Social Graph Anonymization

Huu-Hiep Nguyen

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Anonymisation de Graphes Sociaux
∴ ∵ ∴
Social Graph Anonymization

THÈSE

présentée et soutenue publiquement le date

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

NGUYEN Huu-Hiep

Composition du jury

Rapporteurs :	Clémence Magnien	CNRS, LIP6
	Catuscia Palamidessi	INRIA, LIX
Examinateurs :	Isabelle Chrisment	Université de Lorraine, LORIA
	Sylvain Peyronnet	ix-labs
	Emmanuel Viennet	Université Paris 13, L2TI
Directeurs de thèse :	Abdessamad Imine	Université de Lorraine, LORIA
	Michaël Rusinowitch	INRIA, LORIA

Institut National de Recherche en Informatique et en Automatique
Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

INVENTEURS DU MONDE NUMÉRIQUE

Loria Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thesul.

Acknowledgements

I am deeply indebted to my PhD thesis advisors, Michaël Rusinowitch and Abdessamad Imine, for giving me an excellent opportunity to conduct research on privacy for social networks, for supporting me throughout the three-year PhD period. Their great inspiration, precise guidance and constant encouragement have helped me a lot to improve important skills of a researcher. I have learnt a great deal of problem exploration and solving from them. They have taught me how to develop from half-baked ideas to complete and practical contributions which, in the end, need to be presented in convincing arguments.

I am very grateful to the members of my dissertation committee, Dr. Clémence Magnien, Dr. Catuscia Palamidessi, Professor Isabelle Christment, Dr. Sylvain Peyronnet and Professor Emmanuel Viennet for having accepted to assess my thesis and for their reviewing efforts and insightful remarks.

To be a member of PESTO team is also an honour for me. I would like to thank all the people of PESTO team, including current and former people that I have had chance to interact with in a wonderful environment, Christophe Ringeissen, Mathieu Turuani, Véronique Cortier, Steve Kremer, Laurent Vigneron, Vincent Cheval, Jannik Dreier, Walid Belkhir, Houari Mahfoud, Éric Le Morvan, Ludovic Robin, Ivan Gazeau, Alicia Filipiak and Joseph Lallemand.

Additionally, I am greatly thankful the INRIA-CORDI fellowship that helped my PhD work possible. Many thanks also go to my Vietnamese friends in Nancy, who have been with me to share memorable moments.

Last, and most important, I would like to thank my parents for their love and support during my stay away from home. It is to them that I dedicate this thesis.

To my parents.

Abstract

Privacy is a serious concern of users in daily usage of social networks, especially when online social networks offer free services in exchange for large collection of user information. The motto *"If you're not paying for it; you're the product"* demands effective measures for user privacy protection. At the same time, social networks are a valuable data source for large-scale studies on social organization and evolution. Sanitized social network information is therefore occasionally shared with third parties by service providers. On the other side, by participating to social networks, users keep their own data and they may use the very infrastructure of service providers to gather local view of the network to some extent not only restricted to 1-hop friends, for example by exchanging noisy links. To this end, the problems of privacy protection for social network data are still calling for effective and efficient approaches both in centralized and decentralized settings.

This thesis addresses three privacy problems of social networks: graph anonymization, private community detection and private link exchange. The main goal is to provide new paradigms for publication of social graphs in noisy forms, private community detection over graphs as well as distributed aggregation of graphs via noisy link exchange processes. We start the thesis by giving the big picture of data privacy in social networks and clarifying the categories to which our work belongs. Then we present our three contributions as follows.

First, we tackle the problem of graph anonymization via two different semantics: uncertainty semantics and differential privacy. These are two main categories of graph anonymization in the literature. As for uncertainty semantics, we propose a general obfuscation model called *Uncertain Adjacency Matrix* (UAM) that keeps expected node degrees equal to those in the unanonymized graph. We analyze two recently proposed schemes and show their fitting into the model. We also point out disadvantages in each method and present our scheme *Maximum Variance* (MaxVar) to fill the gap between them. Moreover, to support fair comparisons, we develop a new tradeoff quantifying framework by leveraging the concept of incorrectness in location privacy research. Experiments on large social graphs demonstrate the effectiveness of MaxVar.

Apart from privacy notion via uncertainty semantics, we contribute a new algorithm for graph anonymization under differential privacy, also known as ϵ-DP graph publication. 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 thesis, we categorize the state-of-the-art of ϵ-DP graph publication 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.

Second, we present the problem of community detection under differential privacy. Complex networks usually expose community structure with groups of nodes sharing many links with the other nodes in the same group and relatively few with the nodes of the rest. This feature captures valuable information about the organization and even the evolution of the network.

Over the last decade, a great number of algorithms for community detection have been proposed to deal with the increasingly complex networks. However, the problem of doing this in a private manner is rarely considered. We analyze the major challenges behind the problem and propose several schemes to tackle them from two perspectives: input perturbation and algorithm perturbation. We choose Louvain method as the back-end community detection for input perturbation schemes and propose the method *LouvainDP* which runs Louvain algorithm on a noisy super-graph. For algorithm perturbation, we design *ModDivisive* using exponential mechanism with the modularity as the score. We have thoroughly evaluated our techniques on real graphs of different sizes and verified their outperformance over the state-of-the-art.

Finally, we propose protocols for *private link exchange* over social graphs. It is motivated by the fact that current online social networks (OSN) keep their data secret and in centralized manner. Researchers are allowed to crawl the underlying social graphs (and data) but with limited rates, leading to only partial views of the true social graphs. To overcome this constraint, we may start from user perspective, the contributors of the OSNs. More precisely, if users cautiously collaborate with one another, they can exchange noisy friend lists with their neighbors in several rounds to get better views of the true social graph. The problem is unique in the sense that the disseminated data over the links are the links themselves. However, there exist fundamental questions about the feasibility of this model. The first question is how to define simple and effective privacy concepts for the link exchange processes. The second question comes from the high volume of link lists in exchange which may increase exponentially round after round. While storage and computation complexity may not be big problems, communication costs are non-trivial. We address both the questions by a simple (α, β)-exchange using Bloom filters. We evaluate our proposed schemes on various synthetic graph models and draw a number of critical findings.

Contents

List of Figures

Chapter 1

Introduction

Contents

1.1 Motivation

1.1.1 Privacy in Social Networks

A social network is a social structure made up of a set of social actors, sets of relationships and other social interactions between actors. With the emergence of online social networks [1], people have a powerful social media that they have never seen before. Facebook [2], the biggest OSN, has over 1.65 billion monthly active users which is a 15 percent increase year over year. Most of the OSNs offer free services in exchange for large collection of user information, used in targeting advertising by service providers. Because the protection of user data is not always guaranteed, social network user privacy is usually put at risk. The leakage of information may be caused by many reasons: the carelessness of users, the sharing to third parties by service providers and the brutal attacks by cybercriminals [3].

OSNs are also an important source of Big Data, a changing-world industry. Social networks, together with other complex networks, provide useful constructs for studies in social sciences, statistics and graph theory. Nowadays, people are concerned about how the economic actors can maximize the benefits of big data while minimizing its risks. From the perspective of OSN

[1] https://en.wikipedia.org/wiki/Social_networking_service
[2] https://zephoria.com/top-15-valuable-facebook-statistics/
[3] https://heimdalsecurity.com/blog/10-surprising-cyber-security-facts-that-may-affect-your-online-safety/

users, they always wish the benefits far beyond the risks. This fact is backed by the rapid development of increasingly complex privacy policies by service providers (e.g. Facebook). In the next section, we motivate some risk-minimizing approaches for social networks based on their underlying social graphs.

1.1.2 Privacy Threats Raised by Publishing Social Graphs

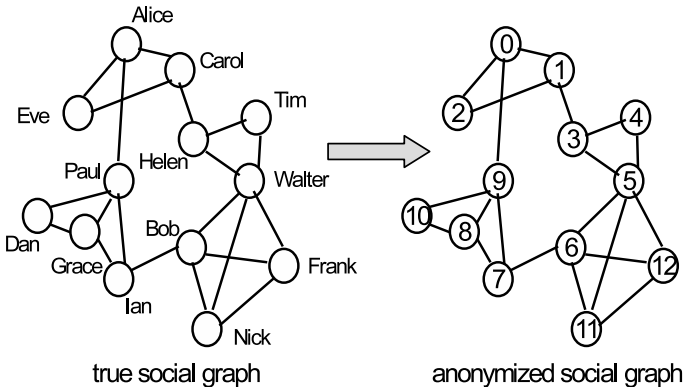
Li et al. [54] summarize two types of information disclosure in the literature: *identity disclosure* and *attribute disclosure*, and identity disclosure often leads to attribute disclosure. Whenever an attacker reveals the mapping from a database record to a specific real-world entity, we say that an identity disclosure has occurred. Attribute disclosure implies a successful inference of sensitive attributes of a target user. Also, the privacy literature identifies two main class of privacy mechanisms: *interactive* and *non-interactive*. In the interactive setting, an attacker is allowed to pose queries to a database and the database owner responds with noisy answers. Based on the information gleaned from previous answers, the attacker may pose *adaptive* queries to avoid wasteful answers. In non-interactive setting, the database owner publishes an anonymized version of the database to meet certain privacy requirements (e.g. user names in published database may be replaced by dummy numbers). Earlier research on data privacy mainly focused on *single-table* data in which the rows represent independent and identically distributed (i.i.d) records and the columns represent attributes [54, 59, 95]. However, real-world data is often *relational* and records are related to one another or to records from other tables. This fact raises radical challenges to preserving the privacy of users.

Each user in online social networks is represented by a rich profile as a set of attributes (e.g. gender, date of birth, hobbies, marital status and location) and their relationships (i.e. friendship links and memberships to groups of interest). In particular, social graphs which represent the underlying structure of social networks are relational data and they possess a lot of information that can be exploited by data analysts as well as attackers. Generally, OSN users have strong perception that service providers keep their private information secure [4] and their identities are blended among other users in published anonymized data. However, strong correlation between users in the same database and with users in other auxiliary databases makes large-scale re-identification attacks possible [70–72, 111].

We illustrate a simple re-identification attack in Fig. 1.1. A social graph of thirteen users is naively anonymized by replacing user names with dummy numbers. An attacker can reidentify the users by crawling the number of friends each user has. This is completely feasible because the OSNs like Facebook allow users to leave their number of friends in public mode. Assuming that the attacker tries to map the node 5 in the anonymized graph to a certain user in the true social graph, he will iterates the OSN and searches for users having five friends. Then, the user Walter will be revealed. After that, further information about Walter may be explored by more complex inference attacks, e.g. by examining which groups Walter joins [111].

1.1.3 New Privacy Preserving Mechanisms for Social Graphs

With the emergence of increasingly complex networks [73], the research community requires large and reliable graph data to conduct in-depth studies. However, this requirement usually conflicts with privacy rights 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 [4, 46]. Therefore, many graph anonymization schemes have been proposed [23, 55, 96, 103, 112, 113]. In any anonymization (or obfuscation) scheme, we have two conflicting



Degree	Users
2	Eve, Dan, Tim
3	Alice, Carol, Helen, Frank, Nick, Ian
4	Bob, Paul, Grace
5	Walter

Figure 1.1: An example of re-identification attack

forces: *privacy* and *utility*. Higher privacy requirements would lead to more modifications to the true graph, so the anonymized graph gets more distorted, reducing the accuracy of intended computations (utility) and vice versa. The conflicts between privacy and utility is also known as privacy/utility *tradeoffs*. As a consequence, the designers of anonymization schemes must clarify their choice of privacy and utility measures (or metrics). Usually, utility for social graphs is measured in graph metrics such as degree sequence, shortest path distribution, community structure and so on. Similarly, privacy is defined by various measures in terms of the success level of inference attacks mounted on the anonymized graphs. We make a short survey on privacy and utility metrics in Sections 2.2.6 and 2.2.7.

Given an unlabeled undirected graph, the existing anonymization methods fall into five main categories.

- The first includes *random* addition, deletion and switching of edges [13, 46, 107, 109] to prevent the re-identification of nodes or edges.
- The second provides k-anonymity [95] by *deterministic* edge additions or deletions [23, 55, 96, 103, 112, 113], assuming attacker's background knowledge regarding certain properties of its target nodes.
- The third relies on *generalization* [18, 46, 97] and clusters nodes into super nodes of size at least k where k is a privacy parameter.
- The methods in the fourth category assign edge probabilities to add uncertainty to the true graph. The edges probabilities may be computed explicitly as in [11] or implicitly via random walks [66].
- Finally, there are schemes for graph anonymization based on the notion of *differential privacy* [33].

Note that schemes in the third and fourth categories induce *possible world* models, i.e. each edge in the anonymized graph exists with a certain probability. We can retrieve *sample graphs* (see Definition 2.4) that are compatible with the probabilistic output graph.

We observe several serious drawbacks of the state-of-the-art in the fourth and the fifth categories. The fourth category leverages the semantics of edge probability to inject uncertainty to a given deterministic graph, converting it into a probabilistic one before publishing sample graphs. The state-of-the-art *(k,ϵ)-obfuscation* [11] has high impact on node privacy and not

good enough privacy-utility tradeoff while *RandWalk* [66] suffers from high lower bounds for utility despite its excellent privacy-utility tradeoff.

The fifth category revolves around the problem of graph publication under differential privacy (for a gentle introduction, see Section 2.1). 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 [21,89,100,101,105]. However, the problem is very challenging because of the huge output space of noisy graphs. A large body of existing schemes on differentially private release of graphs is not consistent with increasing privacy budgets as well as do not answer the question about the upper bounds of privacy budgets. Moreover, some of them have the scalability issue, usually a quadratic complexity.

Apart from publishing the whole graph, other paradigms for private computations over graphs are also of great interest. One such computation is *community detection* in graphs. Many complex networks expose a *mesoscopic* structure, i.e. they appear as a combination of components fairly independent of each other. These components are called communities, modules or clusters and the problem of how to reveal them plays a significant role in understanding the organization and function of complex networks. Over the last decade, a great number of algorithms for community detection (CD) have been proposed to address the problem in a variety of settings, such as undirected/directed, unweighted/weighted networks and non-overlapping/overlapping communities [38]. These approaches, however, are adopted in a non-private manner, i.e. a data collector (such as Facebook) knows all the contributing users and their relationships before running CD algorithms. The output of such a CD, in the simplest form, is a clustering of nodes. Even in this case, i.e. when only a node clustering (not the whole graph) is revealed, contributing users' privacy may still be put at risk.

Last but not least, we formulate an interesting problem of private link exchange. The problem is motivated by the fact that current OSNs mostly keep their data secret and in centralized manner. Conventionally, obfuscated data are released to public research and more exact data are reserved to internal research. Alternatively, researchers are allowed to crawl the underlying social graphs (and data) but with limited rates, leading to only partial views of the true social graphs. To overcome this roadblock, we may start from user perspective, the contributors of the OSNs. More precisely, if users cautiously collaborate with one another, they can exchange noisy friend lists with their neighbors in several rounds to get better views of the true social graph. We argue that each user trusts more in his friends than in strangers. The users therefore want the information about their friend lists to be reduced as the propagation distance increases. However, there exist fundamental questions about the feasibility of this model. The first question is how to define simple and effective privacy concepts for the link exchange processes. The second question comes from the high volume of link lists in exchange which may increase exponentially round after round. While storage and computation complexity may not be big problems, communication costs are non-trivial. Thus, efficient exchange protocols are needed.

1.2 Overview of the Problems

In this section, we outline the main solutions for solving the three problems addressed in the thesis: graph anonymization, private community detection and private link exchange.

1.2.1 Graph Anonymization

We study graph anonymization from two perspectives: uncertainty semantics and differential privacy.

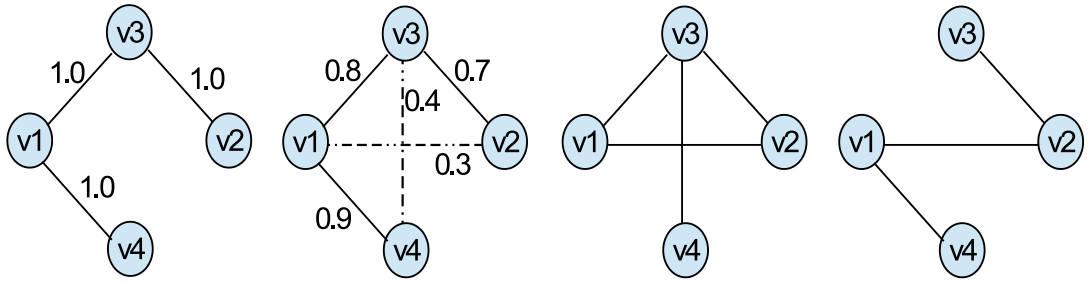


Figure 1.2: Uncertainty semantics of edges. From left to right: true graph, uncertain graph and two sample output graphs

The intuition behind the usage of uncertainty semantics to anonymize social graphs is illustrated in Fig. 1.2. The true graph is transformed to an uncertain graph by adding new edges (called potential edges) and assigning edge probabilities. Sample output graphs are produced from the uncertain by independently sampling edges. The edges probabilities may be computed explicitly as in [11] or implicitly via random walks [66]. The performance of different schemes are quantified by privacy and utility metrics. Privacy metrics may be an information-theoretic quantity such as (k, ϵ)-obfuscate [11] or degree-based incorrectness (Chapter 3). Commonly used utility metrics are degree-based and path-based statistics [12, 100, 107]. Clearly, the more potential edges and the more uncertain edges (i.e. edge probability approximates 0.5), the higher privacy (e.g. lower re-identification risks) but the lower utility (more distorted graph structure). All anonymization schemes aim at optimizing the tradeoff between these two competing forces.

Differential privacy [33] offers a formal definition of privacy with a lot of interesting properties: no computational/informational assumptions about attackers, data type-agnosticity, composability [63]. Differentially private algorithms relate the amount of noise to the *sensitivity* of computation. In the parlance of differential privacy, sensitivity indicates the change in the output of the computation according to a small change in the input (e.g. adding an edge to the input graph), so the term "differential" in the name. Lower sensitivity implies smaller added noise and vice versa. Because edges in simple undirected graphs are usually assumed to be 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 [89, 100] use *dK-series* [60] 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 [101] by graph spectral analysis. The most recent works *Density Explore Reconstruct* (DER) [21] and HRG-MCMC [105] even reduce the sensitivity of graph to $O(\log n)$. However, both of them incur quadratic complexity $O(n^2)$, limiting themselves to medium-sized graphs only.

1.2.2 Private Community Detection

Community structures are quite common in real networks. Social networks include community groups based on common location, interests, occupation, etc. Information networks (e.g. World Wide Web) have communities in the form of groups of hyperlinked pages having topical similarities. Metabolic networks have communities based on functional groupings of proteins. Citation networks form communities by research topic.

In non-private community detection, we are given the true graph and we can run any high-quality detection algorithms over it to obtain good clusterings of nodes. Good clusterings of

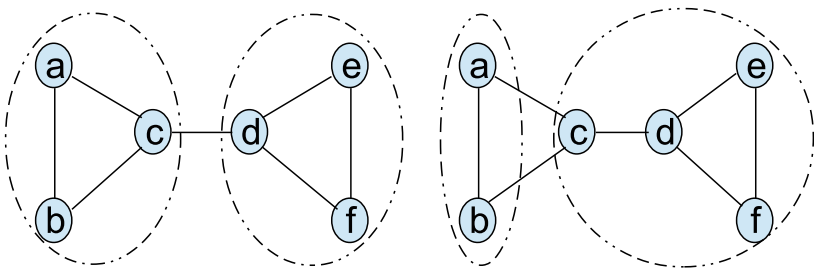


Figure 1.3: Community detection: a good clustering (left), a noisy clustering (right)

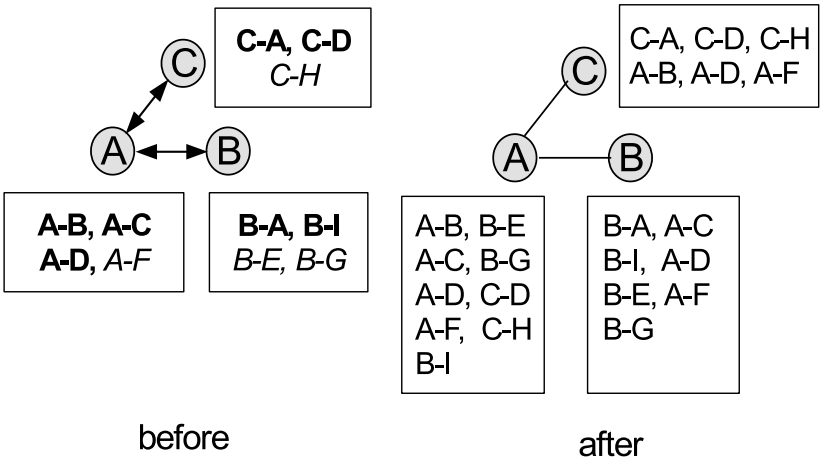


Figure 1.4: Private link exchange. True links are bold. Fake links are italic.

nodes may indicate high modularity, low conductance, grouping similar nodes, etc. [38]. Fig. 1.3 shows an example of community detection with a good clustering and a noisy clustering.

Private community detection means to do the detection in private manners. The data curator may give us a noisy graph over which we run detection algorithms and get noisy clusterings (i.e. clusterings with lower modularity or any clustering quality metrics considered). Alternatively, we are allowed to send to the data curator a number of queries on the true graph and get noisy answers. The number of queries depends on the privacy budget allocated to us (e.g. the value of ϵ in differential privacy). Based on the noisy answers we estimate a noisy clustering. In this thesis, we only consider private non-overlapping community detection under differential privacy.

1.2.3 Private Link Exchange

Social graphs are a valuable source for research on information societies but they are not published in clear-form due to serious privacy concerns. Instead, anonymized social graphs are published in various forms and provided to third party consumers. In the conventional client-server architecture of OSNs, the server keeps the whole social graph. To provide noisy sample graphs, the server can run any graph anonymization schemes on the social graph that it keeps. If the social graph is partitioned among several servers, we may resort to distributed anonymization, for example [97]. Alternatively, social networking sites provide APIs (Facebook ,Twitter) for data crawlers at limited rates and within privacy constraints (e.g. only public friend lists are available). Using this method, the data crawlers can collect friendship information and build a partial (local) view of the target social graph.

