Network Structure Release under Differential Privacy

Hiep Nguyen, Abdessamad Imine, Michael Rusinowitch

Network Structure Release under Differential Privacy

Hiep H. Nguyen*, **Abdessamad Imine***, **Michaël Rusinowitch***

*LORIA/INRIA Nancy-Grand Est, France

E-mail: {huu-hiep.nguyen,michael.rusinowitch}@inria.fr,abdessamad.imine@loria.fr

Abstract. The problem of private publication of graph data has attracted a lot of attention recently. The prevalence of differential privacy makes the problem more promising. However, the problem is very challenging because of the huge output space of noisy graphs, up to $2^{n(n-1)/2}$. In addition, a large body of existing schemes on differentially private release of graphs are not consistent with increasing privacy budgets as well as do not clarify the upper bounds of privacy budgets. In this paper, we categorize the state-of-the-art in two main groups: *direct publication* schemes and *model-based publication* schemes. On the one hand, we explain why model-based publication schemes are not consistent and are suitable only in scarce regimes of privacy budget. On the other hand, we prove that with a privacy budget of $O(\ln n)$, there exist direct publication schemes capable of releasing noisy output graphs with edge edit distance of $O(1)$ against the true graph. We introduce the new linear scheme Top-m-Filter (TmF) and improve the existing technique EdgeFlip. Both of them exhibit consistent behaviour with increasing privacy budgets while the model-based publication schemes do not. As for better scalability, we also introduce HRG-FixedTree, a fast permutation sampling, to learn the Hierarchical Random Graph (HRG) model. Thorough comparative evaluation on a wide range of graphs provides a panorama of the state-of-the-art's performance as well as validates our proposed schemes.

Keywords. Network structure realease, Differential privacy, Upper bounds for privacy budget, Top-m-Filter

1 Introduction

As one of the most general forms of data representation, graph supports all aspects of the relational data mining process. With the emergence of increasingly complex networks [24], the research community requires large and reliable graph data to conduct in-depth studies. However, this requirement usually conflicts with privacy policies of data contributing entities. Naive approaches like removing user ids from a social graph are not effective, leaving users open to privacy risks, especially re-identification attacks [1, 14]. Therefore, many graph anonymization schemes have been proposed [35, 17, 36, 7, 31]. The main techniques used in those works are based on random edge manipulation or deterministic transformations to satisfy k-anonymity [30]. Another popular class of schemes relies on uncertainty semantics [4, 25].

In this paper, we address the problem of graph anonymization from the perspective of differential privacy. This privacy model offers a formal definition of privacy with a lot

of interesting properties: no computational/informational assumptions about attackers, data type-agnosticity, composability and so on [21]. By differential privacy, we want to ensure the existence of connections between users to be hidden in the released graph while retaining important structural information for graph analysis [29, 32, 33, 6, 34].

Differentially private algorithms relate the amount of noise to the sensitivity of computation. Lower sensitivity implies smaller added noise. Because edges in simple undirected graphs are usually assumed independent, standard Laplace mechanism is applicable (e.g. adding Laplace noise to each cell of the adjacency matrix). However, this approach may severely deteriorate the graph structure. Recent methods of graph release under differential privacy try to reduce the graph sensitivity in many ways. Schemes in [29, 32] use *dK-series* [19] to summarize the graph into a distribution of degree correlations. The global sensitivity of 1K-series (resp. 2K-series) is 4 (resp. $O(n)$). Lower sensitivity of $O(\sqrt{n})$ is proposed in [33] by graph spectral analysis. The most recent works [6, 34] even reduce the sensitivity of graph to $O(\ln n)$[1]. While *Density Explore Reconstruct* (DER)[6] employs a data-dependent quadtree to summarize the adjacency matrix into a counting tree, Xiao et al. [34] propose to use *Hierarchical Random Graph* (HRG) [8] to encode graph structural information in terms of edge probabilities. A common disadvantage of the state-of-the-art DER [6] and HRG-MCMC [34] is the scalability issue. Both of them incur quadratic complexity $O(n^2)$, limiting themselves to medium-sized graphs only.

A characteristic shared by the methods 1K-series, 2K-series [29, 32], Kronecker graph model [22], spectrum-based [33], DER [6] and HRG-MCMC [34] is that they use a certain summarization structure (model) to encode the key information of the true graph and then perturb this structure to satisfy ϵ-DP before regenerating noisy sample graphs for output. We observe that these model-based approaches are not *consistent* in the sense of privacy/utility trade-off. At the extreme of $\epsilon = 0$ (best privacy), the summarization structures are random and the utility is lowest. However, at the extreme $\epsilon = \infty$ (no privacy), we cannot get the true graph (best utility). This phenomenon may be explained by the loss of information in the summarization structures. Note that the term *consistency* in the current work is slightly different from [2, 15] where consistency means the noisy output must satisfy certain constraints, e.g. non-negative integrality [2] or ordered sequence [15].

To remedy the scalability and consistency problems, we propose *Top-m Filter* (TmF) algorithm, which runs in $O(m)$, linear in the number of edges and attains highest utility for large enough ϵ. By considering the adjacency matrix as a sparse dataset, TmF leverages the high-pass filtering idea in [9] to avoid the whole matrix manipulation. More importantly, via TmF, we provide a theoretical result stating that $O(\ln n)$ is an upper bound of privacy budget for graph release under differential privacy, i.e. at $\epsilon = O(\ln n)$ we get a noisy graph very close to the true graph (highest utility). This naturally rules out high-sensitivity schemes in [29, 32, 33] and makes 1K-series, DER and HRG-MCMC meaningful only in regimes of scarce privacy budgets (i.e. not exceeding $O(\ln n)$). After the publication of TmF [26], we found the scheme *EdgeFlip* by Mülle et al. [23] which uses *edge flipping* technique. By a deeper analysis, we prove that EdgeFlip also provides an upper bound $O(\ln n)$ for privacy budget and it makes the edit distance converge faster than TmF. However, EdgeFlip costs $O(n^2)$ and we will show that it could be slightly redesigned to run in linear time for $\epsilon \geq \ln(\frac{n(n-1)}{2m} - 1)$.

For a better comparison on large graphs, we introduce *HRG-FixedTree*, a fast permutation sampling, to learn the Hierarchical Random Graph (HRG) model. HRG-FixedTree runs in

[1] In this paper, $\ln n$ is the natural logarithm used to bound the privacy budget ϵ and $\log n$ is the base-2 logarithm for complexity analysis.

$O(m \log n)$ and may be of independent interest for the community detection problem [12].

In brief, our contributions are as follows

- We analyze the two key challenges of graph release under differential privacy: huge output space and consistency. We also justify the relaxation of ϵ to $\ln n$ using the concept of ρ-differential identifiability [16].
- We prove an upper bound of privacy budget ϵ that any differentially private scheme for graph release should not exceed. The upper bound is validated by our proposed linear scheme TmF and the existing scheme EdgeFlip. By deeper theoretical analysis, we prove the fast convergence of EdgeFlip and reveal its limitations. Both TmF and EdgeFlip exhibit consistent behavior for larger privacy budgets.
- We introduce HRG-FixedTree to reduce the runtime of HRG inference by several orders of magnitude, making it feasible to perform the inference over large graphs.
- We conduct a thorough evaluation on real graphs from small to medium and large sizes to see which method performs best for different regimes of ϵ.

Compared with the published conference paper [26], this extended version presents new materials about the categorization of schemes, the analysis of challenges, improved versions of EdgeFlip and HRG-based schemes. The evaluation includes a new graph *dblp*.

The rest of this paper is organized as follows. We discuss the related work in Section 2 with an emphasis on the categorization of ϵ-DP graph release and different regimes of ϵ. Section 3 reviews key concepts and mechanisms of differential privacy. It also analyzes key challenges of graph release under differential privacy. TmF scheme is described in Section 4. In Section 5, we present the deeper analysis and a partially linear algorithm for EdgeFlip. We review HRG model, the limitations of full space exploration and then propose a faster technique HRG-FixedTree in section 6. The comparison of schemes is presented in section 7. In Section 8, the competitors are thoroughly evaluated on a variety of graphs. Finally, Section 9 concludes our work.

2 Related Work

In principle, after releasing a graph satisfying ϵ-DP, we can do any mining operations on it, e.g. graph metrics querying, community detection. The research community, therefore, expresses a strong interest in the problem of graph release via differential privacy. Differentially private algorithms relate the amount of noise to the sensitivity of computation. Lower sensitivity implies smaller added noise. Because edges in simple undirected graphs are usually assumed independent, standard Laplace mechanism [10] is applicable (e.g. adding Laplace noise to each cell of the adjacency matrix). However, this approach may severely deteriorate graph structure.

Generally speaking, the problem of graph publication under differential privacy comprises two main techniques: *direct publication* and *model-based publication*. Direct publication means that the output graph is constructed by directly adding noise to each edge, followed by a post-processing step. TmF [26] and EdgeFlip [23] belong to this category. The other technique, model-based publication, relies on an intermediary structure to extract some crucial statistics. The noise is added to the statistics in a differentially private manner. Finally, sample output graphs are regenerated from these noisy statistics. This category includes 1K-series, 2K-series [29, 32], Kronecker graph model [22], graph spectral analysis [33], DER [6], HRG-MCMC [34] and ERGM (Exponential Random Graph Model) [18].

EdgeFlip [23] applies Randomized Response Technique (RRT) [11, Section 3.2] to all edges. Given a parameter $s \in (0, 1]$, each edge is flipped (1-to-0 or vice versa) with probability $s/2$. EdgeFlip was originally proposed the problem of community detection and the expected number of edges in the output graphs is $2m$. This is equivalent to the choice of $\epsilon = \ln(\frac{n(n-1)}{2m} - 3) \approx \ln n$. In addition, EdgeFlip incurs a quadratic runtime. We give a deeper analysis of EdgeFlip in Section 5.

1K-series and 2K-series [29, 32] use dK-series [19] to summarize the graph into a distribution of degree correlations. 2K-series was expected to give better noisy output graphs but the utility results [29, 32] are not good enough, not to say the huge values of ϵ used (up to thousands). The similar problem happened to graph spectral analysis [33]. Kronecker graph model [22] privately estimates the initiator matrix Θ using smooth sensitivity [27] and only satisfies (ϵ, δ)-DP. The state-of-the-art DER and HRG-MCMC have the scalability issue. Both of them incur quadratic complexity $O(n^2)$, limiting themselves to medium-sized graphs. We improve HRG-MCMC by proposing the near-linear time HRG-FixedTree in Section 6. ϵ-DP graph release is also mentioned in ERGM [18, Section 4.1] but not in detail because its main goal is to support parameter estimation of exponential random graphs. The high complexity of Bayesian estimation in ERGM also confines itself to graphs of hundreds of nodes. Further comparison between schemes is presented in Section 7.

3 Preliminaries

In this section, we review key concepts and mechanisms of differential privacy.

