵ-differential privacy.* ■

3. STRUCTURAL INFERENCE UNDER DIFFERENTIAL PRIVACY

3.1 Overview

Before presenting the details, we first give an overview of our method. Our goal is to release a sanitized network $\widetilde{G}$ that matches the structural properties of the original network G as closely as possible while satisfying ϵ-differential privacy. Our general idea is to identify the hierarchical random graph (HRG) that best fits G and then generate $\widetilde{G}$ from the identified HRG.

Recall that an HRG consists of a dendrogram T and a set of associated probabilities $\{p_r\}$. This means that we need to not only identify a good fitting dendrogram but also calculate its associated probabilities. In this process, we face several major technical challenges: (1) How to find a good dendrogram from a factorial number of candidates while satisfying ϵ-differential privacy, and (2) how to calculate the probabilities that might be dominated by injected noise. We address the first challenge by designing a Markov chain Monte Carlo (MCMC) procedure, which samples a good dendrogram according to its likelihood. We cope with the second challenge by developing an effective thresholding strategy that is backed up by the Erdős-Rényi model. After generating a representative HRG for G, we generate $\widetilde{G}$ by placing edges according to $\{p_r\}$.

3.2 Algorithms

We now formally describe our solution (referred to as *HRG* in the sequel). Our solution is composed of three steps: (1) differentially privately sample a good dendrogram T_{sample} from the entire dendrogram space (Algorithm 1); (2) given the sampled dendrogram T_{sample}, compute the probabilities $\{p_r\}$ associated with T_{sample} (Algorithm 2); (3) generate the sanitized graph according to the identified HRG (Algorithm 3). We divide the total privacy parameter ϵ into 2 portions, ϵ_1 and ϵ_2, each being used in one of the first two

Algorithm 1: Differentially Private Dendrogram Fitting

Input : Input graph G, privacy parameter ϵ_1
Output: Sampled dendrogram T_{sample}

1 Initialize the Markov chain by choosing a random starting dendrogram T_0;
2 **for** each step i of the Markov chain **do**
3 Randomly pick an internal node r in T_{i-1};
4 Pick a neighboring dendrogram T' of T_{i-1} by randomly drawing a configuration of r's subtrees;
5 Accept the transition and set $T_i = T'$ with probability $\min(1, \frac{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T'))}{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T_{i-1}))})$;
6 **end**
7 `//when equilibrium is reached`
8 **return** the sampled dendrogram $T_{sample} = T_i$;

steps. Note that the third step does not require any privacy parameter.

Differentially Private Dendrogram Fitting. Since, for an input graph G with n vertices, each of its dendrograms T is associated with a log-likelihood $\log \mathcal{L}(T)$, which measures its goodness of representing G, a straightforward attempt to achieve differential privacy is to employ the exponential mechanism. Let the utility function be $u(T) = \log \mathcal{L}(T)$. The exponential mechanism samples T with probability proportional to $\frac{\exp(\frac{\epsilon_1}{2\Delta u} \cdot u(T))}{\sum_{T' \in \mathcal{T}} \exp(\frac{\epsilon_1}{2\Delta u} \cdot u(T'))}$, where $\mathcal{T}$ is the entire output space (i.e., the set of all possible dendrograms of G). Unfortunately, this simple idea is computationally infeasible because it requires to enumerate a total of $|\mathcal{T}| = (2n-3)!! \approx \sqrt{2}(2n)^{n-1}e^{-n}$ possible dendrograms. In our solution, we overcome the issue by designing an MCMC process, which simulates the exponential mechanism by a sequence of *local* transitions in $\mathcal{T}$. Our differentially private dendrogram fitting algorithm is summarized in Algorithm 1.

Algorithm 1 is based on the Metropolis algorithm [2]. It starts by choosing an arbitrary dendrogram $T_0 \in \mathcal{T}$ as the initial state of the Markov chain (Line 1). It then iteratively performs the following procedure (Lines 2-6): randomly propose a neighboring dendrogram T' of the dendrogram T_{i-1} in the previous iteration and update the current state in the following way:

$$T_i = \begin{cases} T' & \text{with probability } \alpha \\ T_{i-1} & \text{with probability } 1-\alpha \end{cases}$$

where the acceptance ratio $\alpha = \min(1, \frac{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T'))}{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T_{i-1}))})$ and Δu is the global sensitivity of the utility function u. We show how to calculate Δu in Section 4.2.

To draw a neighbor T' of T_{i-1} uniformly at random, we first randomly choose an internal node r in T_{i-1} (other than the root) and then permute the three subtrees associated with r to generate two alternative configurations of r's subtrees, as illustrated in Figure 2. One of these two configurations is chosen to be the neighboring candidate T'. Let the state space of this Markov chain be $\mathcal{T}$. It is easy to verify that the transitions based on this permutation scheme are both *reversible* and *ergodic* (i.e., any pair of dendrograms

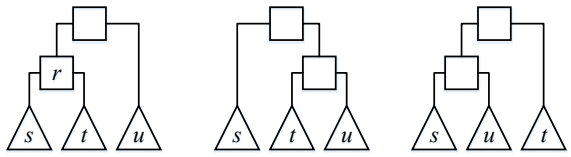


Figure 2: Three configurations of r's subtrees

can be connected by a finite sequence of such transitions). Hence, such an MCMC procedure has a unique stationary distribution after it converges to equilibrium. We run the above Markov chain until equilibrium is reached, which indicates that the desired distribution has already been reached. Therefore, the sampled dendrogram T_{sample} is indeed drawn from the stationary distribution (Line 8).