To overcome the constraints set by the service providers, we can start from *user perspective*, i.e. the contributors of OSNs. More precisely, if users cautiously collaborate with one another, they can exchange noisy friend lists with their neighbors in several rounds to get better views of the true social graph. Our ideas are based on the fact that user IDs are public (e.g. Facebook profiles are searchable [1]) but the friendships are not so, except when a user leaves his friend list in public mode. In this thesis, we introduce a different model in which nodes create noisy link lists and exchange with their neighbors in several rounds. The problem is unique in the sense that the disseminated data over the links are the links themselves. In the end, we have n local graphs where n is the number of nodes. We assume that all nodes are honest-but-curious, i.e. they follow the well-defined protocols and try to infer the true links among the noisy link lists sent to them in exchange steps. Fig. 1.4 depicts the basic idea of link exchange. Each node adds noise to its friend lists (bold font face) by creating fake links (italic font face). Then connected node pairs perform the link exchange and each node automatically removes duplicate links.

1.3 Contributions

The three problems are presented sequentially in Chapters 3, 4, 5 and 6. We make the following key contributions.

- **Chapter 3:** We propose a general model called *uncertain adjacency matrix* (UAM) for anonymizing graph via edge uncertainty semantics. The key property of this model is that the expected degrees of all nodes must be unchanged. We show the fitting of *(k,ϵ)-obf* [11] and *RandWalk* [66] into the model and then analyze their disadvantages. We introduce the *Variance Maximizing* (MaxVar) scheme that satisfies all the properties of the UAM. It achieves good privacy-utility tradeoff by using two key observations: nearby potential edges and maximization of total node degree variance via a simple quadratic program. Towards a fair comparison for anonymization schemes on graphs, this thesis describes a generic quantifying framework by putting forward the distortion measure (also called *incorrectness* in [92]) to measure the re-identification risks of nodes. As for the utility score, typical graph metrics [11] [107] are chosen. We conduct a comparative study of aforementioned approaches on three real large graphs and show the effectiveness of our gap-filling solutions.

- **Chapter 4:** We analyze the two radical 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 [52]. 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 Top-m-Filter (TmF) and the existing scheme EdgeFlip [69]. 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 also present the linear time scheme 1K-series which is based on the configuration model [73]. 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 privacy.

- **Chapter 5:** We analyze the major challenges of community detection under differential privacy [78]. We explain why techniques borrowed from k-Means fail and how the difficulty

of ϵ-DP recommender systems justifies a relaxation of ϵ. We design an input perturbation scheme LouvainDP that runs in linear time using the high-pass filtering technique from [26] and Louvain method [9]. We propose an algorithm perturbation scheme ModDivisive as a divisive approach by using the modularity-based score function in the exponential mechanism. We prove that modularity has small global sensitivity and ModDivisive also runs in linear time. We conduct a thorough evaluation on real graphs of different sizes and show the outperformance of LouvainDP and ModDivisive over the state-of-the-art.

- **Chapter 6:** We introduce a novel private link exchange problem as an alternative to social graph crawling and centralized anonymization of data. The problem is distributed and provides a privacy/utility tradeoff for all nodes. Our proposed problem is unique in the sense that the disseminated data over the links are the links themselves. We present two schemes for (α, β)-exchange protocol: Baseline and Bloom filter based. We protect the true links by adding fake links and require the propagation probability of links to be attenuated by distance from the links to reception nodes. We analyze the advantages and drawbacks of each scheme. We evaluate our proposed schemes on various synthetic graph models and draw a number of critical findings.

Publications Some parts of this thesis have been published and submitted to the following conferences and journals:

1. H. H. Nguyen, A. Imine, and M. Rusinowitch. A Maximum Variance Approach for Graph Anonymization (FPS 2014)
2. H. H. Nguyen, A. Imine, and M. Rusinowitch. Anonymizing Social Graphs via Uncertainty Semantics (ASIACCS 2015)
3. H. H. Nguyen, A. Imine, and M. Rusinowitch. Differentially Private Publication of Social Graphs at Linear Cost (ASONAM 2015)
4. H. H. Nguyen, A. Imine, and M. Rusinowitch. Network Structure Release under Differential Privacy (Transactions on Data Privacy, under 1st revision)
5. H. H. Nguyen, A. Imine, and M. Rusinowitch. Detecting Communities under Differential Privacy (WPES 2016)
6. H. H. Nguyen, A. Imine, and M. Rusinowitch. Private Link Exchange over Social Graphs (in preparation)

1.4 Structure of the Thesis

The rest of this thesis is organized as follows. Chapter 2 reviews prior work related to the three problems considered in this thesis. In Chapter 3, we deal with the uncertainty-based schemes for graph anonymization based on the publications 1 and 2. Chapter 4 continues the first problem, but using differential privacy instead. It is related to the papers 3 and 4. Chapter 5 solves the problem of differentially private community detection with two new methods LouvainDP and ModDivisive. The content of Chapter 5 is primarily from the paper 5. The problems of private link exchange are described in Chapter 6 which is prepared in the manuscript 6. Finally, Chapter 7 concludes with a summary of this dissertation, discuss our achievements as well as limitations, and outline some possible future research directions.

Chapter 2

State of the Art

Contents

In this chapter, we review the state-of-the-art related to the three problems to which we make significant contributions. We start with a gentle introduction to differential privacy, a privacy concept used extensively in Chapters 4 and 5. Then, we review the literature on graph anonymization in Section 2.2. We survey the main existing categories of anonymization techniques from k-anonymity to differential privacy. Section 2.3 highlights the taxonomy of community detection problems and popular algorithms to tackle them in various settings. We reserve Section 2.4 for graph aggregation and link exchange. In the end, we summarize the chapter in Section 2.5.

2.1 Differential Privacy: A Gentle Introduction

2.1.1 Motivation

Nowadays, data is produced and collected at a phenomenal rate. Analysis of huge data sources brings unprecedented benefits but also threatens the privacy of people. Usually, we encounter

a paradox of learning nothing about an individual while learning useful information about a population [35]. A lot of solutions were proposed for decades to solve this paradox, but each time a solution came out, new vulnerabilities were found. Different from previous paradigms, differential privacy provides us a promise: no matter what the adversary knows about you via auxiliary information sources, your *participation* in a given dataset will not be affected (so the name "differential" privacy).

Differential privacy is a formal privacy model initially developed for use on tabular data to offer strong privacy guarantees without depending on an adversary's background knowledge, computational power or subsequent behavior [33,35]. Because *absolute privacy* is impossible (see [32] and [35, Section 1.1]), differential privacy as an instance of *relative privacy* proves to be useful with many successful applications in a wide range of data analysis tasks and is found to have tight relations to other fields such as cryptography, statistics, complexity, combinatorics, mechanism design and optimization. A lot of beautiful results of differential privacy are systematically presented in [35] in which the authors explain all essential aspects of this privacy concept.

The underlying principle of differential privacy is that given two databases D and D' such that $D' = D \cup \{X\}$,i.e. D and D' differ only by a single item, the probability distributions on the results of D and D' under differential privacy will be "essentially the same". More formally,

Definition 2.1. *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for any two neighboring datasets D and D', and for any output $O \in Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(D) \in O] \leq e^{\epsilon} Pr[\mathcal{A}(D') \in O] + \delta$$

If $\delta = 0$, we have the definition of ϵ-differential privacy which is proved stricter than (ϵ, δ)-differential privacy [31], i.e. the (ϵ, δ)-differential privacy requires less distortion than ϵ-differential privacy.

Bear in mind that, as a relative privacy, differential privacy merely ensures that one's participation in a dataset will not be disclosed. It is very possible that conclusions drawn from the dataset may reflect statistical information about an individual. For example, given a dataset of 100 individuals, 80 of them have a certain property P. Noisy answers to the queries of the dataset size and the numbers of users having the property P are, for example, 101.3 and 78.6 respectively. From these noisy counts, an analyst (as well as an attacker) can estimate that any user has the property P with probability of 77.6%, very close to the true statistics 80%. However, this is a statistical information, not related to the participation of any individual to the dataset as guaranteed by differential privacy.

2.1.2 Basic Techniques for Differential Privacy

There are two common settings for releasing data differential privacy: *interactive* versus *non-interactive* as illustrated in Fig. 2.1. In non-interactive setting, given a privacy budget ϵ, the data curator publishes a noisy dataset $\tilde{D}$ and shuts down D. The users can perform any mining operations on $\tilde{D}$. In interactive settings, the users are allowed to submit a number of queries (q_i, ϵ_i) where q_i is the query over D and ϵ_i is the budget of q_i such that $\sum_i \epsilon_i \leq \epsilon$. It means after a query, the privacy budget is reduced and when it reaches zero, no more queries are allowed.

Laplace mechanism [34] and *Exponential mechanism* [63] 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 $\Delta f = \max_{D,D'} ||f(D) - f(D')||_1$ where the maximum is taken over all pairs of neighboring D, D'. Given a function f and a privacy budget ϵ, the noise is drawn from a

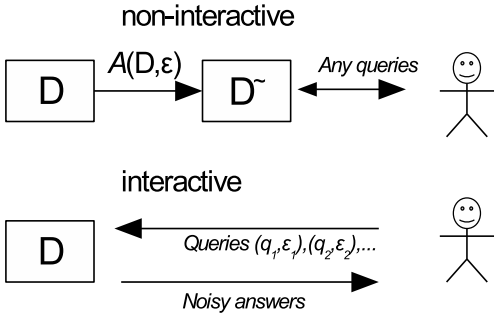


Figure 2.1: Interactive vs. Non-interactive settings

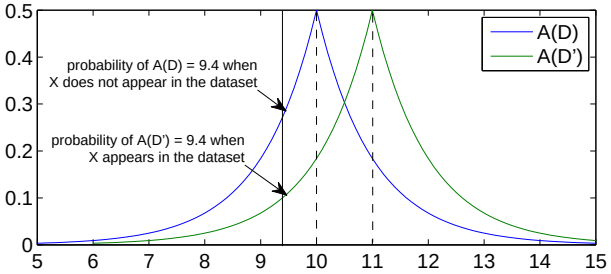


Figure 2.2: Illustration of differential privacy via Laplace mechanism

Laplace distribution $Lap(\lambda) = p(x|\lambda) = \frac{1}{2\lambda}e^{-|x|/\lambda}$ where $\lambda = \Delta f/\epsilon$. Recently, 2-dimensional Laplace distribution is used to construct geo-indistinguishability mechanisms [3, 20] in location privacy. *Geometric mechanism* [40] 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 $\alpha = \exp(-\epsilon)$.

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 (see Section 4.2 for more detail).

Fig. 2.2 shows an example of differential privacy via Laplace mechanism. Assuming that a counting query on dataset D returns 10 if user X does not appear in it and returns 11 otherwise (i.e. the case of dataset D'). If true results are returned, the participation of X is revealed by a simple difference attack. To hide this sensitive information, randomized results would be preferred. Assuming that $\epsilon = 1$, the randomized result via Laplace mechanism will be $\mathcal{A}(D) = 10 + Lap(1)$ because the global sensitivity Δf is 1 for this example. Similarly, $\mathcal{A}(D') = 11 + Lap(1)$. The ratio between the probabilities of $\mathcal{A}(D)$ and $\mathcal{A}(D')$ at any possible values (e.g. 9.4 as in Fig. 2.2) is bounded in $[e^{-1}, e]$.

2.1.3 Compositions Make Differential Privacy Programmable

Not only formally defined, differential privacy is also equipped with two powerful composition properties: serial composition and parallel composition (see Section 4.2 for more detail). Serial composition means that when we run a series of randomized algorithms $\mathcal{A}_i(\epsilon_i)$ on the same dataset D, the total privacy budget ϵ will be the sum of all ϵ_i. Parallel composition states that when we run a series of randomized algorithms $\mathcal{A}_i(\epsilon_i)$ on the *disjoint* subsets D_i of D, the effective privacy budget will be the maximum of $\{\epsilon_i\}$.

These compositions allow many complex differentially private algorithms to break down

to composable steps in which Laplace and exponential mechanisms play the role of building blocks. Most of the cases, the proof of ϵ-DP for a given algorithm is obtained automatically. The remaining task of algorithm designers is to reduce the variance of randomized outputs for better utility. To our knowledge, none of the previous privacy concepts like k-anonymity [95], l-diversity [59], t-closeness [54] and their variants possesses these composition properties.

2.2 Graph Anonymization

There is a vast literature on graph perturbation with many good surveys [2,104] covering multiple aspects of graph anonymization: privacy risks of social data publication, de-anonymization attacks using background knowledge, information loss by different anonymity levels and privacy-preserving mechanisms.

As mentioned in Section 1.1.2, there are five main categories of anonymization techniques. We review the five main categories in the following section along with a discussion on privacy and utility metrics.

2.2.1 Anonymization via Randomization

The anonymization methods in this category perturb a graph by randomly adding, deleting or switching of edges [13,46,107,109]. For social networks, two edge-based randomization strategies have been commonly applied [108].

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

- *Rand Switch*: randomly switch a pair of existing edges (t, w) and (u, v) (provided that edge (t, v) and edge (u, w) do not exist in G) to (t, v) and (u, w), and repeat this process for k times. The degree of each node is preserved by this strategy.

In [108], Ying and Wu investigate the link privacy under Rand Add/Del and Rand Switch by quantifying the difference between attackers' prior and posterior beliefs about the existence of an edge. They relate the existence of an edge to the similarity measure between the two nodes of the edge. Although the random graph editing strategies could mitigate the re-identification attacks, they do not guarantee that the randomized graphs satisfy k-anonymity [95]. Moreover, by arbitrary modification of the edges, the strategies ignore the fact that privacy should be guaranteed for all users. Instead, they benefit only random users.

Ying and Wu [107] argue that Rand Add/Del and Rand Switch have impact on both real and spectral characteristics of a graph. The spectrum of a network corresponds to the set of eigenvalues λ_i $(1 \leq i \leq n)$ of an adjacency matrix derived from the graph. These eigenvalues are closely linked to several topological characteristics of a graph including diameter, the existence of consistent clusters, lengthy paths and bottlenecks, and randomness of the graph. Based on these two observations, they proposed a spectrum preserving randomization approach called *Spctr Add/Del* and *Spctr Switch*.

An advantage of randomization strategies is that many features can be accurately reconstructed from the released randomized graph, e.g. spectrum-based reconstruction [102]. Let λ_i ($\tilde{\lambda}_i$) be A's ($\tilde{A}$'s) i-th largest eigenvalue in magnitude whose eigenvector is $\mathbf{x}_i$ ($\tilde{\mathbf{x}}_i$). Then, the rank l approximations of A and $\tilde{A}$ are respectively given by $A = \sum_{i=1}^{l} \lambda_i \mathbf{x}_i \mathbf{x}_i^T$ and $\tilde{A} = \sum_{i=1}^{l} \lambda_i \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T$.

Bonchi et al. [13] proposed two algorithms called a *random sparsification* and a *random perturbation*. Random sparsification works as follows. Given $G = (V, E)$, the algorithm selects a probability $p \in [0, 1]$ and then removes edges independently with probability $1 - p$. Random perturbation transforms $G = (V, E)$ into its obfuscated form by first removing edges with a probability $p \in [0, 1]$ and then adding an edge $e \in \binom{V}{2} \setminus E$ with probability $q = \frac{|E|.p}{\binom{|V|}{2}-|E|}(1 - p)$. Bonchi et al. used the Shannon entropy to quantify the level of anonymity.

2.2.2 K-anonymization Approaches

For a successful attack targeted to a node, an attacker analyzes topological features of the target node based on his background knowledge. To quantify this kind of privacy breach, Hey et al. [46] proposed a general model for social graphs as follows.

Definition 2.2. *Let $G = (V, E)$ be a given graph. A structural query Q is a query on the local structure of the graph around a node (e.g. node degree, degrees of node's neighbors, etc.).*

A node v is k-candidate anonymous with respect to a structure query Q if there exist at least $k - 1$ other nodes in V that match query Q. In other words, $|cand_Q(v)| \geq k$ where $cand_Q(v) = \{y \in V | Q(y) = Q(v)\}$. A graph satisfies k-candidate anonymity with respect to Q if all the nodes are k-candidate anonymous with respect to Q.

Three types of queries (node refinement queries, subgraph queries, and hub fingerprint queries) were presented and evaluated on the naively anonymized graphs [46].

K-anonymization approaches provide anonymity through editing nodes and edges of a graph deterministically. Many methods have been proposed to prevent node re-identification based on the k-anonymity concept. These methods differ in the types of the structural background knowledge that an attacker may use. In [55], Liu and Terzi assumed that the adversary only knows the degree of a target node and proposed *k-degree* anonymization. In [112], Zhou and Pei assumed one specific subgraph constructed by the immediate neighbors of a target node is known and devised *k-neighborhood* scheme. In [113], Zou et al. considered all possible structural information around the target and proposed *k-automorphism* to guarantee privacy under any structural attack. Wu et al. [103] proposed *k-symmetry*, a concept similar to k-automorphism and made a graph k-symmetric by adding fake nodes. Cheng et al. [23] proposed *k-isomorphism* with the concept of *k-security*.

Definition 2.3. *Let $G = (V, E)$ be a given graph with unique node information $I(v)$ for each node $v \in V$. Each node $v \in V$ is linked to a unique individual $U(v)$. Let G_k be the anonymized graph of G. G_k satisfies k-security with respect to G if for any two target individuals A and B with corresponding Neighborhood Attack Graphs G_A and G_B that are known by the adversary, the following two conditions hold*
- NodeInfo security: the adversary cannot determine from G_k and $G_A(G_B)$ that $A(B)$ is linked to $I(v)$ for any node v with a probability of more than $1/k$
- LinkInfo security: the adversary cannot determine from G_k, G_A and G_B that A and B are linked by a path of a certain length with a probability of more than $1/k$

Finally, Tai et al. [96] proposed *k^2-degree*. The algorithm anonymizes a graph such that an adversary with prior knowledge of the degrees of two adjacent nodes will not be able to mount a friendship attack on the graph. A graph $\tilde{G}$ is k^2-degree anonymous if, for every node with an incident edge of degree pair (d_1, d_2) in G, there exist at least $k - 1$ other nodes, such that each of the $k - 1$ nodes also has an incident edge of the same degree pair. They proposed an Integer

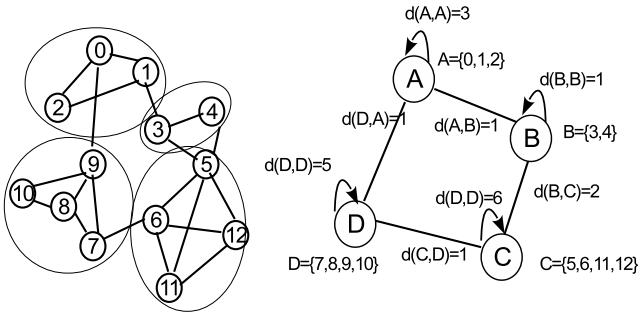


Figure 2.3: Generalization strategy

Programming formulation to find optimal solutions in small-scale networks and also presented an efficient heuristic approach for anonymizing large-scale social networks against friendship attacks.

2.2.3 Anonymization via Generalization

Rather than modifying the graph structure as done in k-anonymity and randomization approaches, generalization strategy [18, 46, 97] transforms the graph into super-nodes and super-edges to mitigate the node re-identification attack. Fig. 2.3 illustrates the generalization process with 4 super-nodes and 8 super-edges (including 4 self-loops). The collapse of nodes and edges into super-nodes and super-edges induces edge probabilities. For example, the probability of the edge (1,3) (in the graph on the right of Fig. 2.3) is $\frac{1}{6}$, equal to the weight of the super-edge (A,B) divided by the product of A's size and B's size. Similarly, the probability of the edge (7,8) becomes $\frac{5}{6}$.

Hay et al. [46] proposed a generalization approach for a simple unlabeled graph. A simulated annealing is used to search for the approximate optimal super-nodes in [46]. The optimality is estimated via a maximum-likelihood approach. The case of nodes having attributes (e.g., demographic information) is considered in Campan and Truta [18]. A batch-based greedy approach that iteratively generalizes graph nodes is used. In every iteration, a seed node is selected and a new super-node is created with a seed node within it. Then the algorithm selects, based on attributes and neighborhood similarity, $k-1$ other nodes and add them to the super-node. This process continues until the entire graph is anonymized.

Tassa and Cohen [97] extended the problem [18] by proposing a sequential clustering algorithm to anonymize a graph with higher utility than [18]. The algorithm begins with partitioning the nodes into $n/(\alpha k)$ clusters randomly, where n is the number of nodes in the graph, k is an integer and α is a parameter set by the user. The algorithm goes over the n nodes in a cyclic manner and moves nodes from one cluster to the most appropriate cluster that would possibly minimize information loss.

Although the methods in [18, 46, 97] resist node re-identification attack, finding globally optimal partition of the graph is an intractable problem. Moreover, the structural information in the super-nodes is reduced to the number of edges in the self-loops. This requires the analyst to sample a graph from all likely super-nodes, which inevitably leads to sampling errors. As the generalization approaches only publish the clustered graph with super-nodes and super-edges, the utility of the original network is significantly impacted.

2.2.4 Probabilistic Approaches

The methods in this category convert deterministic graph into an uncertain graph by leveraging the semantics of edge probability (see Fig. 1.2). Note that the schemes in Sections 2.2.1 and 2.2.3 are also probabilistic but the edge probabilities are not explicitly defined on edges as in this section. In other words, randomization and generalization schemes directly output noisy sample graphs, omitting the intermediary uncertain representations in the form of edge probabilities. Let $\mathcal{G} = (V, E, p)$ be an uncertain graph, where V is the set of nodes, E is the set of edges, $p : E \rightarrow (0, 1]$ is the function that assigns an existence probability to each edge. A sample perturbed graph is denoted as $\tilde{G} = (V, \tilde{E})$. The common assumption is on the *independence* of edge probabilities. Below is the definition of sample graphs which is relevant to the approaches in this section and Section 2.2.3.