3.1 Differential Privacy

Essentially, ϵ-differential privacy (ϵ-DP) [10] is proposed to quantify the notion of *indistinguishability* of neighboring databases. In the context of graph release, two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are neighbors if $V_1 = V_2$, $E_1 \subset E_2$ and $|E_2| = |E_1| + 1$. Formal definition of ϵ-DP for graph data is as follows.

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

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

Laplace mechanism [10] and Exponential mechanism [21] are two standard techniques in differential privacy. The latter is a generalization of the former. Laplace mechanism is based on the concept of *global sensitivity* of a function f which is defined as $\Delta f = \max_{G_1, G_2} ||f(G_1) - f(G_2)||_1$ where the maximum is taken over all pairs of neighboring G_1, G_2. Given a function f and a privacy budget ϵ, the noise is drawn from a Laplace distribution $p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$ where $\lambda = \Delta f/\epsilon$.

Theorem 2. *(Laplace mechanism [10]) For any function $f : G \to \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(G) = f(G) + \langle Lap_1(\frac{\Delta f}{\epsilon}), ..., Lap_d(\frac{\Delta f}{\epsilon}) \rangle \tag{1}$$

satisfies ϵ-differential privacy, where $Lap_i(\frac{\Delta f}{\epsilon})$ are i.i.d Laplace variables with scale parameter $\frac{\Delta f}{\epsilon}$.

Geometric mechanism [13] is a discrete variant of Laplace mechanism with integral output range $\mathbb{Z}$ and random noise Δ generated from a two-sided geometric distribution $Geom(\alpha)$: $Pr[\Delta = \delta|\alpha] = \frac{1-\alpha}{1+\alpha}\alpha^{|\delta|}$. To satisfy ϵ-DP, we set $\alpha = \exp(-\epsilon)$. We use geometric mechanism in our LouvainDP scheme.

For non-numeric data, the exponential mechanism is a better choice. Its main idea is based on sampling an output O from the output space $\mathcal{O}$ using a score function u. This function assigns exponentially greater probabilities to outputs of higher scores. Let the global sensitivity of u be $\Delta u = \max_{O,G_1,G_2} |u(G_1, O) - u(G_2, O)|$.

Theorem 3. *(Exponential mechanism [20]) Given a score 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 $\exp(\frac{\epsilon.u(G,O)}{2\Delta u})$ satisfies ϵ-differential privacy.*

Composability is a nice property of differential privacy which is not satisfied by other privacy models such as k-anonymity. Specifically, parallel composition is a key ingredient in our algorithm TmF (Section 4).

Theorem 4. *(Sequential and parallel compositions [21]) Let each A_i provide ϵ_i-differential privacy. A sequence of $A_i(D)$ over the dataset D provides $\Sigma_{i=1}^{n}\epsilon_i$-differential privacy.*

Let each A_i provide ϵ_i-differential privacy. Let D_i be arbitrary disjoint subsets of the dataset D. The sequence of $A_i(D_i)$ provides $\max_{i=1}^{n} \epsilon_i$-differential privacy.

3.2 Challenges of Graph Release under Differential Privacy

In this section, we discuss two key challenges of the problem addressed in this paper: *huge output space* and *consistency*.

The first challenge is about the size of the output space. While the output is a scalar value for simple counting problems [10], the dimension of the output space becomes much larger (i.e. $O(n^2)$) for graph release because we publish a noisy graph, not a scalar metric. The definition of ϵ-DP requires that for any two neighboring graphs G_1, G_2 and any output graph $\tilde{G}$, the randomized algorithm $\mathcal{A}$ must satisfy $\frac{Pr[\mathcal{A}(G_1)=\tilde{G}]}{Pr[\mathcal{A}(G_2)=\tilde{G}]} \in [e^{-\epsilon}, e^{\epsilon}]$.

Note that $Pr[\mathcal{A}(G_1) = \tilde{G}] = 0$ iff $Pr[\mathcal{A}(G_2) = \tilde{G}] = 0$, i.e. the distributions of $\mathcal{A}(G_1)$ and $\mathcal{A}(G_2)$ have the same *support* over the output space. Intuitively, the smaller the output space, the higher the probability of each $\tilde{G}$ in the space, so the higher the utility because $\tilde{G}$ is more concentrated around G_1. However, the output space is usually super-exponential. If we set no constraints, the output space of $\tilde{G}$ is of size $2^{n(n-1)/2}$ because any edge can appear independently. If we require the (expected) number of edges in $\tilde{G}$ is m as in Top-m-Filter and EdgeFlip (Sections 4 and 5), the output space of $\tilde{G}$ reduces to $\binom{n(n-1)/2}{m}$. If we go further with the constraint of unchanged (expected) degree sequence as in 1K-series [32], the size of the output space of $\tilde{G}$ is approximated by the number of ways to rewire the node stubs [24], i.e. $(2m)!$. If we try to keep the expected 2K-series unchanged [28, 32], the output space of $\tilde{G}$ gets smaller than $(2m)!$ but at the price of much higher sensitivity $O(n)$.

Over the huge output space, the mechanism $\mathcal{A}$ is useful only if it gives higher probabilities to those $\tilde{G}$ "close" to G_1 in the sense of utility metrics. So it is very challenging to find the ϵ-DP mechanism $\mathcal{A}$ that can reduce the output space and at the same time produce a highly concentrated distribution of $\tilde{G}$ around G_1 (as in the case of Laplace noises added to scalar counting values [10]). The huge output space of $\tilde{G}$ therefore relaxes the stringent requirement of ϵ (usually set to 1.0 in the literature). That is why we consider $\epsilon = O(\ln n)$ in

the experiments. Note that this limit of ϵ is much lower than the value 100 in [28], 200-2,000 in [32] or 4,474 in [33].

Another important justification of $\epsilon = \ln n$ is the *edge re-identification risk* quantified by ρ-*differential identifiability* (ρ-DI) [16]. In differential privacy, ϵ limits how much one individual can affect the function f, not how much information is revealed about an individual. Using the semantics of possible worlds, Lee and Clifton show that any ϵ-DP mechanism satisfies $\frac{1}{1+(M-1)e^{-\epsilon}}$-DI where M is the number of possible worlds. In our case, a powerful attacker who knows all edges $E_{G_1} \setminus \{e\}$ tries to infer the existence of e in G_1. The number of possible worlds is $M = \frac{n(n-1)}{2} - m$, so we have $\rho = \frac{1}{1+(n(n-1)/2-m-1)e^{-\epsilon}}$. Substituting $\epsilon = \ln n$ into it, we get $\rho \approx \frac{2}{n}$ given the fact that $m = O(n)$ for sparse graphs. We believe that the factor $\rho = O(1/n)$ at $\epsilon = \ln n$ is acceptable for edge privacy, especially on million-scale graphs.

For the second challenge, consistency means that if we constantly increase the privacy budget ϵ, $\tilde{G}$ must get closer to G_1. Ideally, $\tilde{G} = G_1$ at a certain value of ϵ (a.k.a upper bounds). As we will see in the evaluation (Fig.12), only Top-m-Filter and EdgeFlip are consistent with ϵ. The consistency property allows data owners to have a wider range of choices for ϵ. For example, they can allow ϵ up to $\ln n$ or $1.5 \ln n$ for better utility when the privacy is not too stringent.

4 Top-m Filter

We introduce our linear time algorithm *Top-m Filter* (TmF) in this section. Its basic idea is to consider the adjacency matrix as a sparse dataset. Most of the real-world graphs expose sparsity, i.e. the average degree is of $O(\log n)$ where n is the number of nodes [24, Table 2]. TmF then uses an idea similar to High-pass Filter in [9] to avoid the materialization of the noisy adjacency matrix. Our algorithm processes at most $2m$ cells of the adjacency matrix, so it is linear in the number of edges. By devising TmF, we also reach an upper bound on privacy budget for graph publication in ϵ-DP setting.

4.1 Overview

In [9], Cormode et al. propose several summarization techniques for sparse data under differential privacy. Let M be a contingency table having the domain size m_0 and m_1 non-zero entries ($m_1 \ll m_0$ for sparse data). The conventional publication of a noisy table M' from M that satisfies ϵ-DP requires the addition of Laplace/geometric noise to all m_0 entries because the "differential" item can appear in any counting entry. The entries in M' could be filtered (e.g. removing negative ones) and/or sampled to get a noisy summary M''. This direct approach would be infeasible for huge domain sizes m_0. Techniques in [9] avoid materializing the vast noisy data by computing the summary M'' directly from M using filtering and sampling techniques. Because the counting values are integral, the Geometric mechanism 3.1 is preferred.

Compared to the High-pass filtering technique applied to contingency tables [9], our TmF method is different in two points. First, TmF aims at publication of sparse unweighted graphs with only 0-or-1 entries, not for any non-negative entries as in contingency tables. Second, the integral threshold used with the geometric mechanism in [9] makes the expected sum of noisy entries not equal to the sum of original entries. To keep the expected number of edges in published graphs unchanged, we use real-value thresholds, which lead to the application of the Laplace mechanism.

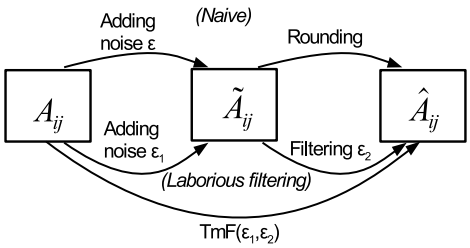


Figure 1: TmF algorithm

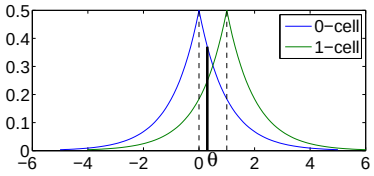


Figure 2: $0 < \theta < 1$

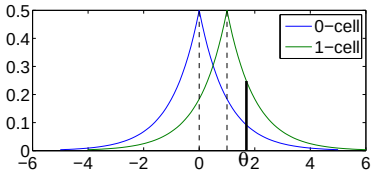


Figure 3: $1 \leq \theta$

Given the input graph G (represented by an adjacency matrix A) and a privacy budget ϵ, by the assumption of edge independence, the naive approach (*Naive*) adds Laplace noise to all cells in the upper-triangle of A, i.e. $\tilde{A}_{ij} = A_{ij} + Lap(1/\epsilon)$ for all $j > i \geq 1$. $\tilde{A}_{ij}$ is then post-processed by rounding $\hat{A}_{ij} = \arg\min_{x=0,1} |\tilde{A}_{ij} - x|$.