In practice, there are many approaches to diagnose MCMC convergence. Here we follow the method used in [6, 7]. Specifically, we use the heuristic of the average log-likelihood to judge whether the Markov chain has converged to the stationary distribution. We will elaborate more details of MCMC convergence time in Section 5.2. Additional discussion about the convergence and its mixing time can be found in [6, 7].

Noisy Probability Calculation. In the second step, we calculate the noisy probabilities associated with T_{sample}'s internal nodes. Recall that, for an internal node r, its associated probability $p_r = \frac{e_r}{n_{Lr} \cdot n_{Rr}}$ (see Section 2.1). It is easy to observe that the probabilities of the internal nodes rooted in smaller subtrees (i.e., in lower levels of T_{sample}) are generally more sensitive to Laplace noise injected. Indeed, according to our experiments, the direct application of the Laplace mechanism to these nodes' probabilities results in poor utility. To relieve such negative effects, we propose a carefully designed thresholding strategy coupled with the Erdős-Rényi model, which is presented in Algorithm 2.

The general idea is that if a probability p_r cannot be "reliably" estimated by applying the Laplace mechanism to $\frac{e_r}{n_{Lr} \cdot n_{Rr}}$, we employ the Erdős-Rényi model to approximate the probability. To measure the reliability of a noisy probability, we set up the sentinel λ_b. For an internal node r^* in T_{sample}, λ_b is set to $\frac{1}{\epsilon_2 \cdot (n_{Lr^*} \cdot n_{Rr^*})}$ (Line 1), which measures the noise scale of the potential noisy probability $\widetilde{p_{r^*}}$. If λ_b is relatively large with respect to a threshold value τ_1 (that is, the probability cannot be reliably calculated by the Laplace mechanism), we model the subgraph induced by all leaf nodes of the subtree rooted at r^* as an Erdős-Rényi random graph. With this model, the connection probability of *any* pair of vertices in this subgraph is $\frac{e_c(r^*)}{(n_{Lr^*}+n_{Rr^*})(n_{Lr^*}+n_{Rr^*}-1)/2}$, which is later perturbed by the Laplace mechanism (Line 5). Otherwise, we can expect that $\frac{e_r}{n_{Lr} \cdot n_{Rr}}$ still gives a good estimation after adding noise. Hence we directly generate the noisy probability as $\min\{1, \frac{e_{r^*} + \mathsf{Lap}(\frac{1}{\epsilon_2})}{n_{Lr^*} \cdot n_{Rr^*}}\}$ (Line 10) and perform the similar procedure on r^*'s children (Lines 11-14).

In Algorithm 2, we calculate the noisy probabilities in a top-down manner over T_{sample}. During this process, the approximated probabilities based on the Erdős-Rényi model also become less accurate due to added Laplace noise. Here, we would also like to guarantee the accuracy of the perturbed approximated probabilities. For this reason, we in-

Algorithm 2: `CalculateNoisyProb`$(G, T_{sample}, r^*, \epsilon_2)$

Input : Input graph G, sampled dendrogram T_{sample}, privacy parameter ϵ_2, internal node r^*
Output: A vector of noisy probabilities $\{\widetilde{p_r}\}$, where $r \in \{r^*$, all internal nodes below $r^*\}$

1 $\lambda_b = \frac{1}{\epsilon_2 \cdot (n_{Lr^*} \cdot n_{Rr^*})}$;
2 $\lambda_c = \frac{1}{\epsilon_2 \cdot ((n_{Lr^*}+n_{Rr^*})(n_{Lr^*}+n_{Rr^*}-1)/2)}$;
3 **if** $\lambda_b \geq \tau_1$ and $\lambda_c \geq \tau_2$ **then**
4 $e_c(r^*) \leftarrow$ number of edges in the subgraph induced by all leaf nodes of the subtree rooted at r^*;
5 $\widetilde{p} = \min\{1, \frac{e_c(r^*)+\texttt{Lap}(\frac{1}{\epsilon_2})}{(n_{Lr^*}+n_{Rr^*})(n_{Lr^*}+n_{Rr^*}-1)/2}\}$;
6 **for** each r in $\{r^*$, all internal nodes below $r^*\}$ **do**
7 $\widetilde{p_r} = \widetilde{p}$;
8 **end**
9 **else**
10 $\widetilde{p}_{r^*} = \min\{1, \frac{e_{r^*}+\texttt{Lap}(\frac{1}{\epsilon_2})}{n_{Lr^*} \cdot n_{Rr^*}}\}$;
11 $r_L \leftarrow r^*$'s left child;
12 $r_R \leftarrow r^*$'s right child;
13 `CalculateNoisyProb`$(G, T_{sample}, r_L, \epsilon_2)$;
14 `CalculateNoisyProb`$(G, T_{sample}, r_R, \epsilon_2)$;
15 **end**

troduce another sentinel λ_c (Line 2), which is compared with a threshold value τ_2 to indicate whether the noise scale of the approximated probabilities is acceptable. In summary, we employ the Erdős-Rényi model when (1) the probability cannot be accurately estimated by $\frac{e_{r^*}+\texttt{Lap}(\frac{1}{\epsilon_2})}{n_{Lr^*} \cdot n_{Rr^*}}$ (guarded by λ_b), *and* (2) injecting noise to $\frac{e_c(r^*)}{(n_{Lr^*}+n_{Rr^*})(n_{Lr^*}+n_{Rr^*}-1)/2}$ would not seriously affect its accuracy (guarded by λ_c). This explains our condition in Line 3. In this case, the probabilities of r^* and *all* internal nodes below r^* will be approximated by the Erdős-Rényi model (Lines 6-8). In our experiments, we observe that setting $\tau_1 = 0.05$ and $\tau_2 = 0.01$ gives good results over different real-life datasets. Note that the choices of these thresholds are *data-independent*: the tuning of τ_1 and τ_2 only relies on ϵ_2.