Definition 2.4. *Let $\mathcal{G} = (V, E, p)$ be an uncertain graph, where V is the set of nodes, E is the set of edges, $p : E \rightarrow (0, 1]$ is the function that assigns an existence probability to each edge. A sample graph $G = (V, E_G)$ is compatible with $\mathcal{G}$ if E_G consists of edges $e \in E$ sampled with probability $p(e)$.*

Several approaches have been developed to address the above problem. Bonchi et al. [13] proposed the *k-obfuscation* privacy model to guarantee that an adversary cannot map a node identity in the perturbed graph to the node of its original graph. K-obfuscation is defined by means of the entropy of the probability distributions that are induced on the nodes of the perturbed graph. Also, it uses the random sparsification (Section 2.2.1) to generate the obfuscated graph. A perturbed graph $\tilde{G} = (U = V, \tilde{E})$ is said to observe k-obfuscation provided that for every node $v \in V$, the entropy of the random variable H_v that denotes the probability that u is the image of v in $\tilde{G}$ over U is at least $\log k$.

Boldi et al. [11] proposed (k, ϵ)-obfuscation to the problem of k-obfuscation where $k \geq 1$ denotes the preferred level of anonymization and $\epsilon \geq 0$ is the parameter for tolerance. Chapter 3 discusses this method in more detail and presents our scheme *MaxVar*, an improvement of (k, ϵ)-obfuscation using several key observations and a formulation based on quadratic optimization.

Apart from explicitly assigning probabilities to edges, we observe an *implicit* realization of edge uncertainty semantics via random-walks in [66] (see Chapter 3). Mittal et al. [66] proposed an edge-rewiring scheme (called *RandWalk* afterwards) by random-walks at distance t. Given the true graph $G = (V, E)$, the sample noisy graph $\tilde{G} = (V, \tilde{E})$ is generated as follows. For each node $u \in V$, we iterate all its neighbors v and perform a walk of length $t - 1$. Assuming that the walk stops at node w, we add the edge (u, w) to $\tilde{G}$. In Chapter 3, by introducing the uncertain adjacency matrix (UAM), we prove that RandWalk, (k, ϵ)-obfuscation and MaxVar fit into UAM.

2.2.5 Differentially Private Approaches

In the context of graph publication, two neighboring datasets D and D' are replaced by two neighboring graphs G and G'.

- If G and G' differ in only one edge, we have the concept of *edge differential privacy*. There exists a lot of research on edge differential privacy [21, 45, 51, 69, 79, 89, 100, 101, 105], just to cite a few.
- If G and G' differ in only one node and its adjacent edges, we have the concept of *node differential privacy*. Node differential privacy is much harder to achieve than edge differential privacy. Only a few works [8, 14, 22, 49, 85] deal with node differential privacy.

Edge Differential Privacy The problem of graph publication under edge 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 [77] and EdgeFlip [69] 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 [89, 100], Kronecker graph model [65], graph spectral analysis [101], DER [21], HRG-MCMC [105] and ERGM (Exponential Random Graph Model) [58].

EdgeFlip [69] applies Randomized Response Technique (RRT) [35, 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 4.4.

1K-series and 2K-series [89, 100] use dK-series [60] to summarize the graph into a distribution of degree correlations. 2K-series was expected to give better noisy output graphs but the utility results [89, 100] are not good enough, not to say the huge values of ϵ used (up to thousands). The similar problem happened to graph spectral analysis [101]. Kronecker graph model [65] privately estimates the initiator matrix Θ using smooth sensitivity [79] 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 4.5. ϵ-DP graph release is also mentioned in ERGM [58, 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.

Node Differential Privacy Kasiviswanathan et al. [49] developed several techniques for designing differentially node-private algorithms, as well as a methodology for analyzing their accuracy on realistic networks. The main idea behind their techniques is to "project" (in one of several senses) the input graph onto the set of graphs with maximum degree below a certain threshold. They presented two different techniques. First, they designed tailored projection operators that have low sensitivity and preserve information about a given statistic and applied to releasing the number of edges in a graph, and counts of small subgraphs such as triangles, k-cycles, and k-stars in a graph, and certain estimators for power law graphs. Second, they analyzed the "naive" projection that simply discards high-degree nodes in the graph with an efficient algorithms for bounding the "local sensitivity" of this projection. Using this, they derived a generic, efficient reduction that allows them to apply any differentially private algorithm for bounded-degree graphs to an arbitrary graph. They used this to design algorithms for releasing the entire degree distribution of a graph.

Blocki et al. [8] introduced the notion of *restricted sensitivity* as an alternative to global and smooth sensitivity to improve accuracy in differentially private data analysis. It takes advantage of any beliefs about the dataset that a querier may have, to quantify over a restricted class of datasets. Blocki et al. presented two main results. First, they showed that for every function $f : \mathcal{G}_{n,D} \to \mathbb{R}$, there exists an extension $g : \mathcal{G}_n \to \mathbb{R}$ that agrees with f on $\mathcal{G}_{n,D}$ and that has global sensitivity $\Delta g = \Delta D f$. The resulting function needs not to be computable efficiently. Second, for the specific definition of graphs of bounded degree, they exhibited efficient ways of constructing new queries using different projection-based techniques.

Chen and Zhou [22] proposed a novel differentially private mechanism, named *recursive mechanism*, to release an approximation to linear statistics of some positive relational algebra calculation over a sensitive database. Their definition of sensitive database takes into account the impact of a participant on multiple tuples, so it is relevant to node differential privacy. They modeled queries to a graph by positive Boolean expressions (not using negation) in which each participant is a Boolean variable. They also introduced several concepts of empirical sensitivity (local, global and universal ones). Their 3-step recursive mechanism can answer any monotonic query on any sensitive database. It is formulated as a linear program.

2.2.6 Privacy Metrics

We now survey commonly used notions of privacy metrics. *Min entropy* [93] quantifies the largest probability gap between the posterior and prior over all items in the input dataset. K-anonymity has the same semantics with the corresponding min entropy of $\log_2 k$. So we say k-anonymity based perturbation schemes belong to min entropy. *Shannon entropy* argued in [13] and [11] is another choice of privacy metrics. Higher Shannon entropy means better privacy protection because an attacker has to confront higher uncertainty about the true dataset. The third metrics that we use in this thesis is the *incorrectness* measure from location privacy [92]. Given the prior information (e.g. node degree in the true graph) and the posterior information harvested from the anonymized output, incorrectness measure is the number of incorrect guesses made by the attacker. This measure gauges the distortion caused by the anonymization algorithm. *Multiplicative ratio* ϵ, a.k.a privacy budget, in differential privacy [35] places constraints on the probabilities of noisy outputs when any item decides to join or not to join the dataset. Lower ϵ indicates stricter constraints, leading to flatter distributions on the output space, i.e. harder to reveal the true input. Moreover, the composability properties bring ready-to-use building blocks for designing differentially private algorithms.

2.2.7 Utility Metrics

To evaluate the utility of anonymized graphs, a lot of graph metrics (properties) are introduced in the literature (e.g. [73]). In this thesis, we consider three groups of graph metrics. The first group is based on the degree sequence of nodes such as number of edges, average degree, maximal degree, degree variance, power-law exponent and degree distribution. The second group is about shortest paths such as average distance, connectivity length, effective diameter, diameter and distance distribution. Finally, other important graph metrics like clustering coefficient and graph cut queries are used. Detailed definitions of these metrics are presented in Sections 3.5.2 and 4.7.1.

2.3 Community Detection

2.3.1 Non-Private Community Detection

For recent comprehensive surveys, we refer to [27, 38, 61]. Detailed discussions and comparisons on quality measures for community detection can be found in [19, 28, 106]. In this section, we discuss modularity function and several classes of detection techniques.

Newman and Girvan [75] proposed *modularity* as a quality measure for network clustering. It is based on the idea that a random graph is not expected to have a modular structure, so the possible existence of clusters is revealed by the comparison between the actual density of

edges in a subgraph and the density one would expect to have in the subgraph if the nodes of the graph were connected randomly (the null model in parlance of hypothesis testing). The modularity Q is defined as

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right] \tag{2.1}$$

where n_c is the number of clusters, l_c is the total number of edges joining nodes of community c and d_c is the sum of the degrees of the nodes of c.

Many methods for optimizing modularity have been proposed over the last ten years, such as agglomerative greedy [25, 74], simulated annealing [64], random walks [82], statistical mechanics [86], label propagation [84], InfoMap [87] or Louvain method [9], just to name a few.

To find a partition that provides the maximum value of modularity is an NP-complete problem [15]. Many greedy heuristics have therefore been proposed. The first algorithm devised to maximize modularity was a greedy method of Newman [74]. It is an agglomerative hierarchical clustering method, where groups of nodes are successively merged to form larger communities such that modularity increases after the merging. One starts from n clusters, each containing a single node. Edges are not initially present, they are added one by one during the procedure. However, the modularity of partitions explored during the procedure is always calculated from the full topology of the graph. The complexity of Newman method is $O((m+n)n)$, or $O(n^2)$ on a sparse graph. Clauset et al. [25] proposed a faster greedy by storing a matrix containing only the values of the communities, i.e. the modularity changes when joining the communities i and j. This operation can be performed more efficiently by using data structures for sparse matrices, like max-heaps, which rearrange the data in the form of binary trees.

Medus et al. [64] presented a simulated annealing method to study the community structure in networks based on the search for that partition that maximizes the value of modularity. Simulated annealing is a probabilistic procedure for global optimization used in different fields and problems. It is a generalization of the well known Markov Chain Monte Carlo (MCMC) procedure. Its standard implementation combines two types of "moves": local moves, where a single node is shifted from one cluster to another, taken at random; global moves, consisting of mergers and splits of communities. Global moves reduce the risk of getting trapped in local minima and they have proven to lead to much better optima than using simply local moves. In practical applications, one typically combines n^2 local moves with n global ones in one iteration. The method can potentially come very close to the true modularity maximum, but it is slow.

Latapy and Pons [82] observed that random walks on a graph tend to get "trapped" into densely connected parts corresponding to communities. They proposed a measure of similarities between nodes based on random walks which has several important advantages: it captures well the community structure in a network, it can be computed efficiently, and it can be used in an agglomerative algorithm to compute efficiently the community structure of a network. Their algorithm called *WalkTrap* runs in time $O(mn^2)$ and space $O(n^2)$ in the worst case, and in time $O(n^2 \log n)$ and space $O(n^2)$ in most real-world cases.

Raghavan et al. [84] have designed a simple and fast method based on *label propagation*. Nodes are initially given unique labels (e.g. their node labels). At each iteration, a sweep over all nodes, in random sequential order, is performed: each node takes the label shared by the majority of its neighbors. If there is no unique majority, one of the majority labels is picked at random. In this way, labels propagate across the graph: most labels will disappear, others will dominate. The process reaches convergence when each node has the majority label of its neighbors. Communities are defined as groups of nodes having identical labels at convergence.

By construction, each node has more neighbors in its community than in any other community.

The modular structure of a graph can be considered as a compressed description of the graph to approximate the whole information contained in its adjacency matrix. Based on this idea, Rosvall and Bergstrom [87] proposed *InfoMap* algorithm which uses the probability flow of random walks on a network as a proxy for information flows in the real system and decomposes the network into modules by compressing a description of the probability flow. The result is a map that both simplifies and highlights the regularities in the structure and their relationships.

Since its introduction in 2008, *Louvain method* [9] became one of the most cited methods for the community detection task. It optimizes the modularity by a bottom-up folding process. The algorithm is divided in passes each of which is composed of two phases that are repeated iteratively. Initially, each node is assigned to a different community. So, there will be as many communities as there are nodes in the first phase. Then, for each node i, the method considers the gain of modularity if we move i from its community to the community of a neighbor j (a *local change*). The node i is then placed in the community for which this gain is maximum and positive (if any), otherwise it stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first pass is then complete.

2.3.2 Private Community Detection

To motivate the problem of private community detection, we show a simple example of how different clusterings of nodes may give hints to attackers. In Fig. 2.3, thirteen nodes are clustered in four communities {{0,1,2}, {3,4}, {5,6,11,12}, {7,8,9,10}}. Assuming that a new edge (0,3) is added, the new clustering of nodes may become {{0,1,2,3}, {4,5,6,11,12}, {7,8,9,10}}. By observing the clusterings before and after the addition of the edge (0,3), an attacker infers that there might have been new edge(s) between 3 and the nodes in the community {0,1,2}.

The problem of community detection under differential privacy is quite new and only mentioned in the recent work [69] where Mülle et al. use a randomized response technique [35, Section 3.2] to perturb the input graph so that it satisfies differential privacy before running the conventional CD algorithms. This technique (we call it *EdgeFlip* afterwards) is classified as *input perturbation* in differential privacy literature (the other two categories are *algorithm perturbation* and *output perturbation*). Similarly, ϵ-DP schemes for graph release in Section 2.2.5 are applicable.

By maximizing the modularity, Louvain method is based on *edge counting* metrics, so it fits well with the concept of *edge differential privacy* (Section 2.2.5). One of the most recent methods *SCD* [83] is not chosen because it is about maximizing Weighted Community Clustering (WCC) instead of the modularity. WCC is based on *triangle counting* which has high global sensitivity (up to $O(n)$) [110]. Moreover, SCD pre-processes the graph by removing all edges that do not close any triangle. This means all 1-degree nodes are excluded and form singleton clusters. The number of output clusters is empirically up to $O(n)$. Thus, we target effective ϵ-DP schemes for community detection using edge differential privacy and modularity optimization.

A recent paper by Campan et al. [17] studies whether anonymized social networks preserve existing communities from the original social networks. The considered anonymization methods are k-anonymity [18] and k-degree method [55] which belong to input perturbation. In this thesis, we only examine differentially private schemes in both categories: input and algorithm perturbations.

2.4 Graph Aggregation and Link Exchange

Epidemic spreading [68, 81] is the most related to our work. In [81], Pastor-Satorras et al. study the spreading of infections on scale-free (power-law) networks via the susceptible-infected-susceptible (SIS) model [5]. They find the absence of an epidemic threshold ($\lambda_c = 0$) and its associated critical behavior when the number of nodes goes to infinity using mean-field approximation. Moreno et al. [68] provide a detailed analytical and numerical study of susceptible-infected-removed (SIR) on Watts-Strogatz (WS) small-world model and Barabási-Albert (BA) scale-free model. WS graphs with exponentially distributed degrees can be considered as a *homogeneous* model in which each node has the same expected number of links. WS graphs have finite epidemic thresholds. On the contrary, BA graphs with power-law distributed degrees are *heterogeneous* and they expose the weaker resistance to epidemics starting on highly connected nodes.

Giakkoupis et al. [41] introduce a distributed algorithm RIPOSTE for disseminating information in a social network that preserves privacy of nodes. Whenever the information reaches a node, the node decides to either forward the information to his neighbors or drop it. RIPOSTE uses two global parameters δ and λ and satisfies differential privacy by applying Randomized Response Technique (RRT) [35]. Our work is also a form of information dissemination over graphs but it spreads a large number of links, not a single item.

Gossip-based protocols [39, 99] aim at providing alternatives to network-level multicast with good scalability and reliability properties. In these protocols, message redundancy for high reliability is ensured by the fact that each member forwards each message to a set of other, randomly chosen, group members. Ganesh et al. [39] propose a fully decentralized and self-configuring protocol SCAMP that provides each member with a partial view of group membership. As the number of participating nodes changes, the size of partial views automatically adapts to the value required to support a gossip algorithm reliably. CYCLON [99] is a protocol for construction of reliable overlay networks. It is targeted to overlays that have low diameter, low clustering, highly symmetric node degrees and highly resilient to massive node failures. These properties are satisfied by random graphs. CYCLON employs an enhanced shuffling operation in which nodes select neighbors for cache exchange based on their age.

By exchanging noisy link lists, our schemes are related to distributed graph anonymization [18, 97]. However, rather than producing a single global anonymized graph as in [97], link exchange protocols result in multiple local outputs. In addition, link exchange operates at finest-grained level (node-level) whereas previous works consider a small number of data holders who manage disjoint sets of nodes.

The idea of adding fake links to hide true links appears in a number of earlier studies, e.g. [76, 91]. Shokri et al. [91] propose a method for privacy preservation in collaborative filtering recommendation systems. They develop a model where each user stores locally an offline profile on his own side, hidden from the server, and an online profile on the server from which the server generates the recommendations. Each user arbitrarily contacts other users over time, and modifies his own offline profile through aggregating ratings from other users. The more ratings a user aggregates, the higher privacy he gets but the lower accuracy in recommendations he obtains. Our work [76] presents a centralized graph anonymization scheme based on edge uncertainty semantics. Fake edges are added to probabilistically hide true edges. We consider distance-2 fake edges to keep higher utility (see Chapter 3).

2.5 Summary and Discussion

The above discussion on related work clarifies how we can improve in graph anonymization, private community detection and private link exchange. Uncertainty-based methods for graph anonymization [11, 66] have some limitations in privacy/utility tradeoff and lack a theoretical analysis model. Our objective is to formalize the probabilistic graph anonymization through an analytical model and design novel schemes to fill the existing gap.

The prevalence of differential privacy makes it appealing in graph anonymization and private community detection. We showed that the existing model-based schemes for graph anonymization [21, 100, 101, 105] are not consistent for large privacy budgets and/or incur quadratic complexity. These drawbacks have left the room for better direct publication schemes that are both efficient and consistent. Using differential privacy to detect communities in social graphs raises considerable challenges. We found the existing schemes ineffective, especially those based on ϵ-DP graph release. That is why we explore other directions such as using Louvain method on noisy super graphs or top-down privately sampling highly modular partitions of nodes.

To advocate the users' rights on their private information, we propose a novel link exchange problem that allows the users to build their own local views of the underlying social graph. The existing work on epidemic spreading of information, gossip-based protocols and distributed graph anonymization are related but not totally in the same vein as our model. Via the private link exchange, we would like to bring up a different mechanism for private and democratic information exchange in online social networks.

In the next chapter, we present our first main contribution which is about social graph anonymization via uncertainty semantics.

Chapter 3

Anonymizing Social Graphs via Uncertainty Semantics

Contents

3.1 Introduction

Graphs represent a rich class of data observed in daily life where entities are described by vertices and their connections are characterized by edges. With the emergence of increasingly complex networks [73], the research community requires large and reliable graph data to conduct in-depth studies. However, this requirement usually conflicts with privacy rights 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 [4] [46]. Therefore, many graph anonymization schemes have been proposed [23, 55, 96, 103, 112, 113].

Given an unlabeled undirected graph, the existing anonymization methods fall into four main categories. The first category includes *random* addition, deletion and switching of edges to prevent the re-identification of nodes or edges. The methods in the second category provide k-anonymity [95] by *deterministic* edge additions or deletions, assuming attacker's background knowledge regarding certain properties of its target nodes. The methods in the third category assign edge probabilities to add uncertainty to the true graph. The edges probabilities may be computed explicitly as in [11] or implicitly via random walks [66]. Finally, the fourth class of techniques, *generalization*, cluster nodes into super nodes of size at least k. Note that the last two classes of schemes induce *possible world* models, i.e., we can retrieve sample graphs that are consistent with the anonymized output graph.

The third category is the most recent class of methods which leverage the semantics of edge probability to inject uncertainty to a given deterministic graph, converting it into an uncertain one. Most of schemes in this category are scalable, i.e. runnable on million-scale graphs or more. As an example, Boldi et al. [11] introduced the concept of *(k,ϵ)-obfuscation* (denoted as (k, ϵ)-obf), where $k \geq 1$ is a desired level of obfuscation and $\epsilon \geq 0$ is a tolerance parameter. However, the pursuit for minimum standard deviation σ in (k,ϵ)-obf has high impact on node privacy and high privacy-utility tradeoff. Edge rewiring method based on random walks (denoted as *RandWalk*) in [66] also introduces uncertainty to edges as we show in section 3.3. This scheme suffers from high lower bounds for utility despite its excellent privacy-utility tradeoff.

Motivated by *(k,ϵ)-obf* and *RandWalk*, we propose in this chapter a general model for anonymizing graphs based on edge uncertainty. Both (k,ϵ)-obf and RandWalk display their fitting into the model. We point out disadvantages in (k,ϵ)-obf and RandWalk, the tradeoff gap between them and present several elegant techniques to fill this gap. Finally, to support fair comparisons, we develop a new tradeoff quantifying framework using the concept of *incorrectness* in location privacy research [92].

Our contributions are summarized as follows:

- We propose a general model called *uncertain adjacency matrix* (UAM) for anonymizing graph via edge uncertainty semantics (Section 3.2). The key property of this model is that expected degrees of all nodes must be unchanged. We show the fitting of *(k,ϵ)-obf* and *RandWalk* into the model and then analyze their disadvantages (Section 3.3).

- We introduce the *Variance Maximizing* (MaxVar) scheme (Section 3.4) that satisfies all the properties of the UAM. It achieves good privacy-utility tradeoff by using two key observations: nearby potential edges and maximization of total node degree variance via a simple quadratic program.

- Towards a fair comparison for anonymization schemes on graphs, this chapter describes a generic quantifying framework (Section 3.5) by putting forward the distortion measure

(also called *incorrectness* in [92]) to measure the re-identification risks of nodes. As for the utility score, typical graph metrics [11] [107] are chosen.

- We conduct a comparative study of aforementioned approaches on three real large graphs and show the effectiveness of our gap-filling solutions (Section 3.6).

Table 3.1 summarizes notations used in this chapter.