Instead of processing each cell independently as in Naive approach, our idea is to keep top-m noisy values $\tilde{A}_{ij}$ and reconstruct them to 1-cells. However, the number of edges m needs to be first obfuscated (note that in edge privacy model, only n is public [34]). We can achieve this by *Laborious filtering*, i.e. first computing the noisy number of edge $\tilde{m} = m + Lap(1/\epsilon_2)$, then adding Laplace noise $Lap(1/\epsilon_1)$ to all $\frac{n(n-1)}{2}$ cells and selecting top-$\tilde{m}$ noisy cells. This approach costs $O(n^2)$ in space and $O(n^2 \log n)$ in time because of the materialization of all cells. TmF avoids such problem by computing the threshold θ so that there are exactly $\tilde{m}$ noisy cells larger than θ. We call those cells *passing cells*. Fig. 1 depicts the processes of Naive, Laborious filtering and TmF.

The distributions for noisy values of 1-cells and 0-cells are shown in Figures 2 and 3. By setting a threshold θ, the probability of a 0-cell (resp. 1-cell) passing the filter is represented by the area under the blue curve (resp. the green curve) on the right of the vertical line at $x = \theta$. Clearly, the area for the 1-cells is always larger than the area for the 0-cells, i.e. the 1-cells have higher probability to pass the filter. By adjusting the threshold, we can control the number of 0-cells and 1-cells in the output graphs for given constraints. Such a constraint is the total number of passing cells.

We have two cases: $0 < \theta < 1$ and $1 \leq \theta$. The case $\theta \leq 0$ results in the number of passing cells is at least $\frac{n(n-1)}{4} \gg \tilde{m}$, and therefore omitted.

Case 1: $0 < \theta < 1$: the number of passing 1-cells is

$$n_1 = m \int_{\theta}^{+\infty} \frac{\epsilon_1}{2} \exp(-\epsilon_1 |x-1|) dx = \frac{m}{2}(2 - e^{-\epsilon_1(1-\theta)}) \tag{2}$$

The number of passing 0-cells is

$$n_0 = (\frac{n(n-1)}{2} - m) \int_{\theta}^{+\infty} \frac{\epsilon_1}{2} \exp(-\epsilon_1|x|)dx$$
$$= (\frac{n(n-1)}{2} - m)\frac{1}{2}e^{-\epsilon_1\theta}$$

By equating the sum of n_1 and n_0 to $\tilde{m}$, we can compute the value of θ. Because $\tilde{m} = m + Lap(1/\epsilon_2)$, we have $E[\tilde{m}] = m$. So to simplify the calculations, we set $n_1 + n_0 = m$. This leads to

$$\theta = \frac{1}{2\epsilon_1} \ln(\frac{n(n-1)}{2m} - 1) + \frac{1}{2} \tag{3}$$

Case 2: $1 \leq \theta$: Similarly, the number of passing 1-cells is

$$n_1 = \frac{m}{2}e^{-\epsilon_1(\theta-1)} \tag{4}$$

The number of passing 0-cells is

$$n_0 = (\frac{n(n-1)}{2} - m)\frac{1}{2}e^{-\epsilon_1\theta} \tag{5}$$

The value of θ is

$$\theta = \frac{1}{\epsilon_1} \ln(\frac{n(n-1)}{4m} + \frac{1}{2}(e^{\epsilon_1} - 1)) \tag{6}$$

To decide whether $\theta \geq 1$ or $0 \leq \theta \leq 1$, we compute the threshold ϵ_t of ϵ_1 at $\theta = 1$. For both cases,

$$\theta = 1 \leftrightarrow \epsilon_t = \ln(\frac{n(n-1)}{2m} - 1) \tag{7}$$

4.2 TmF Algorithm

Algorithm 1 shows steps of Top-m-Filter in which we replace all m by $\tilde{m}$. Line 3 computes the noisy number of edges $\tilde{m}$ using a budget ϵ_2 (set to 0.1 in our experiments). The threshold θ is decided in Lines 4-8. Lines 10-15 process 1-cells using the threshold θ. The remaining passing cells are sampled from 0-cells (Lines 17-22).

Theorem 5. *The complexity of TmF is* $O(m)$

Proof. Processing 1-cells (Lines 10-18) runs in $O(m)$. The maximum value of n_0 (Line 20) is $\tilde{m}$ ($= m$ in expectation). For each 0-cell to be processed, the rejection sampling (Lines 22-25) succeeds with probability at least $1 - \frac{2m}{n(n-1)} = 1 - O(1/n)$. So in summary, the total complexity of TmF is $O(m)$. □

Theorem 5 makes sense if we consider the complexity $O(n^2)$ of the state-of-the-art DER [6] and HRG-MCMC [34].

Algorithm 1 Top-m Filter

Require: input graph $G = (V, E)$, privacy parameters ϵ_1, ϵ_2
Ensure: sanitized graph $\tilde{G}$

$\tilde{G} \leftarrow \varnothing$
// compute $\tilde{m}$ and θ
$\tilde{m} = m + Lap(1/\epsilon_2)$
$\epsilon_t = \ln(\frac{n(n-1)}{2\tilde{m}} - 1)$
if $\epsilon_1 < \epsilon_t$ **then**
    $\theta = \frac{1}{2\epsilon_1} \ln(\frac{n(n-1)}{2\tilde{m}} - 1)$
else
    $\theta = \frac{1}{\epsilon_1} \ln(\frac{n(n-1)}{4\tilde{m}} + \frac{1}{2}(e^{\epsilon_1} - 1))$
end if
// process 1-cells
$n_1 = 0$
for $A_{ij} = 1$ **do**
    compute $\tilde{A}_{ij} = A_{ij} + Lap(1/\epsilon_1)$
    if $\tilde{A}_{ij} > \theta$ **then**
        add edge (i, j) to $\tilde{G}$
        n_1 ++
    end if
end for
// process 0-cells
$n_0 = \tilde{m} - n_1$
while $n_0 > 0$ **do**
    pick an edge $(i, j) \notin E$ uniformly at random
    if $\tilde{G}$ does not contain (i, j) **then**
        add edge (i, j) to $\tilde{G}$
        n_0- -
    end if
end while
return $\tilde{G}$

4.3 Privacy Analysis

In this section, we show that TmF satisfies ϵ-DP where $\epsilon = \epsilon_1 + \epsilon_2$. Our TmF consists of two steps. It is easy to verify that the sensitivity of m is 1. The first step of computing $\tilde{m}$ satisfies ϵ_2-DP. The second step of processing 1-cells and 0-cells is equivalent to independently adding noise $Lap(1/\epsilon_1)$ to each cell and letting them go through a high-pass filter with threshold θ. The sensitivity of each cell is also 1. By the assumption of edge independence, parallel composition (Theorem 4) is applicable at cell level. So the second step satisfies ϵ_1-DP. By sequential composition (Theorem 4), TmF satisfies $(\epsilon_1 + \epsilon_2)$-DP as stated in the following theorem.

Theorem 6. *TmF satisfies ϵ-DP where $\epsilon = \epsilon_1 + \epsilon_2$.*

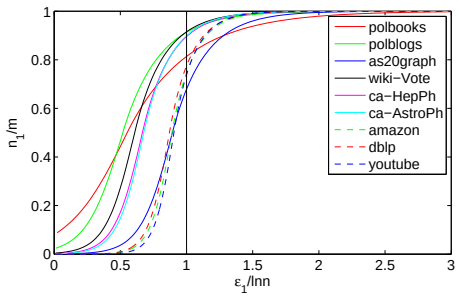


Figure 4: n_1/m as a function of $\epsilon_1/\ln n$

4.4 An Upper Bound for Privacy Budget

Now we proceed to the more important result: TmF could reduce the edit distance between $\tilde{G}$ and G to $O(1)$ at $\epsilon_1 = O(\ln n)$. The edit distance between two graphs having the same node set is defined as

$$D(G, \tilde{G}) = \frac{1}{2}(|E_G \setminus E_{\tilde{G}}| + |E_{\tilde{G}} \setminus E_G|) \tag{8}$$

By the analysis in Section 4.1, the expected number of passing 1-cells is n_1, so the expected edit distance $D(G, \tilde{G}) = m - n_1$. At $\theta = 1$, we have $n_1 = \frac{m}{2} = D(G, \tilde{G})$ and $\epsilon_1 = \epsilon_t = \ln(\frac{n(n-1)}{2\tilde{m}} - 1)$. The cases of small edit distance therefore correspond to the case $0 < \theta < 1$. Setting $D(G, \tilde{G}) = \gamma m$, $\gamma \in [\frac{1}{m}, 1]$, we need to find the value of ϵ_1.

$$\begin{aligned}
& D(G, \tilde{G}) = \gamma m \\
\leftrightarrow \quad & m - \frac{m}{2}(2 - e^{-\epsilon_1(1-\theta)}) = \gamma m \\
\leftrightarrow \quad & e^{-\epsilon_1(1-\theta)} = 2\gamma \\
\leftrightarrow \quad & \theta = 1 + \frac{1}{\epsilon_1}\ln(2\gamma) \\
\leftrightarrow \quad & \frac{1}{2\epsilon_1}\ln(\frac{n(n-1)}{2m} - 1) + \frac{1}{2} = 1 + \frac{1}{\epsilon_1}\ln(2\gamma) \ \text{(from (3))} \\
\leftrightarrow \quad & \epsilon_1 = \ln(\frac{\frac{n(n-1)}{2m} - 1}{4\gamma^2})
\end{aligned}$$

Because real-world graphs are usually sparse, $m = O(n)$, we reach $\epsilon_1 = O(\ln n)$. Specifically, $\epsilon_1 \approx 3\ln n$, $\epsilon_1 \approx 2\ln n$, $\epsilon_1 \approx \ln n$ at $\gamma = \frac{1}{m}$, $\gamma = \frac{1}{\sqrt{m}}$ and $\gamma = \frac{0.5}{O(\sqrt{\bar{d}})}$ respectively ($\bar{d} = \frac{2m}{n}$ is the average degree). We come up with the following theorem.

Theorem 7. *TmF can make the edit distance $D(G, \tilde{G}) = O(1)$ at $\epsilon_1 \approx 3\ln n$.*

Fig. 4 shows the normalized number of passing 1-cells n_1/m as a function of $\epsilon_1/\ln n$ over nine graphs (cf. Table 2). As we can see, at $\epsilon_1 = \ln n$ (the solid vertical line), 65-90% of edges in G are kept in $\tilde{G}$.

This result naturally points out the waste of privacy budget in [29, 32] and [33] where $\epsilon = O(\sqrt{n})$ or $\epsilon = O(n)$. Interestingly, in HRG-MCMC scheme [34], the sensitivity $\Delta u \approx 2\ln n$ which means that the non-private HRG-MCMC with $\epsilon = 2\Delta u \approx 4\ln n$ costs a budget even higher than $\epsilon \approx 3\ln n$ in our non-private TmF. We further analyze HRG-MCMC in Section 6.

5 EdgeFlip: Differential Privacy via Edge Flipping

We present a deeper analysis of EdgeFlip scheme in this section and a partially linear implementation of EdgeFlip using an idea similar to TmF.