Sanitized Graph Generation. With the sampled dendrogram T_{sample} and the set of noisy probabilities $\{\widetilde{p_r}\}$, we generate the sanitized graph as follows (Algorithm 3). For each pair of vertices $i, j \in V$, we find their lowest common ancestor r in T_{sample} (Line 4), and then place an edge between them in $\widetilde{G}$ with probability $\widetilde{p_r}$ (Line 5).

4. PRIVACY ANALYSIS

In this section, we formally analyze the privacy guarantee of our algorithm *HRG*.

4.1 Privacy via Markov Chain Monte Carlo

We first show that the MCMC-based Algorithm 1 can satisfy differential privacy. Recall that the main purpose of applying the MCMC method is to draw a random sample from the desired distribution. Essentially, the standard exponential mechanism for achieving differential privacy is

Algorithm 3: Generate Sanitized Graph $\widetilde{G}$

Input : Input graph G, sampled dendrogram T_{sample}, privacy parameter ϵ_2
Output: Sanitized graph $\widetilde{G}$

1 $r_{root} \leftarrow$ root node of T_{sample};
2 `CalculateNoisyProb`$(G, T_{sample}, r_{root}, \epsilon_2)$;
3 **for** each pair of vertices $i, j \in V$ **do**
4 Find the lowest common ancestor r of i and j in T_{sample};
5 Place an edge in $\widetilde{G}$ between i and j with independent probability $\widetilde{p_r}$;
6 **end**
7 **return** sanitized graph $\widetilde{G}$;

also a method to sample an output $x \in \mathcal{X}$ in the target distribution with probability proportional to $\exp(\epsilon u(x)/2\Delta u)$, where $u(x)$ is the utility function and Δu is its sensitivity. Hence we see that, by matching the stationary distribution of MCMC with the target distribution required by the exponential mechanism, MCMC can be used to realize the exponential mechanism.

In our setting, we set the utility function $u(T)$ of a dendrogram T to be $\log \mathcal{L}(T)$, the *log-likelihood* of T, and the acceptance ratio of MCMC to be $\min(1, \frac{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T'))}{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T_{i-1}))})$. Therefore, when the Markov chain converges to the stationary distribution π, we indeed draw a sample T from π with the probability mass function:

$$Pr(T) = \frac{\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T))}{\sum_{T' \in \mathcal{T}} \exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T'))}.$$

This is equivalent to the exponential mechanism which outputs T with probability proportional to $\exp(\frac{\epsilon_1}{2\Delta u} \cdot \log \mathcal{L}(T))$. Therefore, we can conclude that Algorithm 1 satisfies ϵ_1-differential privacy.

We refer interested readers to [23] in which the idea of applying MCMC to achieve the exponential mechanism was first proposed for more discussion about how MCMC's stationary distribution perfectly matches the required distribution under the exponential mechanism.

4.2 Sensitivity Analysis

We now formally analyze the global sensitivity Δu. In this section, we will first derive how the utility function u (i.e., $\log \mathcal{L}(T)$) varies in neighboring databases. After that, we will formulate Δu and show that Δu monotonically increases as n grows. Lastly, we prove that Δu is $O(\log n)$.

In this work, we consider each possible output to be a dendrogram T in the output space $\mathcal{T}$. From the definition of global sensitivity, we have the following.

Definition 2 (Global sensitivity Δu).

$$\Delta u = \max_{T \in \mathcal{T}, G, G'} |\log \mathcal{L}(T, G') - \log \mathcal{L}(T, G)|$$

where G and G' are neighboring graphs. ■

Intuitively, Δu is the maximum change in the log-likelihood of any dendrogram in the output space if one edge is missing. It is easy to observe that missing one edge will influence

exactly one internal node's probability p_r in a dendrogram. Thus, we have:

LEMMA 1. $\Delta u = \max_{r \in \mathcal{T}} |(-n_{Lr} n_{Rr} h(p_r)) - (-n_{Lr} n_{Rr} h(p_r'))|$, *where* $p_r = \frac{e_r}{n_{Lr} n_{Rr}}$ *and* $p_r{}' = \frac{e_r - 1}{n_{Lr} n_{Rr}}$. ■

We now analyze how Δu varies as parameters change. Let $N = n_{Lr} \cdot n_{Rr}$. It is easy to see that there are two independent variables in Δu, the number of all possible connections N and the number of the observed edges e_r.

THEOREM 3. Δu *monotonically increases as* $n \to +\infty$, *and*

$$\Delta u = \log N_{max} + \log(1 + \frac{1}{N_{max} - 1})^{N_{max} - 1},$$

where $N_{\max} = \frac{n^2}{4}$ *when* n *is even and* $N_{\max} = \frac{n^2 - 1}{4}$ *when* n *is odd.* ■

PROOF. To analyze Δu, we first fix N. Let $f(e) = h(p) - h(p')$ and $\Delta u = \max |f(e)|$. Figure 3(a) plots the entropy value $h(p)$ as p varies. Since $f(e)$ has the format of discrete derivative of $h(p)$, we can analyze the monotonicity of $f(e)$ by computing the second order derivative of $h(p)$. We have

$$h''(p) = -\frac{1}{1-p} - \frac{1}{p}$$

It can be observed that $h''(p) < 0$ for all p. Hence $h(p)$ is a concave function and $h'(p)$ (or the acceleration) monotonically decreases. Therefore, $f(e)$ monotonically decreases.