Table 3.1: List of notations

Symbol	Definition
$G_0 = (V, E_{G_0})$	true graph with $n = \|V\|$ and $m = \|E_{G_0}\|$
$\mathcal{G} = (V, E, p)$	uncertain graph constructed from G_0
$G = (V, E_G)$	sample graph from $\mathcal{G}$, $G \sqsubseteq \mathcal{G}$
$d_u(G), d_u(\mathcal{G})$	degree of node u in $G, \mathcal{G}$
$\Delta(d)$	number of nodes having degree d in G
$\mathcal{N}(u)$	neighbors of node u in $\mathcal{G}$
R_σ	truncated normal distribution on [0,1]
σ	standard deviation of the normal distribution
$r_e \leftarrow R_\sigma$	a sample from the distribution R_σ
p_i (p_{uv})	probability of edge e_i (e_{uv})
n_p	number of potential edges, $\|E\| = m + n_p$
A, $\mathcal{A}$, $A(G)$	adjacency matrices of G_0, $\mathcal{G}$ and G
P_{RW}	random walk transition matrix of G_0
$B^{(t)}$	uncertain adjacency matrix, $B^{(t)} = AP_{RW}^{t-1}$
t	walk length
S	switching matrix
TV	total degree variance

3.2 A General Model

This section starts with definitions and common assumptions on uncertain graphs. It then introduces a general model UAM via semantics of edge uncertainty.

3.2.1 Uncertain Graph

Let $\mathcal{G} = (V, E, p)$ be an uncertain undirected graph, where V is the set of nodes, E is the set of edges, $p : E \rightarrow (0, 1]$ is the function that gives an existence probability to each edge (see Fig.3.3b). The common assumption is on the *independence* of edge probabilities. Following the *possible-worlds* semantics in relational data [29], the uncertain graph $\mathcal{G}$ induces a set $\{G = (V, E_G)\}$ of $2^{|E|}$ deterministic graphs (worlds), each is defined by a subset of E. The probability of $G = (V, E_G) \sqsubseteq \mathcal{G}$ is:

$$Pr(G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)) \tag{3.1}$$

As an example, the uncertain graph in Fig.3.3b has 2^5 possible graphs and the graph with three edges $(v1, v2), (v2, v3)(v3, v4)$ has probability of $P(G) = 0.3 \times 0.7 \times 0.4 \times (1 - 0.8) \times (1 - 0.9) = 0.00168$. Note that deterministic graphs are also uncertain graphs with all existing edges having probabilities 1 and non-existing edges having probabilities 0.

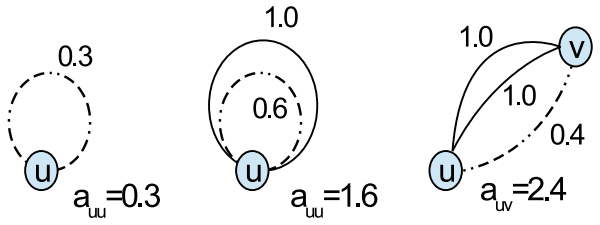


Figure 3.1: Semantics of selfloops (left), multi-selfloops (middle) and multiedges (right) in uncertain graph.

3.2.2 Uncertain Adjacency Matrix (UAM)

Given the true undirected graph G_0, an uncertain graph $\mathcal{G}$ constructed from G_0 must have its uncertain adjacency matrix $\mathcal{A}$ satisfying

1. $\mathcal{A}_{ij} = \mathcal{A}_{ji}$ (symmetry);
2. $\mathcal{A}_{ij} \in [0, 1]$ and $\mathcal{A}_{ii} = 0$ (no multiedges or selfloops);
3. $\sum_{j=1}^{n} \mathcal{A}_{ij} = d_i(G_0) \;\; i = 1..n$, (*expected degrees* of all nodes must be unchanged).

While the constraints (1) and (2) are straightforward for uncertain undirected graph, the third constraint is novel and central to our model of UAM. It stems from the need of preserving the *expected* degree sequence of graph which is useful for degree distribution estimation, e.g. in [45]. In terms of *dK-series* [60], the degree sequence is *d1-series*, the first moment of graph. The higher moments like *d2-series* are left for future work.

By relaxing (2) to (2'): $\mathcal{A}_{ii} \geq 0$ and $\mathcal{A}_{ij} \geq 0$, we allow graphs with *selfloop*s, *multi-selfloop*s and *multiedge*s (see Fig. 3.1). In other words, the non-negative weighted edges fully capture the semantics of selfloops, multi-selfloops and multiedges in the model of UAM.

3.3 Application of UAM

In this section, we analyze several existing and novel schemes using the model of UAM.

3.3.1 RandWalk Scheme

Preliminary results

We first define the transition matrix P_{RW} which is right *stochastic* (i.e. non-negative and row sums equal to 1) as follows (note that we use the short notation $d_i = d_i(G_0)$)

$$P_{RW}(i,j) = \begin{cases} 1/d_i & \text{if } (i,j) \in E_{G_0} \;\; i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The power P_{RW}^t when $t \to \infty$ is $P_{RW}^{\infty}(i,j) = \frac{d_j}{2m}$.

We prove two lemmas on properties of the products $\mathcal{A}P$ and AP^t where P is right stochastic.

Lemma 3.1. *For an adjacency matrix $\mathcal{A}$ and a right stochastic matrix P, the product $\mathcal{A}P$ is non-negative and has row sums equal to those of $\mathcal{A}$.*

Proof. The non-negativity of $\mathcal{A}P$ is trivial. The sum of row i of $\mathcal{A}P$ is $\sum_j(\sum_k \mathcal{A}_{ik}P_{kj}) = \sum_k \mathcal{A}_{ik}(\sum_j P_{kj}) = \sum_k \mathcal{A}_{ik}.1 = \sum_k \mathcal{A}_{ik}$ □

Lemma 3.2. *For a deterministic graph G possessing adjacency matrix A and P_{RW}, the product $B^{(t)} = AP_{RW}^{t-1}$ is also symmetric.*

Proof. We prove the result by induction. The case $t = 1$ is trivial. We prove that for any $t \geq 2$, $B_{ij}^{(t)} = \sum_{p_t(i,j)} \prod_{k\in p_t(i,j),k\neq i,j} 1/d_k$ where $p_t(i,j)$ is a path of length t from i to j.

When $t = 2$, $B_{ij}^{(2)} = \sum_k A_{ik}P_{kj} = \sum_{(i,k),(k,j)\in E} 1/d_k$, so the result holds. Assuming that the result is correct up to $t-1$, i.e. $B_{ij}^{(t-1)} = \sum_{p_{t-1}(i,j)} \prod_{k\in p_{t-1}(i,j),k\neq i,j} 1/d_k$. Because $B^{(t)} = B^{(t-1)}P_{RW}$, we have $B_{ij}^{(t)} = \sum_l B_{il}^{(t-1)}P_{lj} =$
$\sum_{l,(l,j)\in E}(\sum_{p_{t-1}(i,l)} \prod_{k\in p_{t-1}(i,l),k\neq i,l} 1/d_k)1/d_l =$
$\sum_{p_t(i,j)} \prod_{k\in p_t(i,j),k\neq i,j} 1/d_k$.

Because G is undirected, the set of all $p_t(i,j)$ is equal to the set of all $p_t(j,i)$, so $B_{ij}^{(t)} = B_{ji}^{(t)}$. □

We prove the uniqueness of P_{RW} as follows

Proposition 3.1. *Given a deterministic graph G with adjacency matrix A, there exists one and only one right stochastic matrix P that satisfies $P_{uv} = 0$ for all $(u,v) \notin G$ and AP^t is symmetric for all $t \geq 0$. The unique solution is $P = P_{RW}$.*

Proof. Lemma 3.2 shows that $P = P_{RW}$ satisfies $P_{uv} = 0$ for all $(u,v) \notin G$ and AP^t is symmetric for all $t \geq 0$.

To prove that this is the unique solution, we repeat the formula in the proof of Lemma 3.2. Let $B^{(t)} = AP^{t-1}$, then $B_{ij}^{(t)} = \sum_{p_t(i,j)} \prod_{k\in p_t(i,j),k\neq i,j} P_{k,k+1}$ where $k+1$ implies the successive node of k in $p_t(i,j)$. Because $B_{ji}^{(t)}$ has the same number of products as $B_{ij}^{(t)}$ (i.e. the number of paths of length t), $B^{(t)}$ is symmetric if and only if corresponding products are equal, i.e. $\prod_{k\in p_t(i,j),k\neq i,j} P_{k,k+1} = \prod_{k\in p_t(j,i),k\neq i,j} P_{k,k+1}$. At $t = 2$, for any path (i,k,j) we must have $P_{kj} = P_{ki}$. Along with the requirement that P is right stochastic, i.e. $\sum_i P_{ki} = 1$, we obtain $P_{ki} = 1/d_k$. This is exactly P_{RW}. □

Analysis

Now we show the fitting of *RandWalk* [66] to the model of UAM. We should mention that RandWalk is proposed only for *link privacy* analysis in [66] whereas the current work is on *node privacy*. Algorithm 1 depicts the steps of RandWalk. As we show below, the trial-and-error condition in Line 6 makes RandWalk hard to analyze [4]. So we modify it by removing the condition and using parameter α instead of 1.0 in Line 12 [5] (see Algorithm 2). When $\alpha = 0.5$, all edges (u,z) are assigned with probability 0.5. In *RandWalk-mod*, we add a checking for $d_u = 1$ (Line 8) to keep the total degree of G' equal to that of G, which is missing in RandWalk. Note that RandWalk-mod accepts selfloops and multiedges (i.e. it satisfies the relaxed constraint (2'), not (2)).

[4] It also causes edge miss at $t = 2$, e.g. a 2-length walk on edge $(v3, v2)$ (Fig. 3.3a) causes the selfloop $(v3, v3)$.
[5] This line causes errors for degree-1 nodes as shown in *RandWalk-mod*.

Algorithm 1 RandWalk(G_0, t, M) [66]

Input: undirected graph G_0, walk length t and maximum loop count M
Output: anonymized graph G'

1: $G' = null$
2: **for** u in G_0 **do**
3: $count = 1$
4: **for** v in $\mathcal{N}(u)$ **do**
5: $loop = 1$, $z = u$
6: **while** $(u == z \vee (u, z) \in G') \wedge (loop \leq M)$ **do**
7: perform $t - 1$ hop random walk from v
8: z is the terminal node of the random walk
9: $loop++$
10: **if** $loop \leq M$ **then**
11: **if** $count == 1$ **then**
12: add (u, z) to G' with probability 1.0
13: **else**
14: add (u, z) to G' with probability $\frac{0.5d_u - 1}{d_u - 1}$
15: $count++$
return G'

Let Q be the *edge adding* matrix defined as

$$Q_{ij} = \begin{cases} 0.5 & \text{if } d_i = 1 \wedge j \text{ is the unique neighbor of } i \\ \alpha & \text{if } j \text{ is the first neighbor of } i \\ \frac{0.5d_i - \alpha}{d_i - 1} & \text{if } j \text{ is a neighbor of } i \text{ but not the first one} \\ 0 & \text{otherwise.} \end{cases}$$

We show that *RandWalk-mod* can be formulated as an uncertain adjacency matrix $\mathcal{A}_{RW} = (AP_{RW}^{t-1}) \circ (Q + Q^T)$, where $\circ$ is the Hadamard product (element-wise). AP_{RW}^{t-1} is equivalent to computations in lines 2-6 and $Q+Q^T$ is equivalent to computations in lines 7-13. We use $Q+Q^T$ instead of Q due to the fact that when the edge (u, z) is added to G' with probability Q_{uz}, the edge (z, u) is also assigned the same probability. We come up with the following theorem.

Theorem 3.1. *RandWalk-mod can be formulated as* $\mathcal{A}_{RW} = (AP_{RW}^{t-1}) \circ (Q + Q^T)$. $\mathcal{A}_{RW}$ *is symmetric. It satisfies the constraint of unchanged expected degree iff* $\alpha = 0.5$ [6].

Proof. By Lemmas 3.1 and 3.2, let $B_{RW}^{(t)}$ be AP_{RW}^{t-1}, we have symmetric $B_{RW}^{(t)}$ and its row sums are equal to those of A. Because $\mathcal{A}_{RW} = B_{RW}^{(t)} \circ (Q + Q^T)$ and both $B_{RW}^{(t)}$ and $(Q + Q^T)$ are symmetric, $\mathcal{A}_{RW}$ is also symmetric.

Due to the fact that $(Q + Q^T)$ has the same locations of non-zeros as $B_{RW}^{(t)}$, the condition of unchanged expected degree is satisfied if and only if all non-zeros in $(Q + Q^T)$ are 1. This occurs if and only if $\alpha = 0.5$. □

We investigate the limit case when $t \to \infty$ (i.e. $P_{RW}^{t-1} \to P_{RW}^{\infty}$). Correspondingly $B_{RW}^{\infty} = AP_{RW}^{\infty}$ has $B_{RW}^{\infty}(i, j) = \frac{d_i d_j}{2m}$. The following theorem quantifies the number of selfloops and multiedges in B_{RW}^{∞} for power-law (PL) graphs and sparse Erdös-Renyi (ER) random graphs [73].

[6] This implies a mistake in Theorem 3 of [66]

Algorithm 2 RandWalk-mod(G_0, t, α)

Input: undirected graph G_0, walk length t and probability α
Output: anonymized graph G'

$G' = null$
for u in G_0 **do**
  $count = 1$
  for v in $\mathcal{N}(u)$ **do**
    perform $t-1$ hop random walk from v
    z is the terminal node of the random walk
    if $count == 1$ **then**
      if $d_u == 1$ **then**
        add (u, z) to G' with probability 0.5
      else
        add (u, z) to G' with probability α
    else
      add (u, z) to G' with probability $\frac{0.5d_u - \alpha}{d_u - 1}$
    $count++$
return G'

Theorem 3.2. *For power-law graphs with the exponent γ, the number of selfloops in B^{∞}_{RW} is $\frac{\zeta(\gamma-2)}{\zeta(\gamma-1)}$, where $\zeta(\gamma)$ is the Riemann zeta function defined only for $\gamma > 1$; the number of multiedges is zero.*

For sparse ER random graphs with $\lambda = np$ constant where p is the edge probability, the number of selfloops in B^{∞}_{RW} is $\lambda + 1$; the number of multiedges is zero.

Proof. For power-law graphs, the node degree distribution is $P(k) = \frac{k^{-\gamma}}{\zeta(\gamma)}$. The number of selfloops n^{PL}_{sl} in B^{∞}_{RW} is the sum of elements on the main diagonal.

$$n^{PL}_{sl} = \frac{1}{2m}\sum_{i=1}^{n} d_i^2 = \frac{1}{2m}\sum_{k=1}^{\infty} k^2 nP(k) = \frac{n}{nE(k)}\sum_{k=1}^{\infty} k^2 P(k)$$
$$= \frac{1}{E(k)}\sum_{k=1}^{\infty}\frac{k^{-(\gamma-2)}}{\zeta(\gamma)} = \frac{\zeta(\gamma)}{\zeta(\gamma-1)}\frac{\zeta(\gamma-2)}{\zeta(\gamma)} = \frac{\zeta(\gamma-2)}{\zeta(\gamma-1)}$$

To prove that there is no multiedge in B^{∞}_{RW} we show that all elements in B^{∞}_{RW} are less than 1. This is equivalent to show $d_{max} < \sqrt{2m}$. We use the constraint that the number of nodes with degree d_{max} must be at least 1, i.e. $\frac{n d_{max}^{-\gamma}}{\zeta(\gamma)} \geq 1 \leftrightarrow d_{max} \leq (n/\zeta(\gamma))^{1/\gamma}$. Because $\zeta(\gamma) > 1$ and we consider $\gamma > 2$ in social networks, $(n/\zeta(\gamma))^{1/\gamma} < \sqrt{n}$. Meanwhile, $\sqrt{2m} = \sqrt{n\frac{\zeta(\gamma-1)}{\zeta(\gamma)}} > \sqrt{n}$ due to the fact that $\zeta(\gamma)$ is monotonically decreasing. So we conclude $n^{PL}_{me} = 0$.

For sparse ER random graphs, we have $P(k) \to e^{-\lambda}\frac{\lambda^k}{k!}$. The number of selfloops n^{ER}_{sl} is

$$n^{ER}_{sl} = \frac{1}{2m}\sum_{i=1}^{n} d_i^2 = \frac{1}{2m}\sum_{k=1}^{\infty} k^2 nP(k) = \frac{n}{nE(k)}\sum_{k=1}^{\infty} k^2 P(k)$$
$$= \frac{1}{\lambda}E(k^2) = \frac{1}{\lambda}(E(k)^2 + Var(k)) = \frac{1}{\lambda}(\lambda^2 + \lambda) = \lambda + 1$$

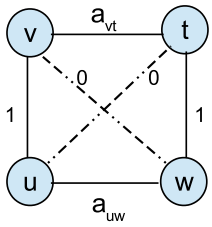


Figure 3.2: Edge switching

$$\begin{bmatrix} 0 & 1 & a_{uw} & 0 \\ 1 & 0 & 0 & a_{vt} \\ a_{uw} & 0 & 0 & 1 \\ 0 & a_{vt} & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -a_{vt} & 1 & a_{vt} \\ -a_{uw} & 0 & a_{uw} & 1 \\ 1 & a_{vt} & 0 & -a_{vt} \\ a_{uw} & 1 & -a_{uw} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & a_{uw} & 1 \\ 0 & 0 & 1 & a_{vt} \\ a_{uw} & 1 & 0 & 0 \\ 1 & a_{vt} & 0 & 0 \end{bmatrix} \tag{3.4}$$

Similar to the case of PL graphs, we show that $d_{max} < \sqrt{2m}$ where $d_{max} = \max_k ne^{-\lambda}\frac{\lambda^k}{k!} \geq 1 = \max_k \frac{k!}{\lambda^k} \leq ne^{-\lambda}$. Using the basic facts $k^{k/2} \leq k!$ and $k > \lambda$ we get $\frac{k^{k/2}}{\lambda^k} \leq ne^{-\lambda} < n$, so $k < n^{2/k}\lambda^2 < \sqrt{n\lambda} = \sqrt{2m}$ as long as n is sufficiently large and $\lambda \geq 4$. So we conclude $n_{me}^{ER} = 0$. □

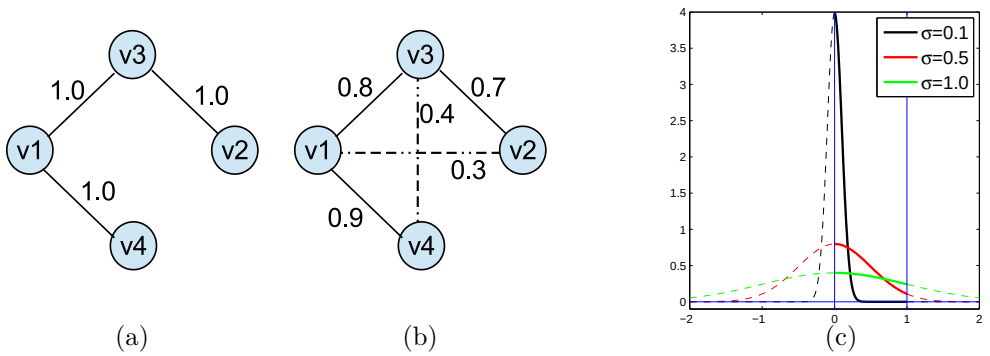
3.3.2 Edge Switching Scheme

In edge switching (*EdgeSwitch*) approaches (e.g. [107]), two edges $(u, v), (w, t)$ are chosen and switched to $(u, t), (w, v)$ if $a_{ut} = a_{wv} = 0$ (Fig. 3.2). This is done in s switches. Using the switching matrix S, we represent 1-step EdgeSwitch in the form $AS = \mathcal{A}$ (Equation (3.4)).

The switching matrix S is feasible if and only if $a_{uw}a_{vt} = 0$. Note that in the full form, S is $n \times n$ matrix with the $n - 4$ remaining elements on diagonal are 1, other off-diagonal are 0. In general, S is not right stochastic and this happens only if $a_{uw} = a_{vt} = 0$. For s-step EdgeSwitch $A\prod_{i=1}^{s} S_i = \mathcal{A}$. If $\forall i, S_i$ is right stochastic (i.e. we choose edges $(u, v), (w, t)$ such that $a_{uw} = a_{vt} = 0$), then Lemma 3.1 applies.

3.3.3 (k, ϵ)-obf Scheme

Given the deterministic adjacency matrix A, we can directly construct $\mathcal{A}$ that satisfies all three constraints (1),(2) and (3) of UAM. *(k,ϵ)-obf* [11] introduces such an approach.



(a) (b) (c)

Figure 3.3: (a) True graph (b) An obfuscation with potential edges (dashed) (c) Truncated normal distribution on [0,1] (bold solid curves)

Construction and Limitations

In [11], Boldi et al. extend the concept of *k-obfuscation* developed earlier [13] which uses Shannon entropy.

Definition 3.1. *(k,ϵ)-obf [11]. Let P be a vertex property, $k \geq 1$ be a desired level of obfuscation, and $\epsilon \geq 0$ be a tolerance parameter. The uncertain graph $\mathcal{G}$ is said to k-obfuscate a given vertex $v \in G$ with respect to P if the entropy of the distribution $Y_{P(v)}$ over the vertices of $\mathcal{G}$ is greater than or equal to $\log_2 k$:*

$$H(Y_{P(v)}) \geq \log_2 k \tag{3.5}$$

The uncertain graph $\mathcal{G}$ is a (k, ϵ)-obf with respect to property P if it k-obfuscates at least $(1-\epsilon)n$ vertices in $\mathcal{G}$ with respect to P. □

Given the true graph G_0 (Fig.3.3a), the basic idea of (k, ϵ)-*obf* (Fig.3.3b) is to transfer the probabilities from existing edges to *potential* (non-existing) edges to satisfy Definition 3.1. For each existing sampled edge e, it is assigned a probability $1 - r_e$ where $r_e \leftarrow R_\sigma$ (Fig. 3.3c) and for each non-existing sampled edge e', it is assigned a probability $r_{e'} \leftarrow R_\sigma$.