5.1 A Tighter Upper Bound for Privacy Budget

EdgeFlip scheme [23] is also a direct publication scheme as TmF. Its basic idea is also based on the nature of edge differential privacy, i.e. all edges are independent. Given the privacy parameter $s \in (0, 1]$, any edge is flipped (from one to zero or vice-versa) with probability $s/2$ and preserves its value with probability $1 - s/2$. The probability ratio that two neighboring graphs are perturbed to the same graph can be expressed as $\frac{1-s/2}{s/2} = 2/s - 1$. To satisfied ϵ-DP, we must have $2/s - 1 \leq e^{\epsilon}$ or equivalently $\epsilon \geq \ln(2/s - 1)$ as stated in the following theorem.

Theorem 8. *(EdgeFlip privacy [23]) EdgeFlip guarantees 1-edge differential privacy for* $\epsilon \geq \ln(2/s - 1)$.

The expected number of edges in $\tilde{G}$ is as follows

Theorem 9. *(Expected number of edges [23])* $E[|V'|] = (1 - s)m + \frac{n(n-1)}{4}s$

By considering all edges in G (there are $\frac{n(n-1)}{2}$ such edges), EdgeFlip incurs a quadratic complexity and needs to be redesigned for large graphs. First, we compute the edit distance $D(G, \tilde{G}) = \frac{1}{2}(|E_G \setminus E_{\tilde{G}}| + |E_{\tilde{G}} \setminus E_G|)$ where

$$|E_G \setminus E_{\tilde{G}}| = |m - (1 - \frac{s}{2})m| = \frac{sm}{2} \tag{9}$$

$$|E_{\tilde{G}} \setminus E_G| = \left(\frac{n(n-1)}{2} - m\right)\frac{s}{2} \tag{10}$$

So, we get $D(G, \tilde{G}) = \frac{n(n-1)s}{8}$. From Theorem 8, $s = \frac{2}{e^{\epsilon}+1}$. Therefore, the relation between the edit distance and the privacy budget in EdgeFlip is as follows

$$D(G, \tilde{G}) = \frac{n(n-1)}{4(e^{\epsilon} + 1)} \tag{11}$$

By solving similar equations as in TmF, we come up with the following theorem

Theorem 10. *(EdgeFlip edit distance) EdgeFlip can reduce the edit distance to 1 at* $\epsilon = \ln\left(\frac{n(n-1)}{4} - 1\right) \approx 2\ln n$, $\sqrt{m}$ *at* $\epsilon = \ln\left(\frac{n(n-1)}{4\sqrt{m}} - 1\right) \approx 1.5\ln n$ *and* $m/2$ *at* $\epsilon = \ln(\frac{n(n-1)}{2m} - 1)$

The edit distance in EdgeFlip decreases faster than that of TmF for $\epsilon \in \left[\ln(\frac{n(n-1)}{2m} - 1), +\infty\right)$. Interestingly, $D(G, \tilde{G})$ is equal to $m/2$ at $\epsilon = \ln(\frac{n(n-1)}{2m} - 1)$ in both of them. However, when ϵ gets smaller, e.g. at $\epsilon \approx 0.5\ln n$, $D(G, \tilde{G})$ will be of $O(n^{1.5})$. This means EdgeFlip costs a super linear space to store the edges of $\tilde{G}$, not feasible on million-scale graphs. In contrast, TmF always outputs a noisy $\tilde{G}$ having the expected number of edges of m.

Algorithm 2 Partially linear-time EdgeFlip(G, s, ϵ_2)

Require: undirected graph $G = (V, E)$, privacy parameter s
Ensure: anonymized graph $\tilde{G}$
1: $\tilde{G} \leftarrow \varnothing$, $\epsilon_2 = 0.1$
2: $\tilde{m} = m + Lap(1/\epsilon_2)$
3: $\epsilon_t = \ln(\frac{n(n-1)}{2\tilde{m}} - 1)$
4: $\epsilon = \ln(\frac{2}{s} - 1) - \epsilon_2$
5: $\tilde{s} = \frac{2}{e^{\epsilon}+1}$
6: **if** $\epsilon + \epsilon_2 \leq \epsilon_t$ **then return** EdgeFlip($G, \tilde{s}$) [23]
7: **end if**
8: // process 1-cells
9: **for** $A_{ij} = 1$ **do**
10: add edge (i, j) to $\tilde{G}$ with prob. $1 - \tilde{s}/2$
11: **end for**
12: // process 0-cells
13: $n_0 = (\frac{n(n-1)}{2} - \tilde{m})\frac{\tilde{s}}{2}$
14: **while** $n_0 > 0$ **do**
15: pick an edge $(i, j) \notin E_G$ uniformly at random
16: **if** $\tilde{G}$ does not contain (i, j) **then**
17: add edge (i, j) to $\tilde{G}$
18: n_0- -
19: **end if**
20: **end while**
21: **return** $\tilde{G}'$

5.2 A Partially Linear Implementation

To make EdgeFlip runnable on large graphs, we propose a partially linear-time version as in Algorithm 2 where we process 1-cells and 0-cells separately. Note that Algorithm 2 is linear-time only if $s \leq \frac{4m}{n(n-1)}$ (see Lines 3-6) when the expected number of edges of $\tilde{G}$ is

$$E[|V'|] = (1-s)m + \frac{n(n-1)}{4}s = m + (\frac{n(n-1)}{4} - m)s$$
$$\leq m + (\frac{n(n-1)}{4} - m)\frac{4m}{n(n-1)} = 2m - \frac{4m^2}{n(n-1)} < 2m$$

We check this condition privately (Line 6) using a small ϵ_2 (set to 0.1 in the experiments) after computing $\tilde{m}$ and ϵ_t. The parameter s is replaced by $\tilde{s}$. For 1-cells, Lines 9 and 10 cost $O(m)$. Because $E[|V'|] \leq 2m$ for $s \leq \frac{4m}{n(n-1)}$, the number of 0-cells that are flipped to 1-cells $n_0 < E[|V'|]$, so EdgeFlip runs in linear time when $s \leq \frac{4m}{n(n-1)}$ and in quadratic time otherwise. This is an improvement over the original EdgeFlip [23].

The interested readers may think about the application of filtering technique of TmF to EdgeFlip for $s \geq \frac{4m}{n(n-1)}$. This idea, however, turns out to be infeasible. In TmF (a shortcut method of Laborious Filtering), all cells of the adjacency matrix are added a *continuous* Laplace noise, so the top-m noisy cells are easily computed. EdgeFlip, on the contrary, by using the flipping technique, outputs *discret* values 0 or 1. For $s \geq \frac{4m}{n(n-1)}$, the number of 1-cells is at least $2m$, so we cannot filter the top-m 1-cells. Figures 5 and 6 visualize the dif-

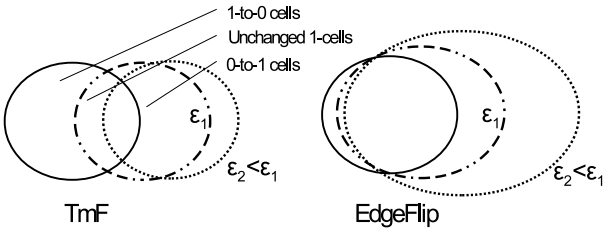


Figure 5: TmF vs. EdgeFlip

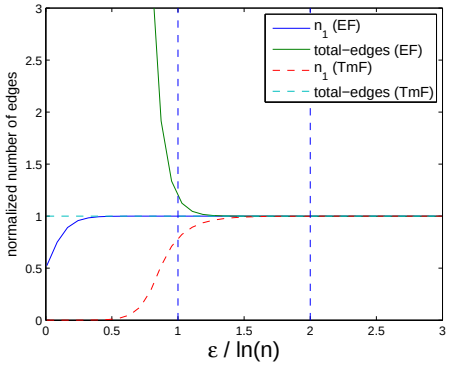


Figure 6: The number of passing 1-cells and the total of edges in EdgeFlip and TmF (on *amazon* graph)

ference between TmF and EdgeFlip. In TmF, the expected number of edges in $\tilde{G}$ is always m while in EdgeFlip, this number increases with s (i.e. when ϵ decreases), making the algorithm infeasible for small values of ϵ. This is the EdgeFlip's trade-off between the smaller upper bounds than TmF (Theorem 10) and the infeasibility at $s \geq \frac{4m}{n(n-1)}$ (corresponding to $\epsilon \leq \ln(\frac{n(n-1)}{2m} - 1)$).

6 HRG-based Schemes for Large Graphs

In this section, we present limitations of HRG-MCMC [34] (Section 6.1) and propose *HRG-FixedTree* (Section 6.2) for large graphs. HRG-FixedTree satisfies ϵ-DP and reduces the complexity to $O(m \log n)$. A fast sampling technique from dendrograms is also introduced (Section 6.3).

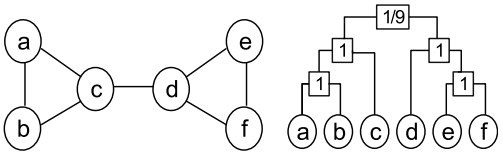


Figure 7: (left) graph G (right) a dendrogram T

6.1 Limitations of HRG-MCMC

Hierarchical Random Graph (HRG) [8] is a graph summary structure of size $O(n)$. It is a binary tree with leaves being graph nodes and $n-1$ internal nodes. Fig. 7 shows an example of a graph G and a possible dendrogram T. Each internal node r is equipped with a connection probability $p_r = \frac{e_r}{n_{Lr}.n_{Rr}}$, where n_{Lr}, n_{Rr} are the number of leaf nodes in the left and right subtrees L_r, R_r of r. Note that e_r is the number of edges connecting leaf nodes in L_r, R_r. Given a graph G, the number of possible dendrograms is super-exponential. In reality, we are concerned with the highly likely dendrograms, where the likelihood of T is measured as

$$\mathcal{L}(T, \{p_r\}) = \prod_{r \in T} p_r^{e_r} (1-p_r)^{n_{Lr} n_{Rr} - e_r} \tag{12}$$

The log-likelihood of T is

$$\log \mathcal{L}(T, \{p_r\}) = -\sum_{r \in T} n_{Lr} n_{Rr} h(p_r) \tag{13}$$

where $h(p_r) = -p_r \log p_r - (1-p_r)\log(1-p_r)$ is the Gibbs-Shannon entropy function. Intuitively, $-n_{Lr} n_{Rr} h(p_r)$ is maximized when p_r is close to 0 or 1, which means high-likelihood dendrograms are those that partition the nodes into groups between which connections are either very common or very rare. This happens, for example, in community-like or multi-partite graphs.