Since $\Delta u = \max |f(e)|$, we just need to derive the extreme values of $f(e)$. Note that $f(e) > 0$ when p is in [0,0.5] and $f(e) < 0$ when p is in (0.5,1]. Hence, $\Delta u = \max(-\min(N \cdot f(e)), \max(N \cdot f(e)))$. Due to the symmetric property of $h(p)$, we can get $\max(f(e)) = -\min(f(e))$. With the monotonic property of $f(e)$, we can derive the value of Δu when $e = 1$ or $e = N_{max}$. Next we fix $e = 1$ and vary N. Let $\Delta u = \max_{N \in [1, N_{max}]} |f(N)|$, where

$$\begin{aligned} f(N) &= 1 \cdot \log \frac{1}{N} + (N-1) \cdot \log(1 - \frac{1}{N}) - 0 \\ &= -\log N + (N-1) \cdot \log(1 - \frac{1}{N}) \end{aligned}$$

The first order derivative of $f(N)$, $f'(N) = \log(1 - \frac{1}{N}) < 0$. Hence $f(N)$ is a decreasing function. Since $f(N) \leq 0$ for N in $[1, +\infty]$, we conclude that $\Delta u = -\min(f(N)) = -f(N_{\max})$. Hence,

$$\begin{aligned} \Delta u &= \log N_{max} - (N_{max} - 1) \cdot \log \frac{N_{max} - 1}{N_{max}} \\ &= \log N_{max} + (N_{max} - 1) \log(1 + \frac{1}{N_{max} - 1}) \\ &= \log N_{max} + \log(1 + \frac{1}{N_{max} - 1})^{N_{max} - 1} \end{aligned}$$

This completes the proof. □

Next we show that Δu is $O(\log n)$, where n is the number of vertices in the input network.

THEOREM 4. *The global sensitivity of a dendrogram's log-likelihood,* Δu, *is* $O(\log n)$. ■

PROOF. Based on Theorem 3, we first analyze the second term of Δu, that is, $\log(1 + \frac{1}{N_{max} - 1})^{N_{max} - 1}$. Let $y = (1 + \frac{1}{x})^x$. We have

$$\begin{aligned} \left(1 + \frac{1}{x}\right)^x &= 1 + \binom{x}{1}\frac{1}{x} + \binom{x}{2}\frac{1}{x^2} + \binom{x}{3}\frac{1}{x^3} + \cdots + \binom{x}{x}\frac{1}{x^x} \\ &= 1 + 1 + \sum_{k=2}^{n} \frac{1}{k!} \frac{x(x-1)\cdots(x-(k-1))}{x^k} \end{aligned}$$

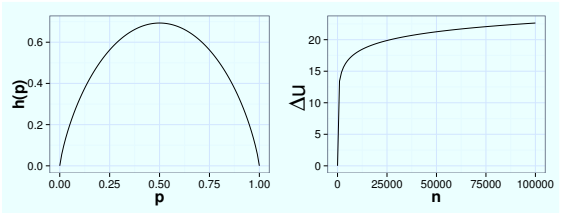
Since, for each $k \in \{2, 3, ..., x\}$,

$$\frac{1}{k!} \frac{x(x-1)\cdots(x-(k-1))}{x^k} = \prod_{j=1}^{k-1} \left(1 - \frac{j}{x}\right)$$

which increases with x, we learn that $y = (1 + \frac{1}{x})^x$ also increases with x. As $x \to \infty$, we have $\lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e$. Therefore we have:

$$\begin{aligned} \Delta u &= \log N_{max} + \log(1 + \frac{1}{N_{max} - 1})^{N_{max} - 1} \\ &< \log N_{\max} + \log e \leq \log \frac{n^2}{4} + 1 \\ &= O(\log n) \end{aligned}$$

This completes the proof. □



(a) (b)

Figure 3: Gibbs-Shannon entropy and plot of Δu

Figure 3(b) plots the value of Δu as n increases. We see that Δu increases slowly when n becomes larger. Thus we expect that applying the exponential mechanism in terms of MCMC in this setting would guarantee good data utility even for large-scale networks.

4.3 Privacy via Structural Inference

Finally, we prove that our solution *HRG* is ϵ-differentially private based on the *sequential composition* property.

THEOREM 5 (SEQUENTIAL COMPOSITION [17]). *Let each* $\mathcal{A}_i$ *provide* ϵ_i*-differential privacy. A sequence of* $\mathcal{A}_i(D)$ *over the database* D *provides* $\sum \epsilon_i$*-differential privacy.* ■

Taken with the above theorem, we can derive that our scheme ensures ϵ-differential privacy.

THEOREM 6. HRG *satisfies* ϵ*-differential privacy.* ■

PROOF. We use ϵ_1 in Algorithm 1 for sampling the dendrogram and ϵ_2 in Algorithm 2 for calculating the probabilities associated with the sampled dendrogram. From the analysis in above sections, we learn that Algorithm 1 is ϵ_1-differentially private. In Algorithm 2, we employ the Laplace mechanism to obtain the noisy answers to a set of *counting queries*. Since, by construction of a dendrogram, a single edge change will affect only one counting query by 1, Algorithm 2 is ϵ_2-differentially private. Since Algorithm 3 is based on the differentially private HRG generated by

Table 1: Network dataset statistics

Dataset	#Nodes	#Edges	Max Degree Pair
polblogs	1,224	16,715	(351, 277)
wiki-Vote	7,115	100,762	(1065, 773)
ca-HepPh	12,008	118,489	(491, 486)
ca-AstroPh	18,772	198,050	(504, 420)