Table 3.2 gives an example of how to compute degree entropy for the uncertain graph in Fig. 3.3b. Here vertex property P is the node degree. Each row in the left side is the degree distribution for the corresponding node. For instance, $v1$ has degree 0 with probability $(1 - 0.8).(1 - 0.3).(1 - 0.9) = 0.014$. The right side normalizes values in each column (i.e. in each degree value) to get distributions $Y_{P(v)}$. The entropy $H(Y_{P(v)})$ for each degree value is shown in the bottom row. Given $k = 3, \log_2 k = 1.585$, then $v1, v3$ with true degree 2 and $v2, v4$ with true degree 1 satisfy (3.5). Therefore, $\epsilon = 0$.

Table 3.2: The degree uncertainty for each node (left) and normalized values for each degree (right)

	node degree uncertainty				$Y_{P(v)}$			
	d=0	d=1	d=2	d=3	d=0	d=1	d=2	d=3
v1	.014	.188	.582	.216	.044	.117	.355	.491
v2	.210	.580	.210	.000	.656	.362	.128	.000
v3	.036	.252	.488	.224	.112	.158	.298	.509
v4	.060	.580	.360	.000	.187	.362	.220	.000
				H	1.40	1.84	1.91	0.99

While (k, ϵ)-*obf* provides a novel technique to come up with an uncertain version of the graph, the specific approach in [11] has two drawbacks. First, it formulated the problem as the minimization of σ. With small values of σ, r_e highly concentrates around zero, so existing sampled edges have probabilities nearly 1 and non-existing sampled edges are assigned probabilities almost 0. Simple rounding techniques can easily reveal the true graph. Even if the graph owner only publishes sample graphs, the re-identification attacks are still effective as we show in Section 3.6. Note that in [11], the found values of σ vary in a wide range from 10^{-1} to 10^{-8}. Second, the approach in [11] does not consider the locality (subgraph) of nodes in selecting pairs of nodes for establishing potential edges. As shown in [36], *subgraph-wise perturbation* effectively reduces structural distortion.

Furthermore, the expected degrees of nodes in (k, ϵ)-obf are *approximately* unchanged due to the fact that $r_e, r_{e'}$ are nearly zero by small σ. So (k,ϵ)-obf satisfies the constraints (1) and (2) but it only approximately satisfies the constraint (3) of UAM. To remedy these shortcomings,

we present the MaxVar approach in Section 3.4. It adds potential edges to G_0, then tries to find the assignment of edge probabilities such that the expected node degrees are unchanged while the total variance is maximized. A comparison among schemes is also shown in Section 3.4.3.

3.3.4 Mixture Scheme

In this section, we present the *Mixture* approach applicable to RandWalk-mod, (k, ϵ)-obf, MaxVar and EdgeSwitch. Its UAM $\mathcal{A}_p$ is parametrized by p, with the output sample graph G_p. Given the true graph G_0 and an anonymized $G \sqsubseteq \mathcal{G}$, every edge (i, j) is chosen into G_p with probability $\mathcal{A}_p(i, j)$ where

$$\mathcal{A}_p(i,j) = \begin{cases} 1 & \text{if } (i,j) \in E_{G_0} \cap E_G \\ 1-p & \text{if } (i,j) \in E_{G_0} \setminus E_G \\ p & \text{if } (i,j) \in E_G \setminus E_{G_0} \end{cases}$$

It is straightforward to show that $\mathcal{A}_p = (1-p)A(G_0) + pA(G)$. When applied to G generated by *RandWalk-mod* with $\alpha = 0.5$, we have $\mathcal{A}_p = (1-p)A + pAP_{RW}^{t-1} = A[(1-p)I_n + pP_{RW}^{t-1}]$ and $\mathcal{A}_p$ satisfies three constraints (1) (2') and (3).

If there exists P_{mix} with constraint $P_{mix}(i, j) = 0$ if $(i, j) \notin E_{G_0}$ such that $P_{mix}^{t-1} = (1-p)I_n + pP_{RW}^{t-1}$, then Mixture can be simulated by the RandWalk-mod approach with the transition matrix P_{mix}.

3.3.5 Partition Scheme

Another approach that can apply to RandWalk-mod, (k, ϵ)-obf, MaxVar and EdgeSwitch is the *Partition* approach. Given true graph G_0, this divide-and-conquer strategy first partitions G_0 into disjoint subgraphs sG, then it applies one of the above anonymization schemes on subgraphs to get anonymized subgraphs $s\mathcal{G}$. Finally, it combines $s\mathcal{G}$ to obtain $\mathcal{G}$. Note that the partitioning may cause orphan edges as in MaxVar (Section 3.4). Those edges must be copied to $\mathcal{G}$ to keep node degrees unchanged.

3.4 Variance Maximizing Scheme

We introduce the *Variance Maximizing* (MaxVar) scheme as an instance of the UAM. It achieves a good privacy-utility tradeoff by two key features: considering nearby potential edges and maximizing total node degree variance via a simple quadratic program. We start this section with the formulation of *MaxVar* in the form of quadratic programming based on two key observations. Then we describe the anonymization algorithm.

3.4.1 Formulation

Two key observations underpinning the MaxVar approach are presented as follows.

Observation #1: Maximizing Degree Variance

We argue that efficient countermeasures against structural attacks should hinge on node degrees. If a node and its neighbors have their degrees changed, the re-identification risk is reduced

significantly. Consequently, instead of replicating local structures as in k-anonymity based approaches [23,55,96,103,112,113], we can deviate the attacks by changing node degrees *probabilistically*. For example, node *v1* in Fig.3.3a has degree 2 with probability 1.0 whereas in Fig.3.3b, its degree gets four possible values $\{0, 1, 2, 3\}$ with probabilities $\{0.014, 0.188, 0.582, 0.216\}$ respectively. Generally, given edge probabilities of node u as $p_1, p_2, ..p_{d_u(\mathcal{G})}$, the degree of u is a sum of independent Bernoulli random variables, so its expected value is $\sum_{i=1}^{d_u(\mathcal{G})} p_i$ and its variance is $\sum_{i=1}^{d_u(\mathcal{G})} p_i(1-p_i)$. If we naively target the maximum (local) degree variance without any constraints, the naive solution is at $p_i = 0.5\ \forall i$. However, such an assignment distorts graph structure severely and deteriorates the utility. Instead, by following the model of UAM, we have the constraint $\sum_{i=1}^{d_u(\mathcal{G})} p_i = d_u(G_0)$. Note that the *minimum variance* of an uncertain graph is 0 and corresponds to the case $\mathcal{G}$ has all edges being deterministic, e.g. when $\mathcal{G} = G_0$ and in switching-edge based approaches. In the following section, we show an interesting result relating the *total* degree variance with the variance of edit distance.

Variance with edit distance

The *edit distance* between two deterministic graphs G, G' is defined as:

$$D(G, G') = |E_G \setminus E_{G'}| + |E_{G'} \setminus E_G| \tag{3.6}$$

A well-known result about the expected edit distance between the uncertain graph $\mathcal{G}$ and the deterministic graph $G \sqsubseteq \mathcal{G}$ is

$$E[D(\mathcal{G}, G)] = \sum_{G' \sqsubseteq \mathcal{G}} Pr(G')D(G, G') = \sum_{e_i \in E_G} (1 - p_i) + \sum_{e_i \notin E_G} p_i$$

Correspondingly, the variance of edit distance is

$$Var[D(\mathcal{G}, G)] = \sum_{G' \sqsubseteq \mathcal{G}} Pr(G')[D(G, G') - E[D(\mathcal{G}, G)]]^2$$

We prove in the following theorem that the variance of edit distance is the sum of all edges' variance (total degree variance) and it does not depend on the choice of G.

Theorem 3.3. *Assume that $\mathcal{G}(V, E, p)$ has k uncertain edges $e_1, e_2, ..., e_k$ and $G \sqsubseteq \mathcal{G}$ (i.e. $E_G \subseteq E$). The edit distance variance is $Var[D(\mathcal{G}, G)] = \sum_{i=1}^{k} p_i(1-p_i)$ and does not depend on the choice of G.*

Proof. We prove the result by induction.

When $k = 1$, we have two cases of G_1: $E_{G_1} = \{e_1\}$ and $E_{G_1} = \emptyset$. For both cases, $Var[D(\mathcal{G}_1, G_1)] = p_1(1 - p_1)$, i.e. independent of G_1.

Assume that the result is correct up to $k-1$ edges, i.e. $Var[D(\mathcal{G}_{k-1}, G_{k-1})] = \sum_{i=1}^{k-1} p_i(1-p_i)$ for all $G_{k-1} \sqsubseteq \mathcal{G}_{k-1}$, we need to prove that it is also correct for k edges. We use the subscript notations $\mathcal{G}_k, G_k$ for the case of k edges. We consider two cases of G_k: $e_k \in G_k$ and $e_k \notin G_k$.

Case 1. The formula for $Var[D(\mathcal{G}_k, G_k)]$ is

$$\begin{aligned} Var[D(\mathcal{G}_k, G_k)] &= \sum_{G'_k \sqsubseteq \mathcal{G}_k} Pr(G'_k)[D(G'_k, G_k) - E[D(\mathcal{G}_k, G_k)]]^2 \\ &= \sum_{e_k \in G'_k} Pr(G'_k)[Dk - E[D_k]]^2 + \sum_{e_k \notin G'_k} Pr(G'_k)[D_k - E[D_k]]^2 \end{aligned}$$

The first sum is $\sum_{G'_{k-1} \sqsubseteq \mathcal{G}_{k-1}} p_k Pr(G'_{k-1})[D_{k-1} - E[D_{k-1}] - (1-p_k)]^2$.

The second sum is $\sum_{G'_{k-1} \sqsubseteq \mathcal{G}_{k-1}} (1-p_k) Pr(G'_{k-1})[D_{k-1} - E[D_{k-1}] + p_k)]^2$.

Here we use shortened notations D_k for $D(G'_k, G_k)$ and $E[D_k]$ for $E[D(\mathcal{G}_k, G_k)]$.

By simple algebra, we have $Var[D(\mathcal{G}_k, G_k)] = Var[D(\mathcal{G}_{k-1}, G_{k-1})] + q_k(1-q_k) = \sum_{i=1}^{k} p_i(1-p_i)$.

Case 2. similar to the Case 1. □

Observation #2: Proposing Nearby Edges

As indicated by Leskovec et al. [53], real graphs reveal two temporal evolution properties: *densification power law* and *shrinking diameters*. Community Guided Attachment (CGA) model [53], which produces densifying graphs, is an example of a hierarchical graph generation model in which the linkage probability between nodes decreases as a function of their relative distance in the hierarchy. With regard to this observation, (k, ϵ)-obf, by heuristically making potential edges solely based on node degree discrepancy, produces many inter-community edges. Shortest-path based statistics will be reduced due to these edges. MaxVar, in contrast, tries to mitigate the structural distortion by proposing only *nearby* potential edges before assigning edge probabilities. Another evidence is from [98] where Vazquez analytically proved that *Nearest Neighbor* can explain the power-law for degree distribution, clustering coefficient and average degree among the neighbors. Those properties are in very good agreement with the observations made for social graphs. Sala et al. [88] confirmed the consistency of Nearest Neighbor model in their comparative study on graph models for social networks.

3.4.2 Algorithms

Overview

The intuition behind the new approach is to formulate the perturbation problem as a *quadratic programming* problem. Given the true graph G_0 and the number of potential edges allowed to be added n_p, the scheme has three phases. The first phase tries to partition G_0 into s subgraphs, each one with $n_s = n_p/s$ potential edges connecting nearby nodes (with default distance 2, i.e. *friend-of-friend*). The second phase formulates a quadratic program for each subgraph with the constraint of unchanged node degrees to produce the uncertain subgraphs $s\mathcal{G}$ with maximum edge variance. The third phase combines the uncertain subgraphs $s\mathcal{G}$ into $\mathcal{G}$ and publishes several sample graphs. The three phases are illustrated in Fig. 3.4.

By keeping the degrees of nodes in the perturbed graph, our approach is similar to the *edge switching* approaches (e.g. [107]) but ours is more subtle as we do it implicitly and the switching occurs not necessarily on pairs of edges.

Graph Partitioning

Because of the complexity of exact quadratic programming (Section 3.4.2), we need a preprocessing phase to divide the true graph G_0 into subgraphs and run the optimization on each subgraph. Given the number of subgraphs s, we run *METIS* [7] to get almost equal-sized subgraphs with minimum number of inter-subgraph edges. Each subgraph has n_s potential edges added before running the quadratic program. This phase is outlined in Algorithm 3.

[7] http://glaros.dtc.umn.edu/gkhome/views/metis

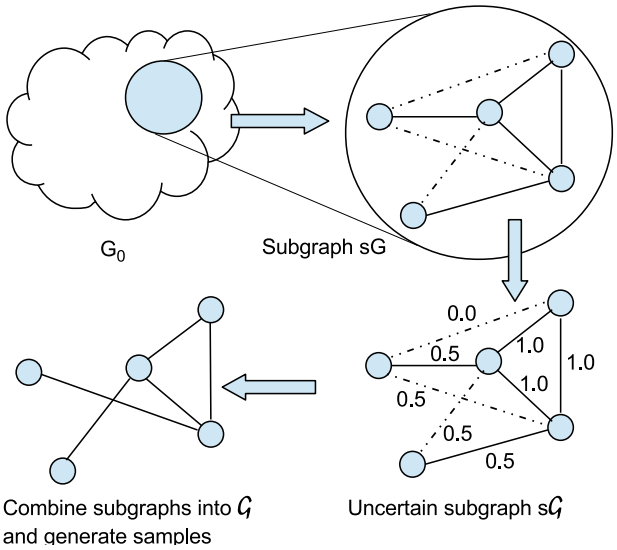


Figure 3.4: MaxVar approach

Algorithm 3 Partition-and-Add-Edges

Input: true graph $G_0 = (V, E_{G_0})$, number of subgraphs s, number of potential edges per subgraph n_s

Output: list of augmented subgraphs gl

1: $gl \leftarrow$ `METIS`(G_0, s).
2: **for** sG in gl **do**
3: $\quad i \leftarrow 0$, $E_{sG} \leftarrow \varnothing$
4: $\quad$ **while** $i < n_s$ **do**
5: $\quad\quad$ randomly pick $u, v \in V_{sG}$ and $(u, v) \notin E_{sG}$ with $d(u, v) = 2$
6: $\quad\quad E_{sG} \leftarrow E_{sG} \cup (u, v)$
7: $\quad\quad i \leftarrow i + 1$
return gl

Quadratic Programming

By assuming the independence of edges, the total degree variance of $\mathcal{G} = (V, E, p)$ for edit distance (Theorem 3.3) is:

$$Var(E) = \sum_{i=1}^{|E|} p_i(1-p_i) = |E_{G_0}| - \sum_{i=1}^{|E|} p_i^2 \tag{3.7}$$

The last equality in (3.7) is due to the constraint that the expected node degrees are unchanged (i.e. $\sum_{i=1}^{d_u(\mathcal{G})} p_i = d_u(G_0)$), so $\sum_{i=1}^{|E|} p_i$ is equal to $|E_{G_0}|$. By targeting the maximum edge variance, we come up with the following quadratic program.

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{|E|} p_i^2 \\ \text{Subject to} \quad & 0 \le p_i \le 1 \ \forall i \\ & \sum_{v \in \mathcal{N}(u)} p_{uv} = d_u(G_0) \ \forall u \end{aligned}$$

The objective function reflects the privacy goal (i.e. sample graphs do not highly concentrate around the true graph) while the expected degree constraints aim to preserve the utility.

By dividing the large input graph into subgraphs, we solve independent quadratic optimization problems. Because each edge belongs to at most one subgraph and the expected node degrees in each subgraph are unchanged, it is straightforward to show that the expected node degrees in $\mathcal{G}$ are also unchanged. We have a proposition on problem feasibility and an upper bound for the total variance.

Proposition 3.2. *The quadratic program in MaxVar is always feasible. The total variance* $TV_{MaxVar} = Var(E)$ *is upper bounded by* $\frac{mn_p}{m+n_p}$.

Proof. The feasibility is due to the fact that $\{p_e | p_e = 1 \ \forall e \in E_{G_0}$ and $p_e = 0$ otherwise$\}$ is a feasible point. Let k_u be the number of potential edges incident to node u. By requiring u's expected degree to be unchanged, we have $\sum_{v \in \mathcal{N}(u)} p_{uv} = d_u$. Applying Cauchy-Schwarz inequality, we get $\sum_{v \in \mathcal{N}(u)} p_{uv}^2 \ge \frac{1}{d_u + k_u} (\sum_{v \in \mathcal{N}(u)} p_{uv})^2 = \frac{d_u^2}{d_u + k_u}$. Now we take the sum over all nodes to get the following

$$\begin{aligned} Var(E) = m - \sum_{i=1}^{m+n_p} p_i^2 &= m - \frac{1}{2} \sum_u \sum_{v \in \mathcal{N}(u)} p_{uv}^2 \\ &\le m - \frac{1}{2} \sum_u \frac{d_u^2}{d_u + d_k} \le m - \frac{1}{2} \frac{(\sum_u d_u)^2}{\sum_u (d_u + k_u)} = \frac{mn_p}{m+n_p} \end{aligned}$$

where the last equality is again due to Cauchy-Schwarz inequality. □

3.4.3 Comparison of schemes

Table 3.3 shows the comparison of schemes we investigate in this chapter. Only MaxVar and EdgeSwitch satisfy all three properties (1),(2) and (3) of the UAM. The next two propositions quantify the TV of (k, ϵ)-obf and *RandWalk-mod*.

Table 3.3: Comparison of schemes

Scheme	Prop.1	Prop.2	Prop.3	Uncertain $\mathcal{A}$
RandWalk-mod	∘ ($\alpha = 0.5$)	×	∘	∘
RandWalk [66]	∘	∘	×	∘
EdgeSwitch	∘	∘	∘	×
(k, ϵ)-*obf* [11]	∘	∘	×	∘
MaxVar	∘	∘	∘	∘
Mixture	depends on the mixed scheme			
Partition	depends on the scheme used in subgraphs			

Proposition 3.3. *The expected total variance of (k, ϵ)-obf TV_{obf} is $(m + n_p)(E[r_e] - E[r_e^2])$. The expressions of $E[r_e], E[r_e^2]$ are given in (3.8) and (3.9).*

Proof. In (k, ϵ)-obf, m existing edges are assigned probabilities $1 - r_e$ while n_p potential edges are assigned probabilities r_e. Therefore, the total variance is $TV_{obf} = m(1 - r_e)(1 - (1 - r_e)) + n_p r_e(1 - r_e) = (m + n_p)r_e(1 - r_e)$ where $r_e \leftarrow R_\sigma$. Take the expectation of TV_{obf}, we get $E[TV_{obf}] = (m + n_p)(E[r_e] - E[r_e^2])$.

R_σ has pdf $f(x) = C\frac{1}{\sigma\sqrt{2\pi}}e^{-x^2/2\sigma^2}$ if $x \in [0, 1]$ and 0 otherwise. The normalization constant $C = 0.5\text{erf}(1/\sigma\sqrt{2})$ where erf is the error function. Basic integral computations (change of variable and integration by parts) give us the formulas for $E[r_e]$ and $E[r_e^2]$ as follows

$$E[r_e] = \frac{C\sigma}{\sqrt{2\pi}}(1 - e^{-1/2\sigma^2}) \tag{3.8}$$

$$E[r_e^2] = \frac{C\sigma}{\sqrt{2\pi}}(\frac{\sigma\sqrt{2\pi}}{C} - e^{-1/2\sigma^2}) \tag{3.9}$$

Note that for $\sigma \leq 0.1$, $C \approx 1$ and $e^{-1/2\sigma^2} \approx 0$, so

$$E[TV_{obf}] \approx (m + n_p)\left(\frac{\sigma}{\sqrt{2\pi}} - \sigma^2\right) \tag{3.10}$$

□

Proposition 3.4. *The total variance of RandWalk-mod $TV_{RW}(t)$ at walk-length t is upper bounded by $\frac{m(K_t - m)}{K_t}$ where K_t is the number of non-zeros in $B^{(t)}$.*

For power-law graphs with the exponent γ, $TV_{RW}^{PL}(\infty) = m - \frac{1}{2}\left[\frac{\zeta(\gamma-2)}{\zeta(\gamma-1)}\right]^2$. For sparse ER random graphs with $\lambda = np$ constant, $TV_{RW}^{ER}(\infty) = m - \frac{1}{2}(\lambda + 1)^2$

Proof. Proof for $TV_{RW}(t)$'s upper bound is obtained in the same way as in the proof of Proposition 3.2. At $t = \infty$, the computations of $TV_{RW}^{PL}(\infty)$ and $TV_{RW}^{ER}(\infty)$ are similar to those in the proof of Theorem 3.2.

$$TV_{RW}^{PL}(\infty) = \frac{1}{2}\sum_{i,j=1}^{n}\frac{d_i d_j}{2m}(1 - \frac{d_i d_j}{2m}) = m - \frac{1}{8m^2}\sum_{i,j=1}^{n} d_i^2 d_j^2 = m - \frac{1}{2E^2(k)}\left(\sum_{k=1}^{\infty} k^2 P(k)\right)^2$$
$$= m - \frac{\zeta(\gamma)^2}{2\zeta(\gamma-1)^2}\frac{\zeta(\gamma-2)^2}{\zeta(\gamma)^2} = m - \frac{1}{2}\left[\frac{\zeta(\gamma-2)}{\zeta(\gamma-1)}\right]^2$$

$$TV_{RW}^{ER}(\infty) = m - \frac{1}{2E^2(k)}\left(\sum_{k=1}^{\infty} k^2 P(k)\right)^2 = m - \frac{1}{2\lambda^2}(\lambda^2 + \lambda)^2 = m - \frac{1}{2}(\lambda + 1)^2$$

□

Note that the K_t increases with t and when t is equal to the diameter of G, $K_t = n^2$. Therefore, the upper bound of $TV_{RW}(t)$ converges very fast to m, compatible with the results in the limit cases of *PL* and *ER* random graphs.

By propositions 3.2, 3.3 and 3.4, we roughly conclude that

$$E[TV_{obf}] < TV_{MaxVar} < TV_{RW}(t) < m \tag{3.11}$$

3.4.4 Directed Graphs

For directed graphs, the UAM model can be extended in several ways. We can apply the constraint of unchanged expected degree to in-degrees, out-degrees or both in/out-degrees of nodes.