Xiao et at. [34] employ HRG model to address the problem of graph release under differential privacy. Their scheme HRG-MCMC first privately samples the dendrogram T by a privacy budget ϵ_1 and then adds noise $Lap(1/\epsilon_2)$ to e_r of the sampled dendrogram. MCMC method fits well to Exponential mechanism (Definition 3), i.e. to sample a huge space of states where direct computation of the normalization constant is infeasible, MCMC can be used to explore the space. To satisfy ϵ-DP, the acceptance probability in [34] is

$$\min(1, \frac{\exp(\frac{\epsilon_1}{2\Delta u} \log \mathcal{L}(\mathcal{T}'))}{\exp(\frac{\epsilon_1}{2\Delta u} \log \mathcal{L}(\mathcal{T}))}) \tag{14}$$

where Δu is the global sensitivity of a dendrogram's log-likelihood, and $\Delta u \approx ln(\frac{n^2}{4}) \approx 2 \ln n$. HRG-Fit in [8] performs MCMC with the acceptance probability similar to Formula 14 without $\frac{\epsilon_1}{2\Delta u}$ in either the nominator or the denominator. Clearly, HRG-Fit is $2\Delta u$-DP in nature. However, even at $\epsilon = 2\Delta u \approx 4 \ln n$, i.e. when HRG-MCMC becomes HRG-Fit, the sample graphs reconstructed are not good enough as we will see in Section 8.

HRG-MCMC induces a huge state space of $(2n-3)!! \approx \sqrt{2}(2n)^{n-1}e^{-n}$ possible dendrograms. Empirical evaluation by Clauset et al. shows that MCMC on HRG converge relatively quickly, with the likelihood reaching the plateau after roughly $O(n^2)$ steps. In ϵ-DP setting, Xiao et al. use $1000n$ steps for MCMC, along with a reconstruction time of $O(n^2 \log n)$. In the subsequent section 6.2, we present *HRG-FixedTree* with complexity of $O(m \log n)$, runnable on large graphs. We also present a fast sampling technique of $O(n \log n)$ on dendrograms in section 6.3.

6.2 HRG-FixedTree: Sampling over Node Permutations

In this section, we present our scheme, *HRG-FixedTree* for structural inference on large graphs. HRG-FixedTree reduces the sampling space to $n!$ by fixing a binary tree D and sampling good permutations of nodes for the leaves of D.

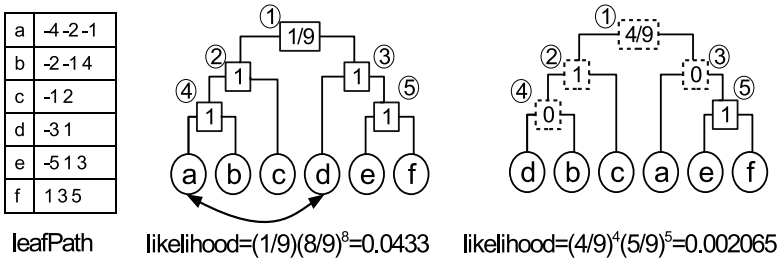


Figure 8: HRG-FixedTree

Fig. 8 illustrates one step of HRG-Fixed. We fix the dendrogram structure with the leaf nodes $\{a, b, c, d, e, f\}$. Then to perform a MCMC step, we randomly choose two nodes, say a and d, and swap them to get a permutation proposal. The affected internal nodes (dotted squares) are updated accordingly. The likelihoods of two permutations are shown under each permutation. Algorithm 3 is essentially similar to HRG-MCMC. The difference is that HRG-MCMC has costly MCMC steps while HRG-FixedTree ensures the logarithmic complexity for each MCMC run as we will see below.

Algorithm 3 HRG-FixedTree

Require: input graph $G = (V, E)$, fixed dendrogram structure D, privacy budget ϵ_1
Ensure: sampled permutation $S_{sampled}$
1: initialize the permutation S_0
2: **for** each step i in the Markov chain **do**
3: pick a neighboring permutation S' of S_{i-1} by randomly swapping two nodes in S_{i-1}.
4: accept the transition and set $S_i = S'$ with probability $min(1, \frac{\exp(\frac{\epsilon_1}{2\Delta_u} \log \mathcal{L}(S'))}{\exp(\frac{\epsilon_1}{2\Delta_u} \log \mathcal{L}(S_{i-1}))})$
5: **end for**
6: // until equilibrium is reached
7: **return** a sampled partition $S_{sampled} = S_i$

The state space of HRG-FixedTree is the set of all permutations over n nodes, so it is connected. It is straightforward to verify that the transitions performed in line 3 of HRG-FixedTree are *reversible* and *ergodic* (i.e. any pair of nodeset partitions can be connected by a sequence of such transitions). Hence, HRG-FixedTree has a unique stationary distribution in equilibrium. By empirical evaluation, we observe that HRG-FixedTree converges after $K|S|$ steps for $K = 1000$.

Fast MCMC step for HRG-FixedTree. Algorithm 4 is called in the line 3 of Algorithm 3. Whenever two leaf nodes u and v are swapped, we have to recompute the p_r for each internal node r along the two paths from u and v to their lowest common ancestor (see Fig. 8 for an example). FastSwap ensures the logarithmic time for this computation by precomputing the signature of each leaf node and storing in *leafPath*. Numbering the internal nodes of D from 1 to n, we compute the signature of each node in the form of an array of size $\log n$. The minus means that the node is in the leaf subtree of that internal node. For example, node a is in the left subtrees of internal nodes 4,2 and 1, so its signature is {-4,-2,-1}. We sort all the signatures in ascending order.

FastSwap works as follows. First, it finds affected internal nodes and store them to $listP$.

Algorithm 4 FastSwap

Require: input graph $G = (V, E)$, fixed dendrogram structure D, permutation S, $\mathcal{L}(S)$, two nodes u, v
Ensure: log-likelihood $\mathcal{L}(S')$ where S' is S with u, v swapped
1: find affected internal nodes $listP$
2: **for** each node $r \in listP$ **do**
3: $\quad \mathcal{L}(S) = \mathcal{L}(S) + n_{Lr} n_{Rr} h(p_r)$
4: **end for**
5: remove u and v from S
6: swap u, v in *leafPath*
7: add u and v back to S
8: **for** each node $r \in listP$ **do**
9: $\quad$ update e_r and p_r
10: $\quad \mathcal{L}(S) = \mathcal{L}(S) - n_{Lr} n_{Rr} h(p_r)$
11: **end for**
12: update pointers of u and v

This step costs $\log n$. Then the log-likelihood $\mathcal{L}(S)$ is subtracted by $-n_{Lr} n_{Rr} h(p_r)$ of all nodes in $listP$ (see formula 13). To remove u from S, we consider all $w \in N(u)$ and match the signatures of w and u to find the infected internal node r. The matching operation here is understood as to find two elements from the two signatures that their sum is zero. For example, to remove node a with $N(a) = \{b, c\}$, we match signatures of a and b to find the infected internal node is 4. Similarly, the infected node of the pair (a, c) is 2. The matching operation for each pair costs $\log n$ because all the signatures are already sorted. Let $\bar{d}$ be the average degree of G, the removal of u and v from S (line 5 of FastSwap) runs in $2\bar{d} \log n$. The replacement of u and v (after swappping) back to S (line 7 of FastSwap) runs in the same manner. Then all nodes $r \in listP$ are updated with new values of e_r and p_r and the log-likelihood $\mathcal{L}(S)$ is added by $-n_{Lr} n_{Rr} h(p_r)$. Finally, u and v update their pointers to parent/child nodes to keep the tree consistent.

Because each call to FastSwap costs $2\bar{d} \log n$ and we run $k.n$ MCMC steps, the complexity of HRG-FixedTree is $O(k.n\bar{d} \log n) = O(k.m. \log n)$ as stated in the following theorem.

Theorem 11. *HRG-FixedTree runs in $O(k.m \log n)$, where $k = 1000$.*

6.3 Fast Sampling From Dendrogram

Apart from costly MCMC of $1000n$ steps, another bottle-neck in HRG-MCMC [34] is its Algorithm 3 (Generate Sanitized Graph), which runs in $O(n^2 \log n)$ to generate a sanitized graph $\tilde{G}$ from a noisy $\tilde{T}$. In this section, we point out that from any dendrogram T, we can sample a graph $\tilde{G}$ in $O(n \log n)$.

In Algorithm 3 of HRG-MCMC, for each pair of nodes (i, j), the lowest common ancestor r of i and j is computed. Then the edge (i, j) is placed in $\tilde{G}$ with probability p_r. We call their approach *node-pair based reconstruction*. Now we propose *edge based reconstruction*.

First, at each internal node r, we build the nodesets of its left and right children. For example the root node on the right of Fig. 7 has left nodeset of $\{a, b, c\}$ and right nodeset of $\{d, e, f\}$. This step costs time and space of $O(n \log n)$ by bottom-up construction. Then we consider each internal node r (there are $n - 1$ such nodes) and sample e_r edges between

r's left and right nodesets. This step costs $O(m)$ because $\sum_{r \in T} e_r = m$. In reality, $m = O(n \log n)$ so the total complexity of *edge based reconstruction* is $O(n \log n)$.

Theorem 12. *From any dendrogram generated by HRG-MCMC or HRG-FixedTree, a sample graph can be generated in $O(n \log n)$.*

7 Comparison of Schemes

We compare the aforementioned schemes by their model complexity, runtime/space complexity and the graph sizes that they can run on. TmF and EdgeFlip output directly the noisy graphs while the two HRG-based schemes as well as DER and 1K-series employ an intermediary structure to model the true graph.

DER [6] uses a quadtree to partition the adjacency matrix to quadrants in h levels. All nodes (except the root node) store the noisy count of 1-cells in their rectangular subregion. So the model complexity of DER is $O(2^h)$. 1K-series has the model complexity of $O(n)$ by using the degree sequence. HRG-MCMC and HRG-FixedTree privately fit the true graph into a binary tree (dendrogram), so their model complexity is also $O(n)$.

DER costs $O(n^2)$ for storing the *count summary matrix* as shown in [6]. All the other schemes incur $O(m)$ for storing the graph G. Table 1 shows the comparison of the schemes investigated in this paper.

Table 1: Comparison of schemes

		Model	Runtime	Space	Graph size
Direct publication	TmF	n/a	$O(m)$	$O(m)$	10^6
	EdgeFlip* [23]	n/a	$O(m)$	$O(m)$	10^6
Model-based publication	HRG-MCMC [34]	$O(n)$	$O(n^2)$	$O(m)$	10^4
	HRG-FixedTree	$O(n)$	$O(k.m \log n)$	$O(m)$	10^6
	DER [6]	$O(2^h)$	$O(n^2)$	$O(n^2)$	10^4
	1K-series [32]	$O(n)$	$O(m)$	$O(m)$	10^6

(* Recall that EdgeFlip is linear only for $\epsilon \geq \ln(\frac{n(n-1)}{2m} - 1)$)

8 Experiments

Our evaluation aims to compare the efficiency (in runtime) and the effectiveness (in terms of utility metrics) among the competitors. We pick six small and medium-sized graphs and three large ones [2]. In Table 2, $\text{logLK}_{\text{Louvain}}$, $\text{logLK}_{\text{HRG-MCMC}}$, $\text{logLK}_{\text{HRG-Fixed}}$ and $\text{logLK}_{\text{Init}}$ are the log-likelihoods of the *non-private* dendrograms created by Louvain algorithm [3], HRG-MCMC, HRG-FixedTree and bottom-up binary initialization (i.e. nodes 1 and 2 are paired, nodes 3 and 4 are paired and so on) respectively. By Louvain method, we recursively partition the graph into nested communities until their sizes are smaller than a threshold (e.g. 50 or 100 nodes). Then we build a dendrogram from those communities.