Algorithm 1 and Algorithm 2, it does not consume any privacy budget. Hence, based on Theorem 5, we can conclude that our solution satisfies ϵ-differential privacy, where $\epsilon = \epsilon_1 + \epsilon_2$. □

5. EXPERIMENTAL EVALUATION

In this section, we experimentally study the equilibrium of our MCMC method and evaluate the utility of *HRG* over four real-life datasets, namely *polblogs*, *wiki-Vote*, *ca-HepPh* and *ca-AstroPh*[1]. *polblogs* is a network of hyperlinks between weblogs on US politics recorded in 2005. *wiki-Vote* is a social network containing Wikipedia voting information for adminship elections. An edge is created between two participants if one voted on or was voted by the other. *ca-HepPh* and *ca-AstroPh* are collaboration networks which cover scientific collaborations between authors submitted to High Energy Physics and Astro Physics categories, respectively. An edge is created if two authors co-authored a paper. The statistics of these datasets are given in Table 1. All datasets are pre-processed to be undirected without self-loops. All experiments were done on Intel Xeon E5607 servers with 2.27G CPU and 32GB RAM.

5.1 Experimental Settings

In our first set of experiments, we fix $\epsilon = 1.0$. Specifically, we assign $(\epsilon_1, \epsilon_2) = \{(0.1, 0.9), (0.5, 0.5), (0.9, 0.1)\}$ for sampling the dendrogram and computing noisy connection probabilities, respectively (see Figures 5-8). In the second set of experiments, we study the influence of different privacy parameters on data utility. Due to space constraint, we only report the results on *wiki-Vote* (Figures 9 and 10). We do observe similar trends on other datasets. In the figures, we denote our solution *HRG* with the legend *hrg-*ϵ_1*-e-*ϵ_2.

For comparison purposes, we implemented two state-of-the-art competitors, *spectral* [25] and *dk2* [24]. Since no systematic approach of choosing parameter values is provided in [25], we tune the parameters in *spectral* and report the best performance we obtain. More specifically, let k be the number of eigenvalues chosen in the scheme, ϵ_1 be the privacy budget for computing noisy eigenvalues and ϵ_2 for computing noisy eigenvectors. The literature [26] referred by Wang et al. in [25] suggests that k is usually in [2, 9]. Hence we vary k from 2 to 9 and report the best case. In the figures, *spectral* is denoted by the legend *spec-k-*ϵ_1*-*ϵ_2.

Due to the poor performance of *dk2* under ϵ-differential privacy, we compare with the scheme under a more relaxed privacy notion, that is, (ϵ, δ)-differential privacy. We follow the parameter settings in [24] and set $\delta = 0.01$. Unfortunately, even under (ϵ, δ)-differential privacy, we still need to use relatively large ϵ values (e.g., 200) to obtain comparable results. Moreover, the sensitivity in this case is data-dependent. It depends on the maximum degree pair in the networks (see Table 1). So we choose ϵ values proportional to the maximum degree pair in each network. The choice of parameters for *dk2* is denoted by the legend *dk2-*ϵ*-*δ.

From a privacy's perspective, *spectral* requires the number of edges in the input network to be known, whereas our scheme *HRG* and *dk2* do not require so. In addition, *dk2* is not able to remap the nodes to the observed network, so the experiments on influential node analysis is not applicable to *dk2*.

[1] *polblogs* is available at `http://www-personal.umich.edu/~mejn/netdata/`; *wiki-Vote*, *ca-HepPh* and *ca-AstroPh* are available at `http://snap.stanford.edu/data/index.html`.

5.2 Log-likelihood and MCMC Equilibrium

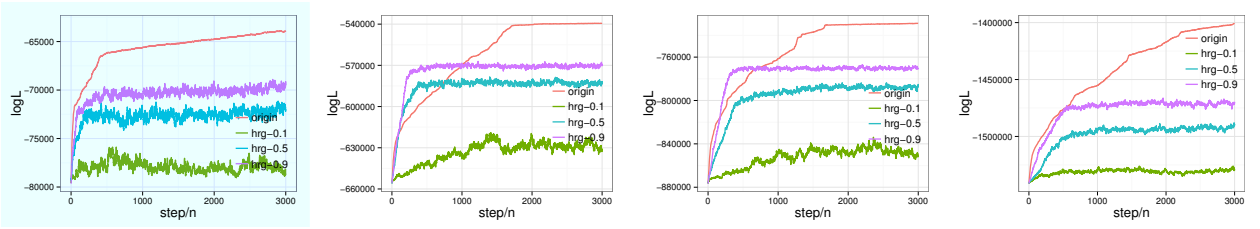
In practice, we diagnose MCMC's convergence by tracing the *log-likelihood*, $\log \mathcal{L}(T)$, of the sampled dendrograms. The diagnostic takes down consecutive non-overlapping windows of the Markov chain (each window consists of 65536 MCMC steps in our experiments) and compares the means of $\log \mathcal{L}(T)$ within these windows. We use the difference of the means to judge whether the means of $\log \mathcal{L}(T)$ within the windows have stabilized. In our experiments, we continuously examine whether the difference falls into the range $[-0.05n, 0.05n]$ to check the equilibrium state, where n is the number of nodes in the network.

In Figure 4, we plot the trace of $\log \mathcal{L}(T)$ as a function of the number of MCMC steps, normalized by n. We observe that the Markov chains mix well over all datasets (i.e., $\log \mathcal{L}(T)$ becomes stable soon after the initial state), indicating the convergence to the stationary distributions. Even though the mixing time can be exponential in the worst case [19], we observe that, in practice, the Markov chain in *HRG* usually can converge within $1000n$ steps on networks of around ten thousand nodes. Figure 4 also shows that the integration of differential privacy actually speeds up the movement of the Markov chains and makes them mix even faster. Roughly, the running time of n MCMC steps in our experiments is 0.18s for *polblogs*, 4.1s for *wiki-Vote*, 9.5s for *ca-HepPh* and 22.9s for *ca-AstroPh*. More details about the mixing time can be found in [6].