In the case of unchanged out-degrees, it is straightforward to verify that *RandWalk-mod* still keeps the expected out-degrees of all nodes (note that we have to double the probabilities of adding every (u, v) due to the directionality of edges). The case of unchanged in-degrees needs reverse random walks (i.e. walks on in-edges) and the same conclusion holds. However, if we require both expected in/out-degrees of nodes to be unchanged, there exists no such random-walks. This is not the case for (k, ϵ)-obf and MaxVar. For MaxVar, the constraints in (3.4.2) is updated to

$$\sum_{v \in \mathcal{N}^-(u)} p_{uv} = d_u^-(G_0) \qquad \sum_{v \in \mathcal{N}^+(u)} p_{uv} = d_u^+(G_0)\ \forall u$$

Interestingly, in the case of unchanged out-degrees, MaxVar has a trivial solution $p_{uv} = d_u^+(G_0)/d_u^+(\mathcal{G})$ due to Cauchy-Schwarz inequality. The case of unchanged in-degrees is similar. Because the main focus of this chapter is on undirected graphs, we leave the full analysis of UAM on directed graphs for future work.

3.5 Quantifying Framework

This section describes a generic framework for privacy and utility quantification of anonymization methods.

3.5.1 Privacy Measurement

We focus on structural re-identification attacks under various models of attacker's knowledge as shown in [46]. We quantify the privacy of an anonymized graph as the *sum* of re-identification probabilities of all nodes in the graph. We differentiate *closed-world* from *open-world* adversaries as in [46]. For example, when a closed-world adversary knows that Bob has three neighbors, this fact is exact. An open-world adversary in this case would learn only that Bob has at least three neighbors. We consider the result of *structural query* Q (Definition 2.2) on a node u as the *node signature* $sig_Q(u)$. For example, by a structural query returning degree of each node, the node signature of u in the graph G is $d_G(u)$. Given a structural query Q, nodes having the same

signature form an *equivalence class*. We define *node privacy score* as the inference probability based on node signatures in the true graph and the sample graph. Given the true graph G_0 and an output anonymized graph G^*, the privacy score is measured as the sum of all node privacy scores. We illustrate the privacy score in the following example.

Example 3.1. Assuming that we have signatures of G_0 and signatures of G^* as in Table 3.4, the re-identification probabilities in G^* of nodes 1,2,8 are $\frac{1}{3}$, of node 4 is $\frac{1}{2}$, of nodes 3,5,6,7 are 0s. And the privacy score of G^* is $\frac{1}{3}+\frac{1}{3}+0+\frac{1}{2}+0+0+0+\frac{1}{3}=1.5$. In G_0, the privacy score is $\frac{1}{3}+\frac{1}{3}+\frac{1}{3}+\frac{1}{2}+\frac{1}{2}+\frac{1}{3}+\frac{1}{3}+\frac{1}{3}=3$, equal to the number of equivalence classes.

Table 3.4: Example of node signatures

Graph	Equivalence classes
G_0	$s_1\{1,2,3\}, s_2\{4,5\}, s_3\{6,7,8\}$
G^*	$s_1\{1,2,6\}, s_2\{4,7\}, s_3\{3,8\}, s_4\{5\}$

Note that in the above example, a more detailed node mapping from G^* to G_0 may assign the privacy score $1/n$ to the nodes whose signatures have been changed. The total privacy score of such nodes is always less than one, much less than the number of equivalence classes in G_0. In this chapter, we simply assign these privacy scores to zero.

We consider two privacy scores in this chapter using two types of structural queries H_1 and $H2_{open}$.

- **H1** score uses node degree as the node signature, i.e. we assume that the attacker knows *apriori* degrees of all nodes.
- **H2$_{\text{open}}$** uses the *set* (not multiset) of degrees of node's friends as the node signature. For example, if a node has 6 neighbors and the degrees of those neighbors are $\{1,2,2,3,3,5\}$, then its signature for $H2_{open}$ attack is $\{1,2,3,5\}$.

Higher-order scores like $H2$ (exact multiset of neighbors' degrees) or $H3$ (exact multiset of neighbor-of-neighbors' degrees) induce much higher privacy scores of the true graph G_0 (in the order of $|V|$) and represent less meaningful metrics for privacy. The following proposition claims the automorphism invariant property of structural privacy scores.

Proposition 3.5. *All privacy scores based on structural queries [46] are automorphism-invariant, i.e. if we find a non-trivial automorphism G_1 of G_0, the signatures of all nodes in G_1 are unchanged.*

Proof. G_1 is an automorphism of G_0 if there exists a permutation $\sigma : V \to V$ such that $(u,v) \in E_{G_0} \leftrightarrow (\sigma(u),\sigma(v)) \in E_{G_1}$. For $H1$ score, it is straightforward to verify that $H1_{G_1}(u) = H1_{G_0}(u)$ according to the definition of σ.

For $H2_{open}$ score, we prove that $\forall d_v \in H2_{G_0}(u)$ we also have $d_v \in H2_{G_1}(u)$ and vice versa. Because $d_v \in H2_{G_0}(u) \to (u,v) \in E_{G_0} \to (\sigma(u),\sigma(v)) \in E_{G_1}$. Note that $d_{\sigma(v)} = d_v$ ($H1$ unchanged), so $d_v \in H2_{G_1}$. The reverse is proved similarly.

This argument can be extended to any structural queries (signatures) in [46]. □

3.5.2 Utility Measurement

Following [11] and [107], we consider three groups of statistics for utility measurement: degree-based statistics, shortest-path based statistics and clustering statistics.

Degree-based statistics

- Number of edges: $S_{NE} = \frac{1}{2}\sum_{v \in V} d_v$

- 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 $\Delta(d) \sim d^{-\gamma}$

Shortest path-based statistics

- 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.

Clustering statistics

- 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.

All of the above statistics are computed on sample graphs generated from the uncertain output $\mathcal{G}$. In particular, to estimate shortest-path based measures, we use Approximate Neighbourhood Function (ANF) [80]. The diameter is lower bounded by the longest distance among all-destination bread-first-searches from 1,000 randomly chosen nodes. The *relative error* (rel.err) for each statistic S is computed as $\frac{|S(G_0) - S_{avg}(G)|}{S(G_0)}$. The relative error for a sample graph is the average of relative errors of all the statistics defined above.

In this chapter, we define the privacy/utility *tradeoff* as $\sqrt{H2_{open}} \times rel.err$ as we conjecture the quadratic and linear behaviors of $H2_{open}$ and $rel.err$ respectively.

3.6 Evaluation

In this section, our evaluation aims to show the disadvantages of (k, ϵ)-*obf* and *RandWalk/RandWalk-mod* as well as the gap between them. We then illustrate the effectiveness and efficiency of the gap-filling approaches *MaxVar* and *Mixture*. The effectiveness is measured by privacy scores (lower is better), the relative error of utility (lower is better) and the privacy/utility tradeoff (lower is better). The efficiency is measured by the running time. All algorithms are implemented in Python and run on a desktop PC with *Intel*® Core i7-4770@ 3.4Ghz, 16GB memory. We use *MOSEK*[8] as the quadratic solver.

Three large real-world datasets are used in our experiments [9]. `dblp` is a co-authorship network where two authors are connected if they publish at least one paper together. `amazon` is a product co-purchasing network where the graph contains an undirected edge from i to j if a product i is frequently co-purchased with product j. `youtube` is a video-sharing web site that includes a social network. The graph sizes $(|V|, |E|)$ of `dblp`, `amazon` and `youtube` are (317080, 1049866), (334863, 925872) and (1134890, 2987624) respectively. We partition `dblp`, `amazon` into 20 subgraphs and `youtube` into 60 subgraphs. The sample size of each test case is 20.

[8] http://mosek.com/

[9] http://snap.stanford.edu/data/index.html

3.6.1 (k, ϵ)-obf and RandWalk

We report the performance of (k, ϵ)-*obf* in Table 3.5. We keep the number of potential edges equal to m (default value in [11]) and vary σ. We find that the scheme achieves low relative errors only at small σ. However, privacy scores, especially $H2_{open}$, rise fast (up to 50% compared to the true graph). This fact incurs high privacy-utility tradeoff as confirmed in Table 3.8.

Table 3.6 shows the performance similarity between *RandWalk* and *RandWalk-mod* except the case of `youtube` and for $t = 2$ in `amazon`. Because *RandWalk-mod* satisfies the third constraint, it benefits several degree-based statistics while the existence of selfloops and multiedges does not impact much on shortest-path based metrics. *RandWalk* misses a lot of edges at $t = 2$ (see footnote 4 in Section 3.3.1). The remarkable characteristics of random-walk schemes are the very low privacy scores and the high relative errors (lower-bounded around 8 to 10%). Clearly, there is a gap between high tradeoffs in (k, ϵ)-*obf* and high relative errors in *RandWalk* where *MaxVar* and *Mixture* may play their roles.

3.6.2 Effectiveness of MaxVar

We assess privacy and utility of *MaxVar* by varying the number of potential edges n_p. The results are shown in Table 3.7 and Fig. 3.5. The *ratio of replaced edges* in Figures 3.5a,3.5b and 3.5c is defined as $\frac{|E_{G_0} \setminus E_G|}{|E_{G_0}|}$. As for privacy scores, if we increase n_p, we gain better privacy (lower privacy scores H_1 and $H2_{open}$) as we allow more edge switches, making more node signature changes. Due to the expected degree constraints in the quadratic program, all degree-based metrics vary only a little.

We observe the near *linear* relationships between $H1$, *rel.err* and the ratio of replaced edges in Figures 3.5a, 3.5c and near *quadratic* relationship of $H2_{open}$ against the ratio of replaced edges in Fig.3.5b. A analytical analysis of these phenomena would be a good suggestion for future work.

The runtime of MaxVar consists of time for (1) partitioning G_0, (2) adding friend-of-friend edges to subgraphs, (3) solving quadratic subproblems and (4) combining uncertain subgraphs to get $\mathcal{G}$. We report the runtime in Fig.3.5d. As we can see, the total runtime is in several minutes and the runtime of the partitioning step is almost negligible. Increasing n_p gives rise to the runtime in steps 2,3 and 4 and the trends are nearly linear. The runtime on `youtube` is three times longer than on the other two datasets, almost linear to their data sizes.

3.6.3 Comparative Evaluation

Table 3.8 shows comparisons between *MaxVar*, (k, ϵ)-obf and *RandWalk/RandWalk-mod* with the tradeoff column. We omit the column $H1 \times rel.err$ because they are almost equal for all schemes considered in this chapter. Clearly, MaxVar gains better privacy-utility tradeoffs than (k, ϵ)-obf, but worse than *RandWalk*, *RandWalk-mod*. However, MaxVar has its own merit as a gap-filling solution. Figures 3.6a,3.6b and 3.6c show that while RandWalk, RandWalk-mod have the best tradeoffs, they suffer from high lower bounds for utility. In other words, if the dataset allows higher privacy risk for better utility (lower rel.err) then the usage of two random walk based solutions may be limited. The simple solution *Mixture* also fills the gap. We omit EdgeSwitch due to its worst tradeoffs.

In addition to the re-identification scores $H1$ and $H2_{open}$, we also compute ϵ for $k \in \{30, 50, 100\}$ to have a fair comparison with (k, ϵ)-obf. Table 3.8 shows that *MaxVar* has the best (k, ϵ) scores. The number of potential edges used in MaxVar could be 20% of $|E_{G_0}|$, much

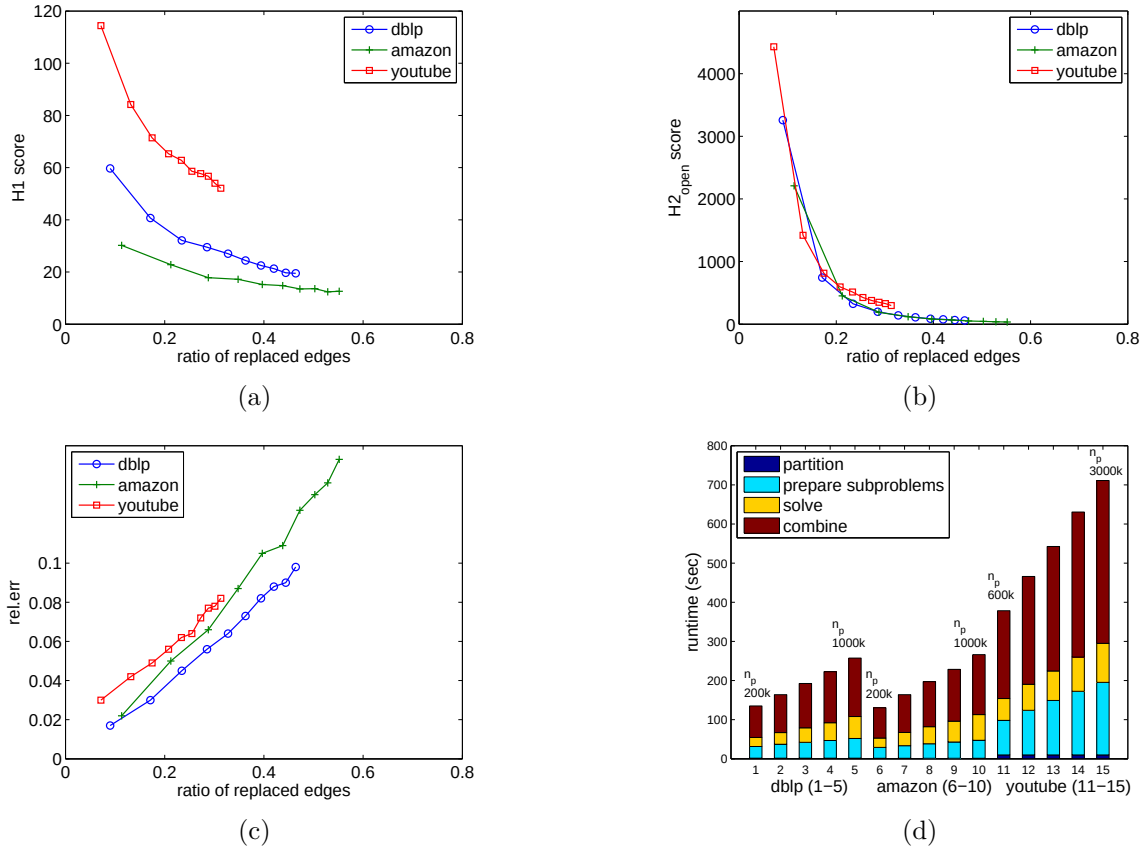


Figure 3.5: Performance of MaxVar (a) $H1$ score (b) $H2_{open}$ score (c) Relative error (d) Runtime

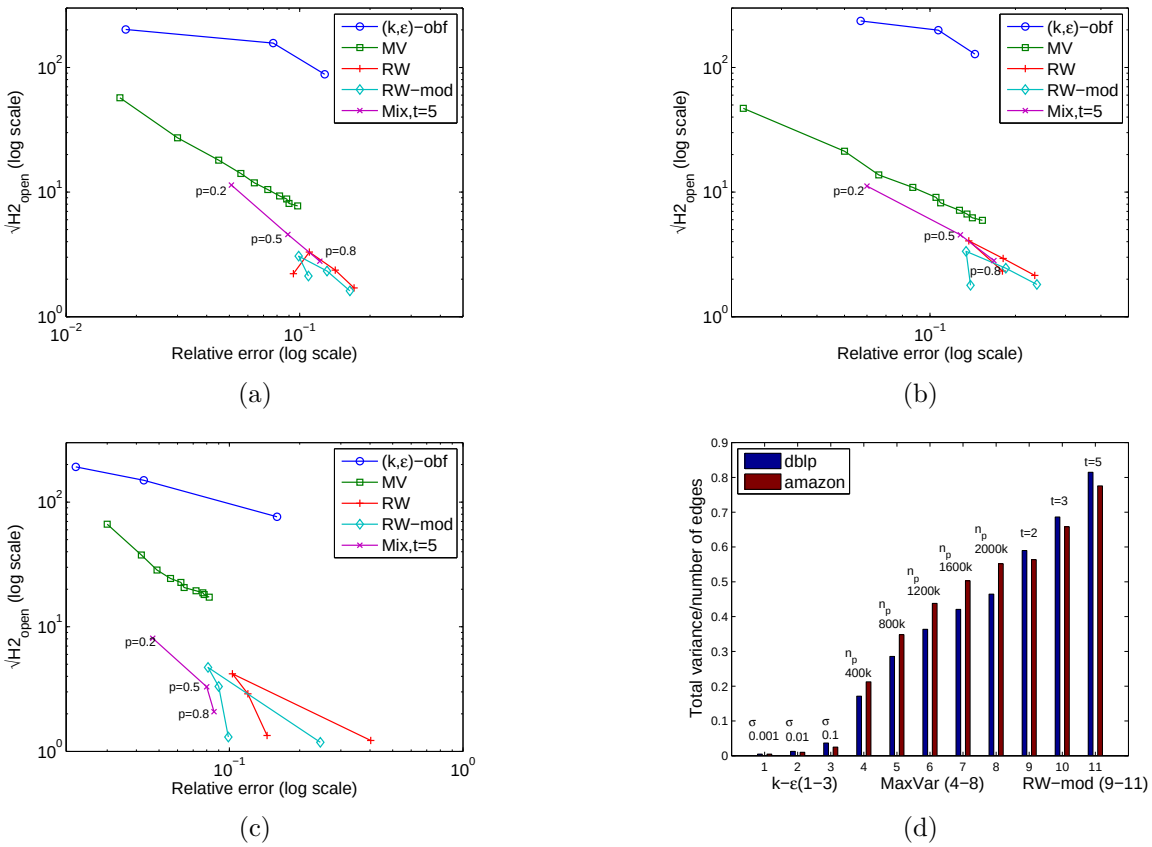


Figure 3.6: Tradeoff (log-log) (a) dblp (b) amazon (c) youtube. (d) Comparison of Total Variance (TV)

less than that of (k, ϵ)-obf (100% for $c = 2$ [11]). *MaxVar* and *RandWalk/RandWalk-mod* have $|E_{G_0} \setminus E_G| \simeq |E_G \setminus E_{G_0}|$ and these two quantities are higher than those of (k, ϵ)-obf where the number of edges is preserved only at small σ. RandWalk and RandWalk-mod do not have many edges preserved due to their rewiring nature. $|E_{G_0} \setminus E_G|$ increases slowly in MaxVar because the edges in G_0 always have positive probabilities. Fig 3.6d compares the normalized total variance (i.e. divided by $|E_{G_0}|$) of three schemes. Again, MaxVar is between (k, ϵ)-obf and RandWalk-mod as shown in inequality (3.11).

3.7 Conclusion

We provide a general view of graph anonymization based on the semantics of edge uncertainty. Via the model of UAM with the constraint of unchanged expected degree for all nodes, we analyze recently proposed schemes and explain why there exists a gap between them by comparing the total degree variance. We propose *MaxVar*, a novel anonymization scheme exploiting two key observations: maximizing the total degree variance while keeping the expected degrees of all nodes unchanged and using nearby potential edges. We also investigate an elegant *Mixture* approach that together with *MaxVar* fill the gap between (k, e)-*obf* and *RandWalk*. Furthermore, we promote the usage of incorrectness measure for privacy assessment in a new quantifying framework rather than Shannon entropy and min-entropy (k-anonymity). The experiments demonstrate the effectiveness and efficiency of our methods. Our work may incite several directions for future research including (1) novel constructions of uncertain graphs based on the model of UAM (2) deeper analysis on the privacy-utility relationships in MaxVar (e.g. explaining the near linear and near quadratic curves) (3) study on directed and bipartite graphs.

In the next chapter, we tackle the problem of graph anonymization again, but within the differential privacy framework.