All algorithms are implemented in Java and run on a desktop PC with $Intel^{®}$ Core i7-4770@ 3.4Ghz, 16GB memory.

The typical utility metrics are listed in Section 8.1. We assess the effectiveness of TmF in Section 8.2. Then the non-private versions of HRG-based schemes and 1K-series are evaluated in Section 8.3. Finally, Section 8.4 compares the overall performance of all competitors and clarifies the contribution of each utility metric.

[2] available at http://www-personal.umich.edu/~mejn/netdata/ and http://snap.stanford.edu/data/index.html

8.1 Utility Metrics

We use the following statistics for utility measurement, according to [4, 6], in which there are five degree-based metrics, five path-based metrics and two other metrics.

Degree-based metrics

- Average degree: $S_{AD} = \frac{1}{n}\sum_{v\in V} d_v$
- Maximal degree: $S_{MD} = \max_{v\in V} d_v$
- Degree variance: $S_{DV} = \frac{1}{n}\sum_{v\in V}(d_v - S_{AD})^2$
- Power-law exponent of degree sequence: S_{PL} is the estimate of γ assuming the degree sequence follows a power-law $\Delta(d) \sim d^{-\gamma}$
- Degree distribution: S_{DD} is the normalized degree histogram.

Path-based metrics

- Average distance: S_{APD} is the average distance among all pairs of vertices that are path-connected.
- Effective diameter: S_{EDiam} is the 90-th percentile distance among all path-connected pairs of vertices.
- Connectivity length: S_{CL} is defined as the harmonic mean of all pairwise distances in the graph.
- Diameter : S_{Diam} is the maximum distance among all path-connected pairs of vertices.
- Distance distribution: S_{PDD} is the normalized node-pair shortest-path histogram.

Other metrics

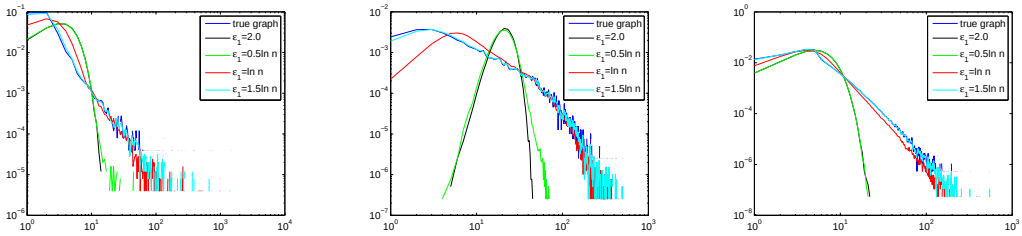
- Clustering coefficient: $S_{CC} = \frac{3N_\Delta}{N_3}$ where N_Δ is the number of triangles and N_3 is the number of connected triples.
- Cut queries: $S_{cut}(X, Y)$ is the number of edges between two disjoint node sets X and Y.

All of the above statistics are taken average over 20 sample graphs. S_{APD}, S_{CL}, S_{EDiam}, S_{Diam} are computed exactly in six small graphs. In *amazon*, *dblp* and *youtube*, we estimate $S_{APD}, S_{CL}, S_{EDiam}$ and S_{Diam} using HyperANF [5]. The relative error (rel.err) for each metric S is computed as $\frac{|S(G)-S_{avg}(\tilde{G})|}{S(G)}$ except S_{DD} and S_{PDD} whose errors are computed as $|S(G) - S_{avg}(\tilde{G})|_1/2$ (total variation distance). The number of cut queries is 1,000 and the size of node set does not exceed 500.

Table 2: Graph dataset statistics (k:thousand, m:million)

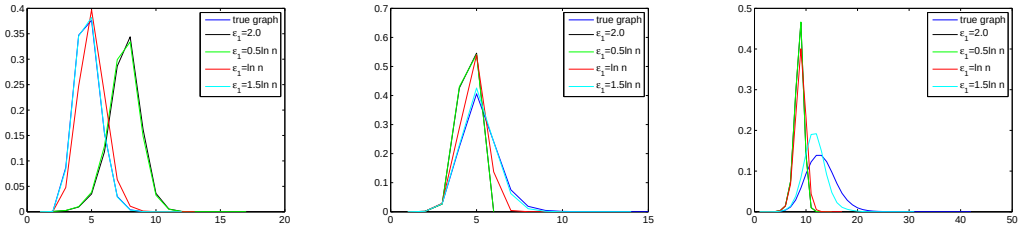
Dataset	#Nodes	#Edges	logLK$_{\text{Louvain}}$	logLK$_{\text{HRG-MCMC}}$	logLK$_{\text{HRG-Fixed}}$	logLK$_{\text{Init}}$
polbooks	105	441	-957	**-781**	-896	-1248
polblogs	1,222	16,714	-65.6k	-59.3k	**-50.7k**	-74k
as20graph	6,474	12,572	**-71k**	-73.8k	-87.6k	-100k
wiki-Vote	7,066	100,736	-576k	-556k	**-432k**	-618k
ca-HepPh	11,204	117,619	**-328k**	-746k	-377k	-854k
ca-AstroPh	17,903	196,972	**-903k**	-1426k	-1,006k	-1,515k
amazon	334k	925k	**-3.6m**	n/a	-8.6m	-11.1m
dblp	317k	1,050k	**-5.0m**	n/a	-9.3m	-11.1m
youtube	1,134k	2,987k	**-24.7m**	n/a	-31.5m	-35.8m

The schemes are abbreviated as EdgeFlip (EF), Top-m-Filter (TmF), 1K-series (1K), DER, HRG-MCMC and HRG-FixedTree (HRG-Fixed). We test all schemes for ϵ in $\{2.0, 0.25\ln n, 0.5\ln n, 0.75\ln n, \ln n, 1.25\ln n, 1.5\ln n\}$. The rationale behind this choice of ϵ is presented in Section 3.2. By ρ-differential identifiability [16], the identification risk is $\rho = O(1/n)$ (resp. $O(1/\sqrt{n})$) for $\epsilon = \ln n$ (resp. $\epsilon = 1.5\ln n$). At $\epsilon = 2\ln n$, $\rho = O(1)$, i.e. blatantly non-private.



(a) as20graph (b) ca-AstroPh (c) amazon

Figure 9: Effectiveness of TmF: degree distributions (log-log scale)



(a) as20graph (b) ca-AstroPh (c) amazon

Figure 10: Effectiveness of TmF: distance distributions

Therefore, the values of $\epsilon \geq 2 \ln n$ are not valid. Recall that EF is only linear with ϵ in $\{\ln n$, $1.25\ln n$, $1.5\ln n\}$ while DER and HRG-MCMC only runs on six small graphs. We also test EF at $\epsilon = 0.75 \ln n$ on six small graphs when the number of edges in output graphs starts to become super-linear.

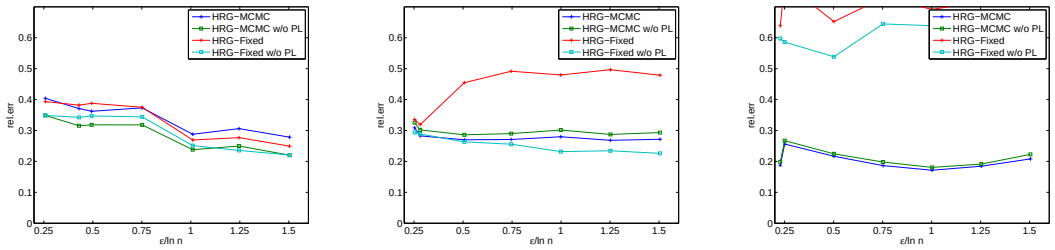
8.2 Effectiveness of TmF

We assess the utility of TmF by varying ϵ_1 while fixing $\epsilon_2 = 0.1$. Figures 9 and 10 display S_{DD} and S_{PDD} for three graphs *as20graph*, *ca-AstroPh* and *amazon*. At $\epsilon = 2.0$ and $0.5 \ln n$, TmF produces highly deformed degree distributions and distance distributions. Additionally, the S_{DD} and S_{PDD} at $\epsilon = 2.0$ and $0.5 \ln n$ are nearly identical. This could be explained by the large edit distance $D(G, \tilde{G})$ at those values. At $\epsilon = \ln n$, the noisy degree and distance distributions become asymptotic to the true ones.

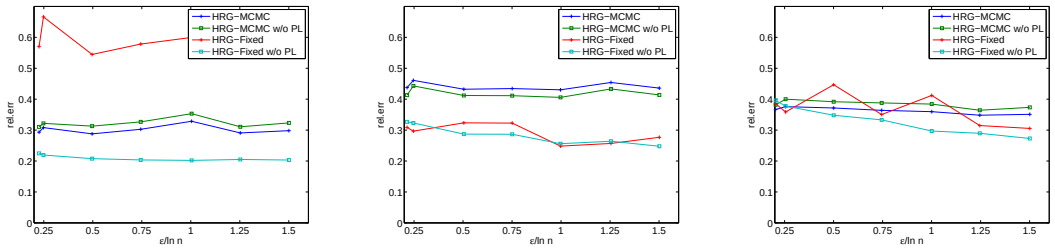
8.3 HRG-based Schemes

In this section, we assess the effectiveness of HRG model. We try to answer the question: *is HRG a good model for graph summarization?* The columns $\text{logLK}_{\text{Louvain}}$, $\text{logLK}_{\text{HRG-MCMC}}$ and $\text{logLK}_{\text{HRG-Fixed}}$ of Table 2 shows the log-likelihood of *non-private* dendrograms generated by Louvain method, HRG-MCMC and HRG-Fixed. As one of the best community detection techniques, Louvain method (non-private) [3] gives us high-likelihood dendrograms, especially on the three large graphs. HRG-MCMC and HRG-Fixed always return dendrograms of higher likelihood than the initial D_0 ($\text{logLK}_{\text{Init}}$). The non-private HRG-MCMC samples better dendrograms than HRG-Fixed only on *polbooks* and *as20graph*.