Figure 4 also shows the comparison of the sampled dendrograms' $\log \mathcal{L}(T)$ in different parameter settings, including that of the dendrogram sampled in the non-private manner. We can observe that, for networks with around ten thousand vertices, $\log \mathcal{L}(T)$ of the dendrogram sampled under a relatively small privacy parameter (e.g., $\epsilon_1 = 0.5$) is still comparable with that under a relatively large privacy parameter (e.g., $\epsilon_1 = 0.9$). Hence, we expect that even assigning a relatively small ϵ_1 for sampling the dendrogram will not significantly harm the data utility of the released network. To validate this, we further conduct experiments with smaller ϵ_1, such as 0.2 and 0.4. The performance shown in Figures 9 and 10 confirm that our scheme preserves reasonably good data utility even under a stringent privacy parameter (see Section 5.3 for the explanation of the utility metrics in Figures 9 and 10).

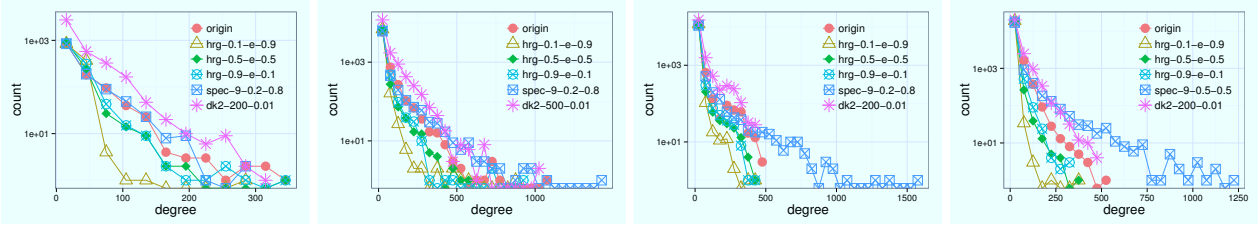
5.3 Utility Analysis

To show the utility of the released networks, we compare their degree distributions, shortest path length distributions and influential node ranking with those of the original networks. Due to the randomness of our algorithm, we examine the variance of its performance by running the algorithm ten times on each network for each parameter setting. We observe that the variance in all cases is small.



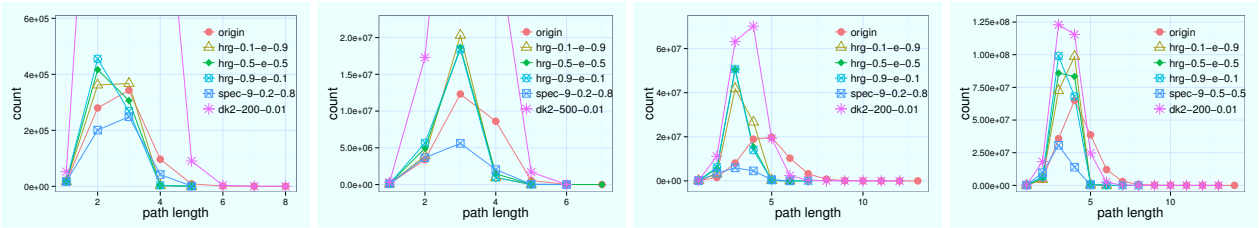
(a) *polblogs* (b) *wiki-Vote* (c) *ca-HepPh* (d) *ca-AstroPh*

Figure 4: Trace of log-likelihood as a function of the number of MCMC steps, normalized by n



(a) *polblogs* (b) *wiki-Vote* (c) *ca-HepPh* (d) *ca-AstroPh*

Figure 5: Degree distribution



(a) *polblogs* (b) *wiki-Vote* (c) *ca-HepPh* (d) *ca-AstroPh*

Figure 6: Shortest path length distribution

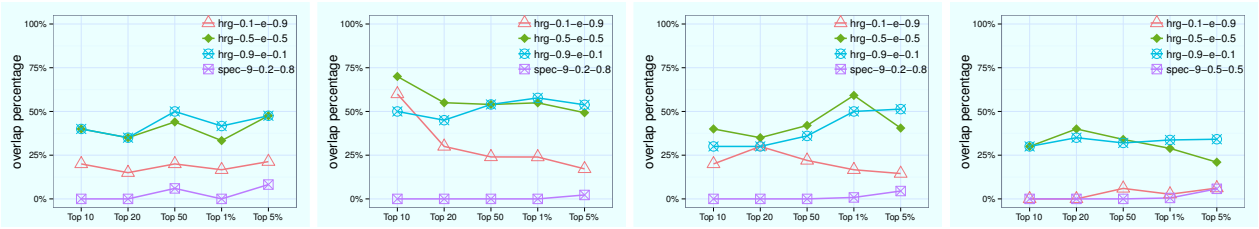
Degree Distribution. Figure 5 shows the degree distributions of the released data under different sanitization schemes, with y-axis in log-scale. It can be seen that, in all cases, *HRG* preserves well the right-skewness of the original networks, meaning that it preserves good distance scale between "hubs" (i.e., nodes having high degrees) and the majority of low-degree nodes.