Table 3.5: (k, ϵ)-*obf*

σ	$H1$	$H2_{open}$	S_{NE}	S_{AD}	S_{MD}	S_{DV}	S_{CC}	S_{PL}	S_{APD}	S_{ED}	S_{CL}	S_{Diam}	rel.err
dblp	199	125302	1049866	6.62	343	100.15	0.306	2.245	7.69	9	7.46	20	
0.001	72.9	40712.1	1048153	6.61	316.0	97.46	0.303	2.244	7.74	9.4	7.50	20.0	**0.018**
0.01	41.1	24618.2	1035994	6.53	186.0	86.47	0.294	2.248	7.82	9.5	7.59	19.8	**0.077**
0.1	19.7	7771.4	991498	6.25	164.9	64.20	0.284	2.265	8.08	10.0	7.85	20.0	**0.128**
amazon	153	113338	925872	5.53	549	33.20	0.205	2.336	12.75	16	12.10	44	
0.001	55.7	55655.9	924321	5.52	479.1	31.73	0.206	2.340	12.14	15.2	11.65	33.2	**0.057**
0.01	34.5	39689.8	915711	5.47	299.7	27.18	0.220	2.348	12.40	15.6	11.91	32.4	**0.101**
0.1	19.2	16375.4	892140	5.33	253.9	21.87	0.232	2.374	12.52	15.5	12.06	31.4	**0.144**
youtube	978	321724	2987624	5.27	28754	2576.0	0.0062	2.429	6.07	8	6.79	20	
0.001	157.2	36744.6	2982974	5.26	28438	2522.6	0.0062	2.416	6.24	8.0	6.01	19.5	**0.022**
0.01	80.0	22361.7	2940310	5.18	26900	2282.6	0.0061	2.419	6.27	8.0	6.04	19.0	**0.043**
0.1	23.4	5806.9	2624066	4.62	16353	970.8	0.0070	2.438	6.59	8.1	6.36	20.4	**0.160**

Table 3.6: *RandWalk* and *RandWalk-mod*

t	$H1$	$H2_{open}$	S_{NE}	S_{AD}	S_{MD}	S_{DV}	S_{CC}	S_{PL}	S_{APD}	S_{ED}	S_{CL}	S_{Diam}	rel.err
`dblp`	199	125302	1049866	6.62	343	100.15	0.306	2.245	7.69	9	7.46	20	
(RW) 2	10.0	4.9	1001252	6.32	309.3	86.16	0.152	2.197	7.43	9.1	7.20	19.7	**0.094**
3	11.8	10.9	1048129	6.61	315.4	98.04	0.107	2.155	7.08	8.7	6.88	17.8	**0.110**
5	11.7	5.6	1049484	6.62	321.6	100.77	0.065	2.148	6.79	8.0	6.62	16.4	**0.142**
10	11.9	2.9	1049329	6.62	329.2	103.06	0.030	2.144	6.54	8.0	6.40	14.3	**0.171**
(RW-mod) 2	11.8	4.5	1049921	6.62	327.0	105.3	0.093	2.110	7.75	9.7	7.48	23.0	**0.109**
3	11.9	9.4	1049877	6.62	343.3	105.1	0.071	2.117	7.32	9.0	7.10	20.4	**0.099**
5	12.0	5.4	1049781	6.62	340.5	105.1	0.044	2.115	6.95	8.4	6.76	18.3	**0.131**
10	11.9	2.6	1049902	6.62	340.0	105.3	0.021	2.116	6.59	8.0	6.44	16.0	**0.164**
`amazon`	153	113338	925872	5.53	549	33.20	0.205	2.336	12.75	16	12.10	44	
(RW) 2	5.7	5.4	861896	5.15	274.9	23.11	0.148	2.337	10.70	13.8	10.19	38.7	**0.180**
3	10.0	16.5	923793	5.52	495.6	32.72	0.113	2.282	10.33	13.1	9.87	34.1	**0.137**
5	10.4	8.6	925185	5.53	507.7	33.52	0.080	2.276	9.45	12.1	9.07	29.6	**0.181**
10	10.2	4.6	925748	5.53	498.1	34.37	0.046	2.273	8.55	10.5	8.25	25.7	**0.234**
(RW-mod) 2	9.8	3.2	925672	5.53	255.1	37.61	0.099	2.246	12.02	15.5	11.40	43.2	**0.139**
3	9.9	11.2	925532	5.53	535.3	37.32	0.082	2.254	10.89	14.0	10.38	37.9	**0.134**
5	9.7	6.0	926163	5.53	522.8	37.42	0.059	2.252	9.83	12.5	9.40	33.0	**0.185**
10	9.9	3.3	925809	5.53	491.4	37.45	0.035	2.251	8.76	11.0	8.44	28.7	**0.238**
`youtube`	978	321724	2987624	5.27	28754	2576.0	0.0062	2.429	6.07	8	6.79	20	
(RW) 2	13.4	1.5	2636508	4.65	19253.8	1139.7	0.022	2.191	6.18	7.9	5.93	23.5	**0.403**
3	23.8	17.6	2982204	5.26	26803.6	2389.6	0.004	2.108	5.73	7.0	5.52	18.0	**0.103**
5	24.6	8.4	2985967	5.26	26018.7	2340.0	0.005	2.106	5.55	7.0	5.38	16.3	**0.120**
10	21.9	1.8	2984115	5.26	24695.8	2099.4	0.009	2.100	5.49	6.9	5.33	18.7	**0.145**
(RW-mod) 2	26.4	1.4	2987228	5.26	23829.7	2578.5	0.018	2.053	6.27	8.0	6.02	22.1	**0.245**
3	26.9	22.3	2988011	5.27	28611.5	2579.7	0.005	2.077	5.75	7.2	5.54	19.0	**0.081**
5	26.1	11.0	2987479	5.26	28619.3	2581.4	0.005	2.076	5.61	7.0	5.44	18.3	**0.090**
10	26.3	1.7	2987475	5.26	28432.2	2579.9	0.008	2.073	5.58	7.0	5.41	18.8	**0.099**

Table 3.7: Effectiveness of *MaxVar* (k denotes one thousand)

n_p	$H1$	$H2_{open}$	S_{NE}	S_{AD}	S_{MD}	S_{DV}	S_{CC}	S_{PL}	S_{APD}	S_{ED}	S_{CL}	S_{Diam}	rel.err
`dblp`	199	125302	1049866	6.62	343	100.15	0.306	2.245	7.69	9	7.46	20	
200k	59.7	3257.2	1049774	6.62	342.3	100.73	0.279	2.213	7.66	9.3	7.43	19.5	**0.017**
400k	40.7	744.0	1049813	6.62	343.5	101.26	0.255	2.189	7.56	9.1	7.33	18.9	**0.030**
600k	32.1	325.7	1050066	6.62	343.4	101.73	0.235	2.173	7.46	9.0	7.25	17.7	**0.045**
800k	29.5	199.2	1049869	6.62	345.9	102.07	0.219	2.163	7.45	9.0	7.24	17.0	**0.056**
1000k	27.0	140.7	1049849	6.62	345.4	102.29	0.205	2.155	7.34	9.0	7.15	17.0	**0.064**
`amazon`	153	113338	925872	5.53	549	33.20	0.205	2.336	12.75	16	12.10	44	
200k	30.2	2209.1	925831	5.53	551.5	33.83	0.197	2.321	12.38	16.1	11.72	40.5	**0.022**
400k	22.8	452.4	925928	5.53	550.2	34.40	0.182	2.306	11.88	15.3	11.28	37.1	**0.050**
600k	17.8	188.4	925802	5.53	543.9	34.79	0.167	2.296	11.60	15.0	11.04	36.9	**0.066**
800k	17.2	118.8	925660	5.53	550.0	35.11	0.154	2.289	11.33	14.4	10.81	34.5	**0.087**
1000k	15.2	82.4	925950	5.53	551.8	35.43	0.142	2.282	11.13	14.1	10.62	31.8	**0.105**
`youtube`	978	321724	2987624	5.27	28754	2576.0	0.0062	2.429	6.07	8	6.79	20	
600k	114.4	4428.8	2987898	5.27	28759	2576	0.0065	2.373	6.19	7.8	5.97	18.6	**0.030**
1200k	84.2	1419.2	2987342	5.26	28754	2576	0.0064	2.319	6.02	7.2	5.82	17.9	**0.042**
1800k	71.4	814.4	2987706	5.27	28745	2577	0.0062	2.287	5.97	7.1	5.78	17.2	**0.049**
2400k	65.3	595.5	2987468	5.26	28749	2577	0.0060	2.265	5.96	7.1	5.77	16.6	**0.056**
3000k	62.8	513.7	2987771	5.27	28761	2578	0.0058	2.251	5.89	7.1	5.71	16.4	**0.062**

Table 3.8: *MaxVar* vs. (k, ϵ)-*obf*, *RandWalk* and *RandWalk-mod* (lower tradeoff is better)

	PRIVACY							UTILITY	
graph	$H1$	$H2_{open}$	$\lvert E_{G_0} \setminus E_G \rvert$	$\lvert E_G \setminus E_{G_0} \rvert$	$\epsilon(k=30)$	$\epsilon(k=50)$	$\epsilon(k=100)$	rel.err	tradeoff
`dblp`	199	125302			0.00238	0.00393	0.00694		
$\sigma = 0.001$	72.9	40712.1	6993.0	5280.2	0.00039	0.00122	0.00435	0.018	**3.61**
$\sigma = 0.01$	41.1	24618.2	19317.3	5444.9	0.00051	0.00062	0.00082	0.077	**12.03**
$\sigma = 0.1$	19.7	7771.4	65285.1	6916.8	0.00179	0.00199	0.00245	0.128	**11.33**
(MV) $n_p = 200k$	59.7	3257.2	94508.0	94416.5	0.00033	0.00077	0.00152	0.017	**0.99**
(MV) $n_p = 600k$	32.1	325.7	246155.6	246355.3	0.00017	0.00029	0.00085	0.045	**0.82**
(RW) $t = 2$	10.0	4.9	615966.2	567352.2	0.00318	0.00439	0.00789	0.094	**0.21**
(RW) $t = 5$	11.7	5.6	754178.1	753796.3	0.00271	0.00386	0.00689	0.142	**0.34**
(RW-mod) $t = 2$	11.8	4.5	719361.2	719416.5	0.00073	0.00135	0.00252	0.109	**0.23**
(RW-mod) $t = 5$	12.0	5.4	784872.3	784786.8	0.00057	0.00113	0.00228	0.131	**0.30**
`amazon`	153	113338			0.00151	0.00218	0.00456		
$\sigma = 0.001$	55.7	55655.9	6158.9	4607.4	0.00048	0.00119	0.00293	0.065	**13.40**
$\sigma = 0.01$	34.5	39689.8	14962.0	4801.3	0.00038	0.00052	0.00066	0.114	**21.33**
$\sigma = 0.1$	19.2	16375.4	39382.6	5650.3	0.00068	0.00102	0.00190	0.145	**18.46**
(MV) $n_p = 200k$	30.2	2209.1	104800.9	104759.9	0.00023	0.00032	0.00065	0.022	**1.03**
(MV) $n_p = 600k$	17.8	188.4	266603.7	266533.7	0.00015	0.00023	0.00047	0.066	**0.91**
(RW) $t = 2$	5.7	5.4	649001.0	585025.5	0.00213	0.00338	0.00550	0.180	**0.42**
(RW) $t = 5$	10.4	8.6	629961.8	629274.9	0.00146	0.00239	0.00423	0.181	**0.53**
(RW-mod) $t = 2$	9.8	3.2	725440.1	725239.9	0.00048	0.00073	0.00133	0.139	**0.25**
(RW-mod) $t = 5$	9.7	6.0	671694.2	671985.4	0.00038	0.00058	0.00137	0.185	**0.45**
`youtube`	978	321724			0.00291	0.00402	0.00583		
$\sigma = 0.001$	157.2	36744.6	19678.5	15028.5	0.00143	0.00232	0.00421	0.022	**4.28**
$\sigma = 0.01$	80.0	22361.7	62228.6	14914.3	0.00060	0.00105	0.00232	0.043	**6.38**
$\sigma = 0.1$	23.4	5806.9	378566.0	15007.5	0.00038	0.00052	0.00074	0.160	**12.20**
(MV) $n_p = 600k$	114.4	4428.8	213097.3	213371.4	0.00047	0.00063	0.00108	0.030	**2.00**
(MV) $n_p = 1800k$	71.4	814.4	521709.9	521791.6	0.00040	0.00052	0.00090	0.049	**1.38**
(RW) $t = 2$	13.4	1.5	2836169.3	2485053.4	0.00319	0.00425	0.00623	0.403	**0.50**
(RW) $t = 5$	24.6	8.4	2468068.6	2466411.3	0.00304	0.00408	0.00598	0.120	**0.35**
(RW-mod) $t = 2$	26.4	1.4	2863112.1	2862716.3	0.00159	0.00226	0.00355	0.245	**0.29**
(RW-mod) $t = 5$	26.1	11.0	2467414.9	2467269.5	0.00153	0.00322	0.00159	0.090	**0.30**

Chapter 4

Network Structure Release under Differential Privacy

Contents

4.1 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 [73], 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 [4,46]. Therefore, many graph anonymization schemes have been proposed [23, 55, 96, 112, 113]. The main techniques used in those works are based on random edge manipulation or deterministic transformations to satisfy k-anonymity [95]. Another popular class of schemes relies on uncertainty semantics [11, 76].

In this chapter, 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 [63]. 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 [21, 89, 100, 101, 105].

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 [89, 100] use *dK-series* [60] 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 [101] by graph spectral analysis. The most recent works [21,105] even reduce the sensitivity of graph to $O(\ln n)$[10]. While *Density Explore Reconstruct* (DER) [21] employs a data-dependent quadtree to summarize the adjacency matrix into a counting tree, Xiao et al. [105] propose to use *Hierarchical Random Graph* (HRG) [24] to encode graph structural information in terms of edge probabilities. A common disadvantage of the state-of-the-art DER [21] and HRG-MCMC [105] 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 [89, 100], Kronecker graph model [65], spectrum-based [101], DER [21] and HRG-MCMC [105] 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 tradeoff. 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 [7, 47] where consistency means the noisy output must satisfy certain constraints, e.g. non-negative integrality [7] or ordered sequence [47].

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

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

enough ϵ. By considering the adjacency matrix as a sparse dataset, TmF leverages the high-pass filtering idea in [26] 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 [89, 100, 101] 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 [77], we found the scheme *EdgeFlip* by Mülle et al. [69] which uses *edge flipping* technique. By a deeper analysis, we prove that EdgeFlip also provides an upper bound $O(\ln n)$ for privacy budget ϵ and it makes the expected 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 $O(m \log n)$ and may be of independent interest for the community detection problem [38]. Finally, we present 1K-series scheme which uses the degree sequence instead of the degree distribution [100] and also runs in linear time.

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 [52].

- 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.

The rest of this chapter is organized as follows. Section 4.2 explains more about the key concepts and mechanisms of differential privacy. It also analyzes key challenges of ϵ-DP graph release. TmF scheme is described in Section 4.3. In Section 4.4, we present the deeper analysis and a linear algorithm for EdgeFlip. We review HRG model, the limitations of full space exploration and then propose a faster technique HRG-FixedTree in section 4.5. The 1K-series method and the comparison of schemes are described in section 4.6. In Section 4.7, six competitors are evaluated on a variety of graphs. Finally, Section 4.8 concludes the chapter.

4.2 Preliminaries

In this section, we explain in more detail about key concepts and mechanisms of differential privacy. Then, we present the key challenges of graph release under differential privacy.

4.2.1 Edge Differential Privacy for Graphs

Essentially, ϵ-differential privacy (ϵ-DP) [34] 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 4.1. *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 [34] and Exponential mechanism [63] 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 4.1. *(Laplace mechanism [34]) For any function $f : G \to \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(G) = f(G) + \langle Lap_1(\frac{\Delta f}{\epsilon}), ..., Lap_d(\frac{\Delta f}{\epsilon}) \rangle \tag{4.1}$$

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 [40] 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 1K-series scheme (Section 4.6) and LouvainDP scheme in the next chapter.

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 4.2. *(Exponential mechanism [62]) 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 4.3).

Theorem 4.3. *(Sequential and parallel compositions [63]) 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.

4.2.2 Challenges of Graph Release under Differential Privacy

In this section, we discuss two key challenges of the problem addressed in this chapter: *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 [34], 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 4.3 and 4.4), 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 (Section 4.6), the size of the output space of $\tilde{G}$ is approximated by the number of ways to rewire the node stubs [73], i.e. $(2m)!$. If we try to keep the expected 2K-series unchanged [88, 100], 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 [34]). 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 [88], 200-2,000 in [100] or 4,474 in [101].

Another important justification of $\epsilon = \ln n$ is the *edge re-identification risk* quantified by *ρ-differential identifiability* (ρ-DI) [52]. 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, the expected edit distance between $\tilde{G}$ and G_1 is one at a certain value of ϵ (a.k.a upper bounds of privacy budget to obtain the best utility, see Theorems 4.6 and 4.9). 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.

4.3 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 [73, Table 2]. TmF then uses an idea similar to High-pass Filter in [26] 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.

4.3.1 Overview

In [26], 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 [26] 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 4.2.1 is preferred.

Compared to the High-pass filtering technique applied to contingency tables [26], 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 [26] 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.

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 [105]). 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. 4.1 depicts the processes of Naive, Laborious filtering and TmF.

The distributions for noisy values of 1-cells and 0-cells are shown in Figures 4.2 and 4.3. 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.

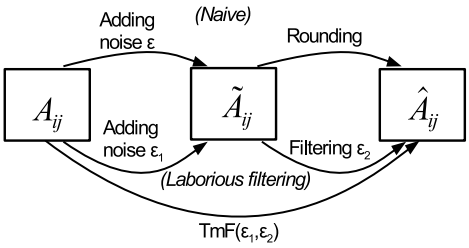


Figure 4.1: TmF algorithm

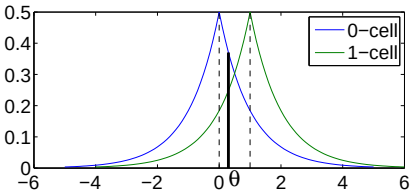


Figure 4.2: $0 < \theta < 1$

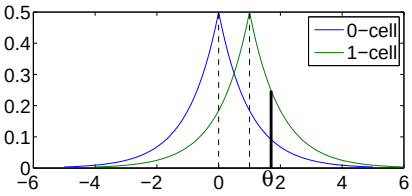


Figure 4.3: $1 \leq \theta$

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 expected number of passing 1-cells is

$$n_1 = m \int_{\theta}^{+\infty} \frac{\epsilon_1}{2} \exp(-\epsilon_1 |x-1|) dx = \frac{m}{2}(2 - e^{-\epsilon_1(1-\theta)}) \tag{4.2}$$

The expected number of passing 0-cells is

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

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

$$\theta = \frac{1}{2\epsilon_1} \ln(\frac{n(n-1)}{2m} - 1) + \frac{1}{2} \tag{4.3}$$

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

$$n_1 = \frac{m}{2} e^{-\epsilon_1(\theta - 1)} \tag{4.4}$$

The number of passing 0-cells is

$$n_0 = (\frac{n(n-1)}{2} - m) \frac{1}{2} e^{-\epsilon_1 \theta} \tag{4.5}$$

The value of θ is

$$\theta = \frac{1}{\epsilon_1} \ln(\frac{n(n-1)}{4m} + \frac{1}{2}(e^{\epsilon_1} - 1)) \tag{4.6}$$

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,

$$\theta = 1 \leftrightarrow \epsilon_t = \ln(\frac{n(n-1)}{2m} - 1) \tag{4.7}$$

4.3.2 TmF Algorithm

Algorithm 4 Top-m Filter

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

1: $\tilde{G} \leftarrow \varnothing$
2: // compute $\tilde{m}$ and θ
3: $\tilde{m} = m + Lap(1/\epsilon_2)$
4: $\epsilon_t = \ln(\frac{n(n-1)}{2\tilde{m}} - 1)$
5: **if** $\epsilon_1 > \epsilon_t$ **then**
6: $\quad \theta = \frac{1}{2\epsilon_1} \ln(\frac{n(n-1)}{2\tilde{m}} - 1) + \frac{1}{2}$
7: **else**
8: $\quad \theta = \frac{1}{\epsilon_1} \ln(\frac{n(n-1)}{4\tilde{m}} + \frac{1}{2}(e^{\epsilon_1} - 1))$
9: // process 1-cells
10: $n_1 = 0$
11: **for** $A_{ij} = 1$ **do**
12: $\quad$ compute $\tilde{A}_{ij} = A_{ij} + Lap(1/\epsilon_1)$
13: $\quad$ **if** $\tilde{A}_{ij} > \theta$ **then**
14: $\quad\quad$ add edge (i, j) to $\tilde{G}$
15: $\quad\quad$ n_1 ++
16: // process 0-cells
17: $n_0 = \tilde{m} - n_1$
18: **while** $n_0 > 0$ **do**
19: $\quad$ pick an edge $(i, j) \notin E$ uniformly at random
20: $\quad$ **if** $\tilde{G}$ does not contain (i, j) **then**
21: $\quad\quad$ add edge (i, j) to $\tilde{G}$
22: $\quad\quad$ n_0- -
23: **return** $\tilde{G}$

Algorithm 4 shows steps of Top-m-Filter in which we replace all m by $\tilde{m}$ [11]. 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 4.4. *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 4.4 makes sense if we consider the complexity $O(n^2)$ of the state-of-the-art DER [21] and HRG-MCMC [105].

[11] Note that in [26], Cormode et al. made a minor error by using the number of non-zero cells m_1 in public form. We can fix it by reserving a small ϵ_2 to mask m_1, similar to what has been done to m in TmF. See more in Appendix A.

4.3.3 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 4.3) is applicable at cell level. So the second step satisfies ϵ_1-DP. By sequential composition (Theorem 4.3), TmF satisfies $(\epsilon_1 + \epsilon_2)$-DP as stated in the following theorem.

Theorem 4.5. *TmF satisfies ϵ-DP where $\epsilon = \epsilon_1 + \epsilon_2$.*

Now we proceed to the more important result: TmF reduces the expected edit distance between $\tilde{G}$ and G to $O(1)$ at $\epsilon_1 = O(\log n)$. The expected edit distance is defined as (to simplify the notation, we omit the expectation operator in the right hand side expression of the following formula)

$$D(G, \tilde{G}) = \frac{1}{2}(|E_G \setminus E_{\tilde{G}}| + |E_{\tilde{G}} \setminus E_G|) \tag{4.8}$$

By the analysis in Section 4.3.1, the expected number of passing 1-cells is n_1, so the expected edit distance $D(G, \tilde{G})$ is $m - n_1$. For a deeper analysis of the effect of ϵ_2 on n_1, see Appendix A.

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.

$$\begin{aligned}
& D(G, \tilde{G}) = \gamma m \\
\leftrightarrow \quad & m - \frac{m}{2}(2 - e^{-\epsilon_1(1-\theta)}) = \gamma m \\
\leftrightarrow \quad & e^{-\epsilon_1(1-\theta)} = 2\gamma \\
\leftrightarrow \quad & \theta = 1 + \frac{1}{\epsilon_1}\ln(2\gamma) \\
\leftrightarrow \quad & \frac{1}{2\epsilon_1}\ln(\frac{n(n-1)}{2m} - 1) + \frac{1}{2} = 1 + \frac{1}{\epsilon_1}\ln(2\gamma) \quad \text{(from (4.3))} \\
\leftrightarrow \quad & \epsilon_1 = \ln(\frac{\frac{n(n-1)}{2m} - 1}{4\gamma^2})
\end{aligned}$$

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 which proves that TmF has the consistency property (cf. discussion in Section 4.1).

Theorem 4.6. *TmF makes the expected edit distance $D(G, \tilde{G})$ decrease to $O(1)$ at $\epsilon_1 \approx 3\ln n$.*

Fig. 4.4 shows the normalized number of passing 1-cells n_1/m as a function of $\epsilon_1/\ln n$ over nine graphs (cf. Table 4.2). 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 [89, 100] and [101] where $\epsilon = O(\sqrt{n})$ or $\epsilon = O(n)$. Interestingly, in HRG-MCMC scheme [105], 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 in Section 4.5.