As we can see in Table 3, despite the high-likelihood scores, dendrograms generated by Louvain and non-private HRG based schemes cannot reduce the relative errors signifi-



(a) polbooks ($\ln n = 4.7$) (b) polblogs ($\ln n = 7.1$) (c) as20graph ($\ln n = 8.8$)



(d) wiki-Vote ($\ln n = 8.9$) (e) ca-HepPh ($\ln n = 9.3$) (f) ca-AstroPh ($\ln n = 9.8$)

Figure 11: Relative errors of HRG-MCMC and HRG-Fixed (with and without S_{PL})

cantly. This fact implies inherent limitations of HRG models. In other words, ϵ-DP schemes based on HRG models are not consistent with ϵ. The same conclusion could be draw for 1K-series, i.e. compared with the true graph, the graphs regenerated from the true degree sequence have a significant gap in utility metrics.

Fig. 11 compares the relative errors of HRG-MCMC and HRG-Fixed on six small/medium graphs in two cases: with and without S_{PL}. Clearly, the relative error of S_{PL} in HRG-MCMC is much more stable than in HRG-Fixed. In other words, S_{PL} is the main contributor to the high relative error of HRG-Fixed in most of the cases. Without S_{PL}, HRG-Fixed outperforms HRG-MCMC on *polblogs*, *wiki-Vote*, *ca-HepPh* and *ca-AstroPh*. Details on the consistency of individual utility metric are investigated in the next section.

Table 3: Relative error of *non-private* 1K-series and HRG-based schemes

Dataset	1K-series	Louvain	HRG-MCMC	HRG-Fixed
polbooks	**0.191**	0.390	0.242	0.255
polblogs	**0.089**	0.405	0.277	0.474
as20graph	**0.122**	0.712	0.186	0.693
wiki-Vote	**0.046**	0.486	0.318	0.452
ca-HepPh	0.248	0.357	0.439	**0.241**
ca-AstroPh	**0.220**	0.370	0.352	0.309
amazon	0.381	**0.372**	n/a	0.511
dblp	**0.265**	0.371	n/a	0.327
youtube	**0.200**	0.584	n/a	0.523

8.4 Comparative Evaluation

We report the comparisons between the competitors in Fig. 12. As ϵ increases (lower privacy guarantee), we gain better utility (lower relative errors) for TmF and EF while the other methods do not have this trend. TmF performs poorly for ϵ in $\{2, 0.25 \ln n, 0.5 \ln n, 0.75 \ln n\}$ because of the number of passing 1-cells is low. As ϵ exceeds the threshold $\epsilon_t \approx \ln n$, the edit distance $D(G, \tilde{G})$ decreases quickly (i.e. the number of passing 1-cells increases, Fig. 4), so does the relative error. At $\epsilon = 0.75 \ln n$, TmF works slightly better than EF except on *as20graph*. We do not run EF at $\epsilon = 0.75 \ln n$ on the three large graphs because the number of edges becomes super-linear in this case and the relative errors of the five degree-based metrics overwhelm the rest.

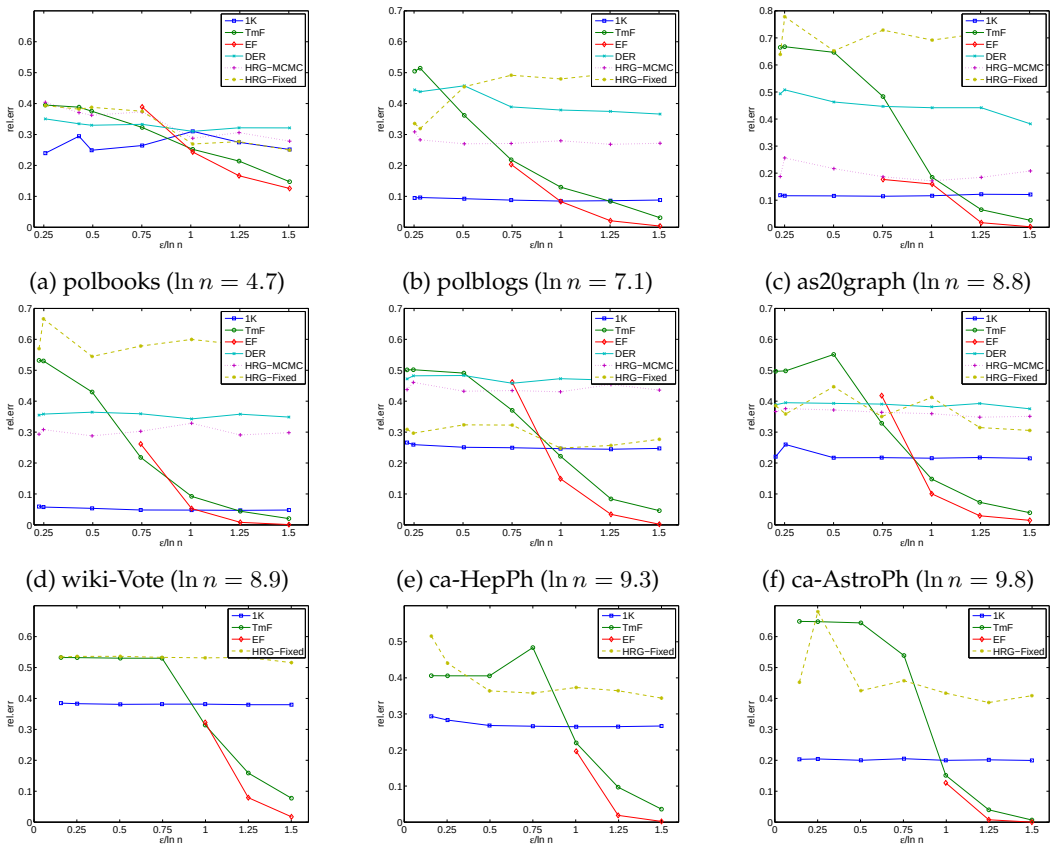
1K-series provides the best utility for ϵ in $\{2, 0.25 \ln n, 0.5 \ln n, 0.75 \ln n\}$. Because the degree sequence has small sensitivity, a small privacy budget is enough to keep the degree sequence almost identical to that of the true graph while larger values of ϵ are redundant. Put differently, 1K-series benefits a lot from the five degree-based metrics (details below in Figures 13, 14 and 15). On the six small graphs, HRG-MCMC performs better than DER but the gap is small in *polbooks*, *wiki-Vote*, *ca-HepPh* and *ca-AstroPh*.

Over the six small and medium graphs, HRG-Fixed is comparable to HRG-MCMC on *polbooks* and *ca-AstroPh*. It works better than HRG-MCMC only on *ca-HepPh*. However, the near-linear complexity makes HRG-Fixed runnable on large graphs.

Except TmF and EdgeFlip, the remaining schemes (1K, HRG-MCMC, HRG-Fixed and DER) do not show strong consistency with ϵ. We argue that the consistency of TmF and EdgeFlip is due to their nature of direct publication while the remaining schemes are model-based (indirect) and rely too much on regeneration processes. In direct methods, we quantify exactly the relationship between the edit distance $D(G_1, \tilde{G})$ and ϵ. On the contrary, the similar quantification in model-based methods is not well defined and to our knowledge, has not been considered in the literature.

To see how much each metric contributes to the average relative error, we plot the relative errors for all twelve metrics in Figures 13, 14 and 15. We pick the three graphs *polbooks*, *ca-HepPh* and *dblp*. Because all the subfigures have the same set of curves, we only show the legend in the first subfigure to save the space for the plots. TmF and EF are consistent with increasing ϵ on all metrics except S_{AD} which is preserved in expectation, i.e. zero relative error. The model-based schemes show weak consistency on several metrics, for example HRG-Fixed on S_{DD} and S_{Diam}. 1K-series, the best scheme for small ϵ, has nearly zero errors on the five degree-based metrics but it gives poor results on the path-based metrics as well as S_{CC}. HRG-Fixed outperforms HRG-MCMC on many metrics, especially on the five path-based metrics and S_{CC}. The good performance of 1K-series and HRG-based schemes on different subsets of metrics confirms again the information loss in each model-based scheme. Reversely, it suggests that more complex summarization structures can combine the best of both models. We leave this for future work.

The runtime is reported in Table 4. As expected, TmF, EF and 1K-series run very fast, linear in the number of edges. They run in several seconds on *youtube*, the largest graph considered in this paper. Although incurring the quadratic time complexity as HRG-MCMC, DER runs faster because HRG-MCMC has the big constant $k = 1000$. Both of them is infeasible on the three large graphs. Recall that HRG-Fixed runs in $O(m \log n)$, an improvement over HRG-MCMC [34].



(a) polbooks ($\ln n = 4.7$) (b) polblogs ($\ln n = 7.1$) (c) as20graph ($\ln n = 8.8$)

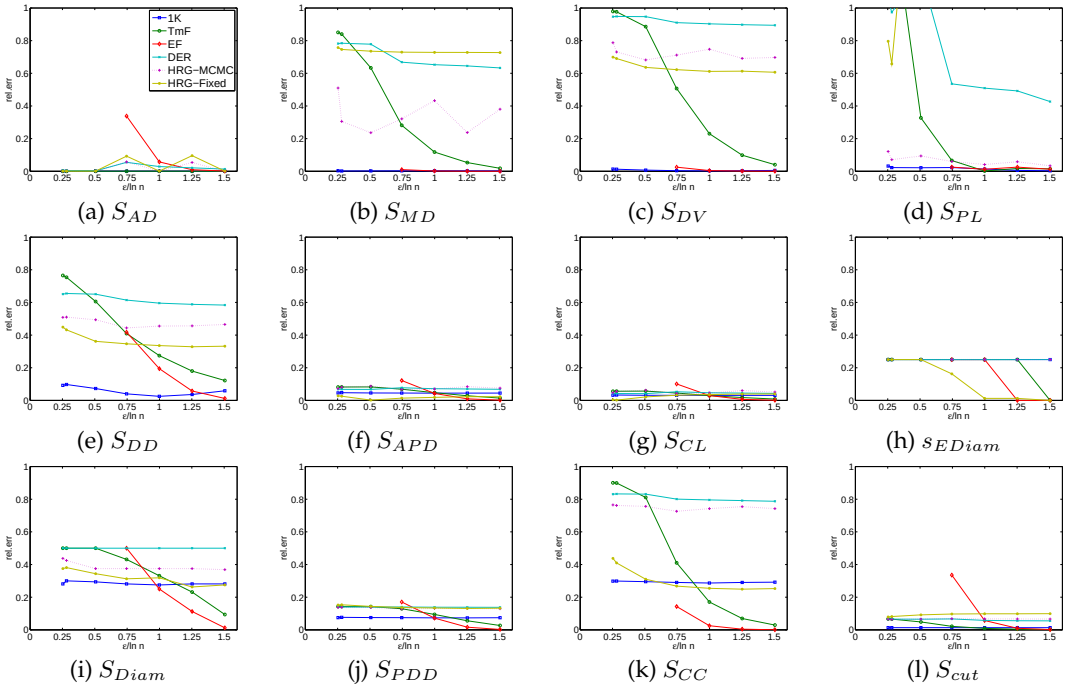
(d) wiki-Vote ($\ln n = 8.9$) (e) ca-HepPh ($\ln n = 9.3$) (f) ca-AstroPh ($\ln n = 9.8$)

(g) amazon ($\ln n = 12.7$) (h) dblp ($\ln n = 12.7$) (i) youtube ($\ln n = 13.9$)

Figure 12: Comparative evaluation: the relative error is averaged on twelve utility metrics

9 Conclusion

We prove an upper bound $O(\ln n)$ for the privacy budget ϵ that any differentially private scheme for graph release should not exceed. Based on a filtering technique, we design the algorithm TmF that reduces the edit distance between the noisy graph and the true graph to $O(1)$ at an upper bound of $\epsilon = O(\ln n)$. By further investigation of EdgeFlip, we show that it also satisfies the upper bound. Moreover, TmF and EdgeFlip, as representatives of direct publication schemes, show a strong consistency with the large privacy budgets. We explain the inherent information loss in model-based methods which prevents them from achieving the perfect utility for $\epsilon \geq 1.5 \ln n$. On scalability, we show that TmF, EdgeFlip (partially), HRG-FixedTree and 1K-series have linear complexity while HRG-MCMC and DER do not. The comprehensive experiments demonstrate the efficiency and effectiveness of our TmF and explain the inconsistency in the model-based schemes HRG-MCMC, HRG-Fixed, DER and 1K-series. For future work, we intend to (1) find better consistent schemes and (2) examine summary structures for graphs other than HRG and 1K-series.