Shortest Path Length Distribution. Figure 6 depicts the shortest path length distribution of each network. We observe that, in general, the released networks preserve the shapes of the distributions with respect to those of the original networks. However, we also observe the increase of paths of small lengths (e.g., 1-3). We believe this is due to the extra edges added to the low levels of the sampled dendrogram, which corresponds to the local structures in a network. But this does not have a big influence on the network's global structure.

Influential Node Analysis. Identifying the most influential nodes in social networks is a key problem in social network analysis. In our experiments, we consider the *eigenvector centrality* (EVC) score as the measure to rank the nodes in networks. EVC scores correspond to the values of the first leading eigenvector of the graph's adjacency matrix (the one with the greatest eigenvalue). Intuitively, EVC measures the nodes' influence by virtue of their positions in a network, that is, the sum of the geodesic distances from each node to all others. The eigenvector approach attempts to find the most central nodes (i.e., those with the smallest geodesic distance to others) in terms of the "global" or "overall" structure of the network. The first eigenvector usually captures the "global" aspects of distances among nodes, while the second and subsequent ones capture more specific and local structures.

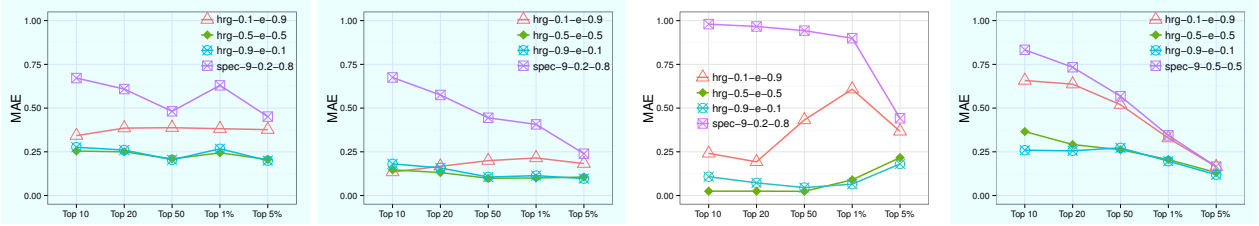
In our experiments, we first test the percentage of common nodes in top-k most influential nodes of the original graphs and those of the sanitized graphs. We examine top 10, 20, 50 influential nodes as well as top 1% and 5% nodes in the networks. The results are presented in Figure 7. We see that *HRG* guarantees a consistent 25%-75% overlap of common nodes in all the cases.

We then calculate the mean absolute error of the top-k most influential nodes' EVC scores. Let the set of top-k nodes in the original graph be α and that of the sanitized graph be β. To show that nodes in β have similar centralities to those in α, we use the *mean absolute error* (MAE) to compare the EVC scores in β with those in α. Formally, the MAE value is formulated by: $MAE(\alpha, \beta) = \frac{1}{k}\sum_{i=1}^{k}\left|EVC(v_\alpha^i) - EVC(v_\beta^i)\right|$, where v_α^i and v_β^i are the top-i nodes in α and β, respectively. In Figure 8 , we observe that the MAE of *HRG* with $\epsilon_1 = 0.5$ and 0.9 is reasonably low (e.g., less than 25% in most cases). The overlaps in top-k nodes and the low MAE together indicate that *HRG* well preserves the hub nodes, which represent the global structure of the sanitized graph.



(a) *polblogs* (b) *wiki-Vote* (c) *ca-HepPh* (d) *ca-AstroPh*

Figure 7: Overlaps of top-k vertices



(a) *polblogs* (b) *wiki-Vote* (c) *ca-HepPh* (d) *ca-AstroPh*

Figure 8: Mean absolute error of top-k vertices

6. RELATED WORK

Early works on *privacy-preserving network data publishing* [16, 27, 28, 5] mainly focus on developing anonymization techniques for specific types of privacy attacks. Most of them employ privacy models derived from k-anonymity [22] by assuming different types of adversarial knowledge. Unfortunately, all these anonymization techniques are vulnerable to attackers with stronger background knowledge than assumed, which has stimulated the use of differential privacy for more rigorous privacy guarantees.

The recent research on applying differential privacy to network data roughly falls into two directions. The first direction aims to release certain differentially private data mining results, such as degree distributions, subgraph counts and frequent graph patterns [12, 14, 11, 23]. However, our problem is substantially more challenging than publishing certain network statistics or data mining results. Our goal is to publish the entire graph, which incurs a much larger global sensitivity. Note that the sensitivity in the problem setting of [23] is only 1. In contrast, our key technical contribution is to achieve a smaller sensitivity in releasing a graph (i.e., $O(\log n)$ as opposed to $O(n)$ and $O(\sqrt{n})$ of our state-of-the-art competitors [21, 24, 25]). In addition, some latest studies [4, 1, 15] attempt to answer graph queries under a more stringent instantiation of differential privacy known as *node differential privacy* [12]. In spite of the significant progress, it is still difficult to develop practical solutions under node differential privacy in our problem setting.