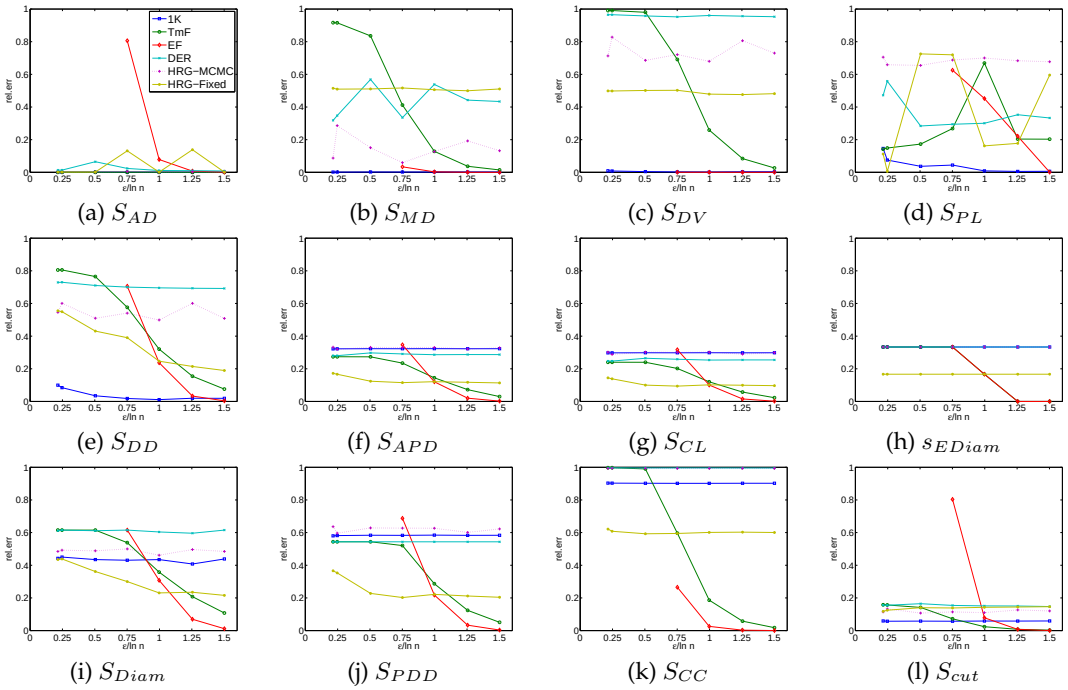


(a) S_{AD} (b) S_{MD} (c) S_{DV} (d) S_{PL}

(e) S_{DD} (f) S_{APD} (g) S_{CL} (h) s_{EDiam}

(i) S_{Diam} (j) S_{PDD} (k) S_{CC} (l) S_{cut}

Figure 13: Relative errors of utility metrics on *polblogs*

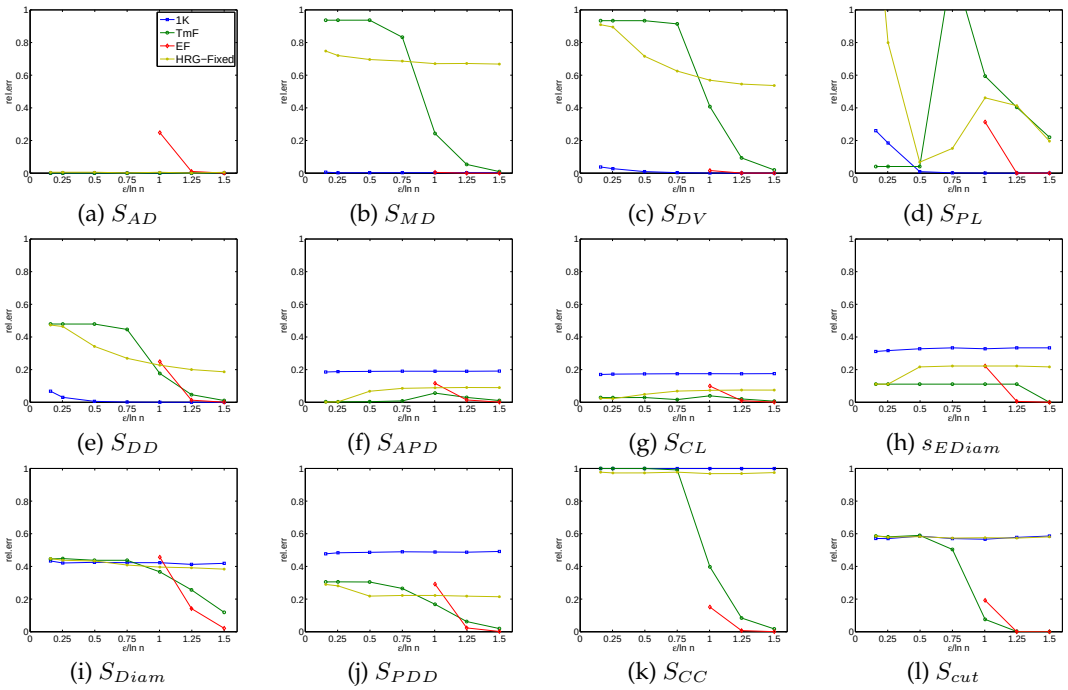


(a) S_{AD} (b) S_{MD} (c) S_{DV} (d) S_{PL}

(e) S_{DD} (f) S_{APD} (g) S_{CL} (h) s_{EDiam}

(i) S_{Diam} (j) S_{PDD} (k) S_{CC} (l) S_{cut}

Figure 14: Relative errors of utility metrics on *ca-HepPh*



(a) S_{AD} (b) S_{MD} (c) S_{DV} (d) S_{PL}

(e) S_{DD} (f) S_{APD} (g) S_{CL} (h) s_{EDiam}

(i) S_{Diam} (j) S_{PDD} (k) S_{CC} (l) S_{cut}

Figure 15: Relative errors of utility metrics on *dblp*

References

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190. ACM, 2007.

[2] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 273–282. ACM, 2007.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

Table 4: Runtime in milliseconds

Dataset	1K	TmF	EF	DER	HRG-MCMC	HRG-Fixed
polbooks	6	3	3	13	3922	665
polblogs	39	24	36	1,486	1,576,871	20,878
as20graph	47	30	31	25,140	544,298	31,591
wiki-Vote	232	113	86	45,486	1,185,644	153,488
ca-HepPh	287	135	95	81,989	5,547,744	206,520
ca-AstroPh	469	226	97	182,556	14,734,655	358,263
amazon	2,780	1,565	1,185	n/a	n/a	3,743,018
dblp	3,188	1,576	524	n/a	n/a	4,091,923
youtube	11,369	5,265	1,853	n/a	n/a	13,049,549

[4] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting uncertainty in graphs for identity obfuscation. *Proceedings of the VLDB Endowment*, 2012.

[5] P. Boldi, M. Rosa, and S. Vigna. Hyperanf: Approximating the neighbourhood function of very large graphs on a budget. In *WWW*, pages 625–634. ACM, 2011.

[6] R. Chen, B. C. Fung, P. S. Yu, and B. C. Desai. Correlated network data publication via differential privacy. *VLDB Journal*, 23(4):653–676, 2014.

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

[8] A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[9] G. Cormode, C. Procopiuc, D. Srivastava, and T. T. Tran. Differentially private summaries for sparse data. In *ICDT*, pages 299–311. ACM, 2012.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *TCC*, pages 265–284, 2006.

[11] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[12] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[13] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

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

[15] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

[16] J. Lee and C. Clifton. Differential identifiability. In *KDD*, pages 1041–1049. ACM, 2012.

[17] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, pages 93–106. ACM, 2008.

[18] W. Lu and G. Miklau. Exponential random graph estimation under differential privacy. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 921–930. ACM, 2014.

[19] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat. Systematic topology analysis and generation using degree correlations. In *SIGCOMM*. ACM, 2006.

[20] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

[21] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30. ACM, 2009.

[22] D. Mir and R. N. Wright. A differentially private estimator for the stochastic kronecker graph model. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 167–176. ACM, 2012.

[23] Y. Mülle, C. Clifton, and K. Böhm. Privacy-integrated graph clustering through differential privacy. In *PAIS 2015*, 2015.

[24] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[25] H. H. Nguyen, A. Imine, and M. Rusinowitch. Anonymizing social graphs via uncertainty semantics. In *ASIACCS*, pages 495–506. ACM, 2015.

[26] H. H. Nguyen, A. Imine, and M. Rusinowitch. Differentially private publication of social graphs at linear cost. In *ASONAM*, pages 596–599. ACM, 2015.

[27] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84. ACM, 2007.

[28] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao. Measurement-calibrated graph models for social network experiments. In *WWW*. ACM, 2010.

[29] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98. ACM, 2011.

[30] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[31] C.-H. Tai, P. S. Yu, D.-N. Yang, and M.-S. Chen. Privacy-preserving social network publication against friendship attacks. In *KDD*. ACM, 2011.

[32] Y. Wang and X. Wu. Preserving differential privacy in degree-correlation based graph generation. *TDP*, 6(2):127, 2013.

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

[34] Q. Xiao, R. Chen, and K.-L. Tan. Differentially private network data release via structural inference. In *KDD*, pages 911–920. ACM, 2014.

[35] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515. IEEE, 2008.

[36] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. *VLDB Endowment*, 2009.