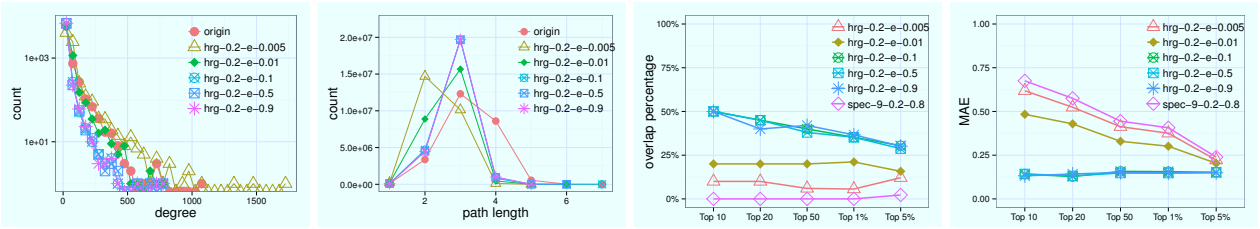
The second direction aims to publish a sanitized graph, which is also the objective of this paper. Most research in this direction [21, 20, 24] projects an input graph to dK-series and ensures differential privacy on dK-series statistics. These private statistics are then either fed into graph generators or used by MCMC to generate a fitting synthetic graph. While the general idea is appealing, the current techniques are still inadequate to provide desirable data utility for many graph mining tasks unless the privacy parameter is unreasonably large (e.g., $\epsilon \geq 100$), as demonstrated in Section 5. Wang et al. [25] propose to project a graph to the spectral domain and inject noise to the eigenvalues and eigenvectors. This approach successfully reduces the sensitivity to $O(\sqrt{n})$, which, unfortunately, is still not able to achieve good data utility. Comparing with all these studies based on the idea of projection, our approach takes an important step to achieve desirable data utility on real-life datasets in many practical settings. Gupta et al. [10] give an iterative database construction algorithm to generate synthetic graphs for answering cut queries. However, it requires an input graph to be dense, which is unlikely to be satisfiable on real-life network data and leaves finding an efficient algorithm as an open problem. Very recently, Chen et al. [3] propose a data-dependent solution by identifying and reconstructing the dense regions of a graph's adjacency matrix. Though it achieves reasonable utility on some datasets, its performance heavily relies on the fundamental assumption that most edge information must be captured by the dense regions of the adjacency matrix. In addition, this solution involves multiple ad-hoc parameters, which are difficult for a data publisher to tune.

7. CONCLUSION

In this paper, we address the privacy concerns in network data release by proposing a novel data sanitization method under differential privacy. Our solution is based on structural inference over the hierarchical random graph (HRG) model. Compared with the existing works, we theoretically prove that the sensitivity of our solution is much smaller, only logarithmic in the order of the network size(i.e., the number of vertices), implying a significant utility improvement. Extensive experiments on four real-life datasets confirm that our solution outperforms the state-of-the-art competitors.

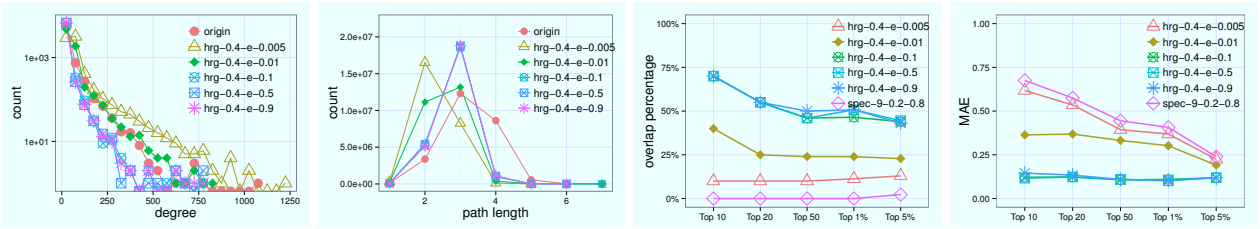
8. REFERENCES

[1] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, 2013.



(a) Degree distribution (b) Shortest path length (c) Overlaps of top-k vertices (d) Mean absolute error

Figure 9: *wiki-Vote* with *hrg*-0.2



(a) Degree distribution (b) Shortest path length (c) Overlaps of top-k vertices (d) Mean absolute error

Figure 10: *wiki-Vote* with *hrg*-0.4

[2] S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011.

[3] R. Chen, B. C. M. Fung, P. S. Yu, and B. C. Desai. Correlated network data publication via differential privacy. *VLDBJ*, in press.

[4] S. Chen and S. Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *SIGMOD*, 2013.

[5] J. Cheng, A. W. C. Fu, and J. Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *SIGMOD*, 2010.

[6] A. Clauset, C. Moore, and M. E. J. Newman. Structural inference of hierarchies in networks. In *ICML on Statistical Network Analysis*, 2007.

[7] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

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

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[10] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *TCC*, 2012.

[11] M. Hardt and A. Roth. Beating randomized response on incoherent matrices. In *STOC*, 2012.

[12] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, 2009.

[13] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *PVLDB*, 1(1):102–114, 2008.

[14] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *PVLDB*, 4(11):1146–1157, 2011.

[15] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *TCC*, 2013.

[16] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, 2008.

[17] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9), 2010.

[18] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.

[19] E. Mossel and E. Vigoda. Phylogenetic MCMC algorithms are misleading on mixtures of trees. *Science*, 309(5744):2207–2209, 2005.

[20] D. Proserpio, S. Goldberg, and F. McSherry. A workflow for differentially-private graph synthesis. In *WOSN*, 2012.

[21] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *IMC*, 2011.

[22] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, 13(6):1010–1027, 2001.

[23] E. Shen and T. Yu. Mining frequent graph patterns with differential privacy. In *SIGKDD*, 2013.

[24] Y. Wang and X. Wu. Preserving differential privacy in degree-correlation based graph generation. *TDP*, 6(2), 2013.

[25] Y. Wang, X. Wu, and L. Wu. Differential privacy preserving spectral graph analysis. In *PAKDD*, 2013.

[26] L. Wu, X. Ying, X. Wu, and Z.-H. Zhou. Line orthogonality in adjacency eigenspace with application to community partition. In *IJCAI*, 2011.

[27] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.

[28] L. Zou, L. Chen, and M. T. Ozsu. K-automorphism: a general framework for privacy preserving network publication. *PVLDB*, 2(1):946–957, 2009.