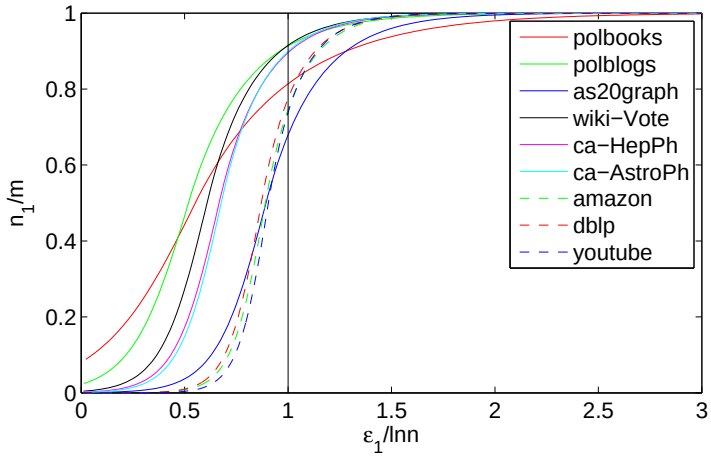


Figure 4.4: n_1/m as a function of $\epsilon_1/\ln n$

4.4 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.

4.4.1 A Tighter Upper Bound for Privacy Budget

EdgeFlip scheme [69] is also a direct publication scheme as TmF. It is presented in Algorithm 5. 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 satisfy 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 4.7. *(EdgeFlip privacy [69]) 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 4.8. *(Expected number of edges [69])* $E[|V'|] = (1 - s)m + \frac{n(n-1)}{4}s$

By considering all edges in G (there are $\frac{n(n-1)}{2}$ such edges), Algorithm 5 incurs a quadratic complexity and needs to be redesigned for large graphs. First, we compute the expected edit distance $D(G, \tilde{G}) = \frac{1}{2}(|E_G \setminus E_{\tilde{G}}| + |E_{\tilde{G}} \setminus E_G|)$ where

$$|E_G \setminus E_{\tilde{G}}| = |m - (1 - \frac{s}{2})m| = \frac{sm}{2} \tag{4.9}$$

$$|E_{\tilde{G}} \setminus E_G| = \left(\frac{n(n-1)}{2} - m\right)\frac{s}{2} \tag{4.10}$$

So, we get $D(G, \tilde{G}) = \frac{n(n-1)s}{8}$. From Theorem 4.7, $s = \frac{2}{e^{\epsilon}+1}$. Therefore, the relation between the expected edit distance and the privacy budget in EdgeFlip is as follows

$$D(G, \tilde{G}) = \frac{n(n-1)}{4(e^{\epsilon} + 1)} \tag{4.11}$$

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

Algorithm 5 Original EdgeFlip(G, s) [69]

Input: undirected graph $G = (V, E)$, privacy parameter s
Output: anonymized graph $\tilde{G}$

1: $\tilde{G} = (V, E')$
2: construct adjacency matrix A from E of G
3: initialize the adjacency matrix A' for E' of $\tilde{G}$
4: **for** $a_{ij} \in A$ with $i < j$ **do**
5: **if** a_{ij} is chosen with probability $1 - s$ **then**
6: set $a'_{ij} = a'_{ji} = a_{ij}$ in A' of $\tilde{G}$
7: **else**
8: **if** 0 is chosen with probability 0.5 **then**
9: set $a'_{ij} = a'_{ji} = 0$ in A' of $\tilde{G}$
10: **else**
11: set $a'_{ij} = a'_{ji} = 1$ in A' of $\tilde{G}$
12: **return** $\tilde{G}'$

Theorem 4.9. *(EdgeFlip edit distance) EdgeFlip can reduce the expected 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 expected 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.

4.4.2 A Partially Linear Implementation

To make EdgeFlip runnable on large graphs, we propose a partially linear-time version as in Algorithm 6 where we process 1-cells and 0-cells separately. Note that Algorithm 6 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 [69] (Algorithm 5).

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 short-cut 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

Algorithm 6 Partially linear-time EdgeFlip(G, s, ϵ_2)

Input: undirected graph $G = (V, E)$, privacy parameter s
Output: anonymized graph $\tilde{G}$

$\tilde{G} \leftarrow \varnothing$, $\epsilon_2 = 0.1$
$\tilde{m} = m + Lap(1/\epsilon_2)$
$\epsilon_t = \ln(\frac{n(n-1)}{2\tilde{m}} - 1)$
$\epsilon = \ln(\frac{2}{s} - 1) - \epsilon_2$
$\tilde{s} = \frac{2}{e^{\epsilon}+1}$
if $\epsilon + \epsilon_2 \leq \epsilon_t$ **then return** EdgeFlip$(G, \tilde{s})$ [69]
// process 1-cells
for $A_{ij} = 1$ **do**
    add edge (i, j) to $\tilde{G}$ with prob. $1 - \tilde{s}/2$
// process 0-cells
$n_0 = (\frac{n(n-1)}{2} - \tilde{m})\frac{\tilde{s}}{2}$
while $n_0 > 0$ **do**
    pick an edge $(i, j) \notin E_G$ uniformly at random
    if $\tilde{G}$ does not contain (i, j) **then**
        add edge (i, j) to $\tilde{G}$
        n_0- -
return $\tilde{G}'$

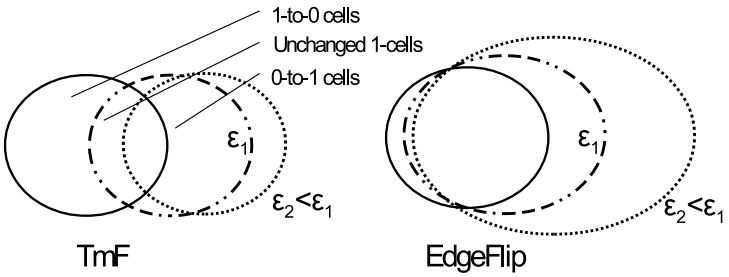


Figure 4.5: TmF vs. EdgeFlip

least $2m$, so we cannot filter the top-m 1-cells. Figures 4.5 and 4.6 visualize the difference 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 tradeoff between the smaller upper bounds than TmF (Theorem 4.9) and the infeasibility at $s \geq \frac{4m}{n(n-1)}$ (corresponding to $\epsilon \leq \ln(\frac{n(n-1)}{2m} - 1)$.

4.5 HRG-based Schemes for Large Graphs

In this section, we propose *HRG-FixedTree* (section 4.5.2) for the problem of sampling dendrogram in HRG-MCMC [105]. It satisfies ϵ-DP and is runnable on large graphs by reducing the complexity to $O(n \log n)$.

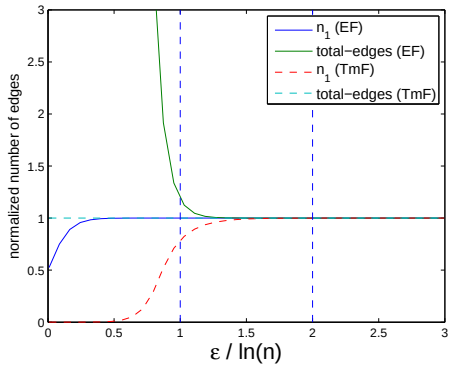


Figure 4.6: The number of passing 1-cells and the total of edges in EdgeFlip and TmF (on `amazon` graph)

4.5.1 HRG-MCMC and Limitations

Hierarchical Random Graph (HRG) [24] is a graph summary structure of size $O(n)$. It is a binary tree with n leaves being graph nodes and $n-1$ internal nodes. Fig. 4.7 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

$$\mathcal{L}(T, \{p_r\}) = \prod_{r \in T} p_r^{e_r} (1-p_r)^{n_{Lr} n_{Rr} - e_r} \tag{4.12}$$

The log-likelihood of T is

$$\log \mathcal{L}(T, \{p_r\}) = -\sum_{r \in T} n_{Lr} n_{Rr} h(p_r) \tag{4.13}$$

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.

Clauset et al. [24] propose *HRG-Fit* using Markov Chain Monte Carlo (MCMC) to learn an ensemble of high-likelihood dendrograms. Given the current dendrogram T, a neighbor dendrogram T' is proposed by randomly selecting an internal node r (other than the root node) and performing 1 of 2 possible subtree reorderings. T' is accepted with probability

$$\min(1, \frac{\exp(\log \mathcal{L}(\mathcal{T}'))}{\exp(\log \mathcal{L}(\mathcal{T}))}) \tag{4.14}$$

Xiao et at. [105] employ HRG model to address the problem of graph release under differential privacy. Their scheme 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 the exponential mechanism (Definition 4.2), 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. The score function u of dendrogram T is the log-likelihood of T, i.e. $u(G, T) = \log \mathcal{L}(\mathcal{T})$.

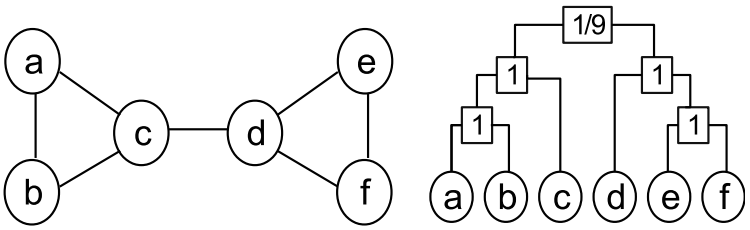


Figure 4.7: (left) graph G (right) a dendrogram T

To satisfy ϵ-DP, the acceptance probability in [105] is

$$\min(1, \frac{\exp(\frac{\epsilon_1}{2\Delta u} \log \mathcal{L}(\mathcal{T}'))}{\exp(\frac{\epsilon_1}{2\Delta u} \log \mathcal{L}(\mathcal{T}))}) \tag{4.15}$$

where Δu is the global sensitivity of a dendrogram's log-likelihood, and $\Delta u \approx ln(\frac{n^2}{4}) \approx 2 \ln n$. By comparing Formula 4.14 with Formula 4.15, we see that HRG-Fit in [24] is $2\Delta u$-DP in nature. However, even at $\epsilon = 2\Delta u = 4 \ln n$, i.e. when HRG-MCMC becomes HRG-Fit, the sample graphs reconstructed are not good enough as we will see in Section 4.7.

Limitations of MCMC on full HRG. 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 of $O(n^2 \log n)$-complexity. In the subsequent section 4.5.2, we present *HRG-FixedTree* with complexity of $O(n \log n)$, runnable on large graphs. We also present a fast sampling technique on dendrograms in section 4.5.3.

4.5.2 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.

Fig. 4.8 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 7 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.

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 8 is called in the line 3 of Algorithm 7. 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. 4.8 for

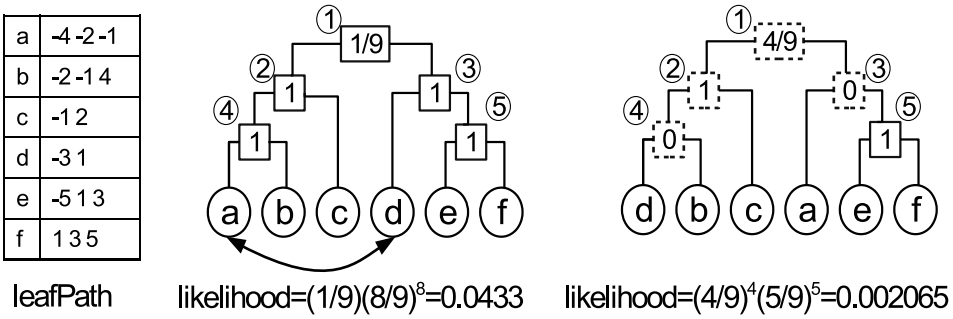


Figure 4.8: HRG-FixedTree

Algorithm 7 HRG-FixedTree

Input: input graph $G = (V, E)$, fixed dendrogram structure D, privacy budget ϵ_1
Output: 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: // until equilibrium is reached
6: **return** a sampled partition $S_{sampled} = S_i$

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 $listP$. 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 4.13). 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 4.10. *HRG-FixedTree runs in $O(k.m\log n)$, where $k = 1000$.*

Algorithm 8 FastSwap

Input: input graph $G = (V, E)$, fixed dendrogram structure D, permutation S, $\mathcal{L}(S)$, two nodes u, v

Output: log-likelihood $\mathcal{L}(S')$ where S' is S with u, v swapped

find affected internal nodes $listP$
for each node $r \in listP$ **do**
    $\mathcal{L}(S) = \mathcal{L}(S) + n_{Lr} n_{Rr} h(p_r)$
remove u and v from S
swap u, v in $leafPath$
add u and v back to S
for each node $r \in listP$ **do**
    update e_r and p_r
    $\mathcal{L}(S) = \mathcal{L}(S) - n_{Lr} n_{Rr} h(p_r)$
update pointers of u and v

4.5.3 Fast Sampling From Dendrogram

Apart from costly MCMC of $1000n$ steps, another bottle-neck in HRG-MCMC [105] 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. 4.7 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 4.11. *From any dendrogram generated by HRG-MCMC or HRG-FixedTree, a sample graph can be generated in $O(n \log n)$.*

4.6 1K-series Scheme

4.6.1 Algorithm

First coined by Mahadevan et al. [60], the term dK-series specifies all degree correlations within d-sized subgraphs of a given graph. So, the term 0k-series represents the average degree of all nodes; the term 1K-series denotes the degree distribution; the term 2K-series denotes the graph's joint degree distribution and so on. For example, the graph in Fig. 4.7 has {0K: $\bar{d} = 2.33$}, {1K: P(2)=4, P(3)=2} and {2K: P(2,2)=2 P(2,3)=4}. In this section, we describe a simple version of *1k-Series* scheme which is slightly different from [100]. As shown in [100], the differentially private 2K-series scheme is only better than 1K-series with weak privacy enforcement (very large ϵ value). By TmF and EdgeFlip we know that the upper-bound for ϵ is only $O(\log n)$, so we focus on 1K-series.

Algorithm 9 outlines the main steps of 1K-series scheme. Our 1K-series scheme is different from DP-1K of [100] in several aspects. First, we use geometric mechanism (see section 4.2) because the degree values are always integral. Second, our 1K-series uses the degree sequence having the global sensitivity of 2 while DP-1K relies on the degree distribution with the global sensitivity of 4. We regenerate noisy graphs from the noisy degree sequence so that the node ids are kept intact for the cut queries (see section 4.7.1). The following example shows the difference between the degree distribution and the degree sequence.

Example 4.1. The degree distribution of the graph in Fig. 4.7 is {0,0,4,2,0,0}, i.e. we have no nodes of degree 0, 1, 4 or 5, 4 nodes of degree 2 and 2 nodes of degree 3. The degree sequence is {2,2,3,3,2,2}. If we remove the edge *(a,b)*, the new degree distribution is {0,2,2,2,0,0}, so the L1-distance is 4. The degree sequence, on the other hand, becomes {1,1,3,3,2,2}, so the L1-distance is 2.

After adding noise (lines 1 to 3), we compute the sum s of noisy degrees (line 4) and sort $\{\tilde{d}(u)\}$. We adjust negative noisy degrees (lines 6 to 9) so that all of them are positive and their sum is still equal to s. Finally, line 10 regenerates a noisy sample graph $\tilde{G}$ from $\{\tilde{d}(u)\}$ using the configuration model [73].

Algorithm 9 1K-series

Input: input graph $G = (V, E)$, privacy budget ϵ
Output: anonymized graph $\tilde{G}$

1: $\alpha = \exp(-\epsilon/2)$
2: get degree sequence $\{d(u)\}, u \in V$
3: add geometric noise $\tilde{d}(u) = d(u) + Geom(\alpha), u \in V$
4: $s = \sum_u \tilde{d}_u$
5: sort $\{\tilde{d}_u\}$
6: $\forall u \in V$, $r_u = \tilde{d}_u/s$ if $\tilde{d}_u > 0$, $r_u = 1/s$ otherwise
7: $c = \sum_u \lceil r_u.s \rceil$
8: $n_c = n - (c - s)$
9: $\tilde{d}_u = \lceil r_u.s \rceil$ if $u \leq n_c$, $\tilde{d}_u = \lceil r_u.s \rceil - 1$ otherwise
10: generate $\tilde{G}$ from $\{\tilde{d}_u\}$ using the configuration model [73]
11: **return** $\tilde{G}$

4.6.2 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 [21] 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 [21] while the other schemes incur only $O(m)$ for storing the graph G. Table 4.1 shows the comparison of the schemes investigated in this chapter.

Table 4.1: Comparison of schemes

		Model	Runtime	Space	Graph size
Direct publication	TmF	n/a	$O(m)$	$O(m)$	10^6
	EdgeFlip* [69]	n/a	$O(m)$	$O(m)$	10^6
Model-based publication	HRG-MCMC [105]	$O(n)$	$O(n^2)$	$O(m)$	10^4
	HRG-FixedTree	$O(n)$	$O(k.m \log n)$	$O(m)$	10^6
	DER [21]	$O(2^h)$	$O(n^2)$	$O(n^2)$	10^4
	1K-series [100]	$O(n)$	$O(m)$	$O(m)$	10^6

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

4.7 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 [12]. In Table 4.2, $\text{logLK}_{\text{Louvain}}$, $\text{logLK}_{\text{HRG-MCMC}}$, $\text{logLK}_{\text{HRG-Fixed}}$ and $\text{logLK}_{\text{Init}}$ are the log-likelihoods of the *non-private* dendrograms created by Louvain algorithm [9], 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 4.7.1. We assess the effectiveness of TmF in Section 4.7.2. Then the non-private versions of HRG-based schemes and 1K-series are evaluated in Section 4.7.3. Finally, Section 4.7.4 compares the overall performance of all competitors and clarifies the contribution of each utility metric.

4.7.1 Utility Metrics

We reuse the statistics in Section 3.5.2 for utility measurement along with the three following metrics

- Degree distribution: S_{DD} is the normalized degree histogram.
- Distance distribution: S_{PDD} is the normalized node-pair shortest-path histogram.
- Cut queries: $S_{cut}(X, Y)$ is the number of edges between two disjoint node sets X and Y.

We group twelve metrics into three groups: *degree-based metrics* (S_{AD}, S_{MD}, S_{DV}, S_{PL}, S_{DD}), *path-based metrics* (S_{APD}, S_{EDiam}, S_{CL}, S_{Diam}, S_{PDD}) and *other metrics* (S_{CC}, S_{cut}).

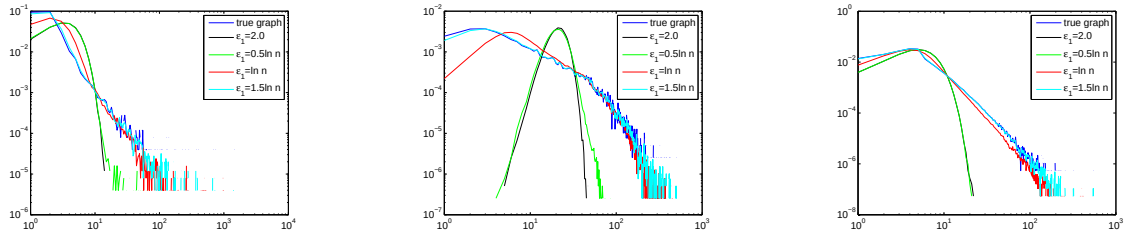
All of the above statistics are taken average over 20 sample graphs. S_{APD}, S_{CL}, S_{EDiam}, S_{Diam} are computed exactly in six small graphs. In *amazon*, *dblp* and *youtube*, we estimate $S_{APD}, S_{CL}, S_{EDiam}$ and S_{Diam} using HyperANF [12]. The relative error (rel.err) for each metric S is computed as $\frac{|S(G)-S_{avg}(\tilde{G})|}{S(G)}$ except S_{DD} and S_{PDD} whose errors are computed as $|S(G) - S_{avg}(\tilde{G})|_1/2$. The number of cut queries is 1,000 and the size of node set does not exceed 500.

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 4.2.2. By ρ-differential identifiability [52], 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.

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

Table 4.2: Graph dataset statistics (k:thousand, m:million)

Dataset	#Nodes	#Edges	$\log LK_{\text{Louvain}}$	$\log LK_{\text{HRG-MCMC}}$	$\log LK_{\text{HRG-Fixed}}$	$\log LK_{\text{Init}}$
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



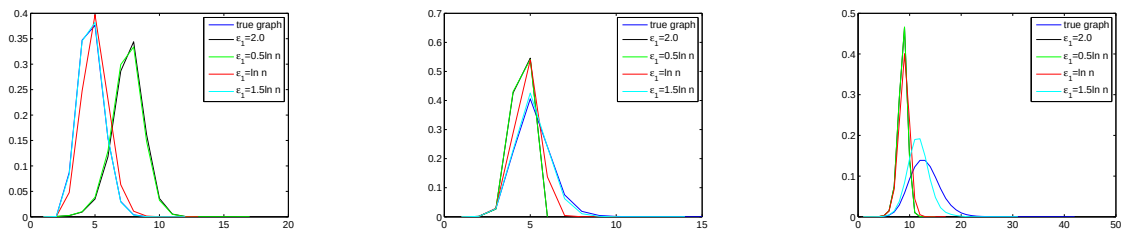
(a) as20graph ($\ln n = 8.8$) (b) ca-AstroPh ($\ln n = 9.8$) (c) amazon ($\ln n = 12.7$)

Figure 4.9: Effectiveness of TmF: degree distributions (log-log scale)

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.

4.7.2 Effectiveness of TmF

We assess the utility of TmF by varying ϵ_1 while fixing $\epsilon_2 = 0.1$. Figures 4.9 and 4.10 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.



(a) as20graph ($\ln n = 8.8$) (b) ca-AstroPh ($\ln n = 9.8$) (c) amazon ($\ln n = 12.7$)

Figure 4.10: Effectiveness of TmF: distance distributions

4.7.3 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 6.2 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) [9] 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 4.3, despite the high-likelihood scores, dendrograms generated by Louvain and non-private HRG based schemes cannot reduce the relative errors significantly. 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. 4.11 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.

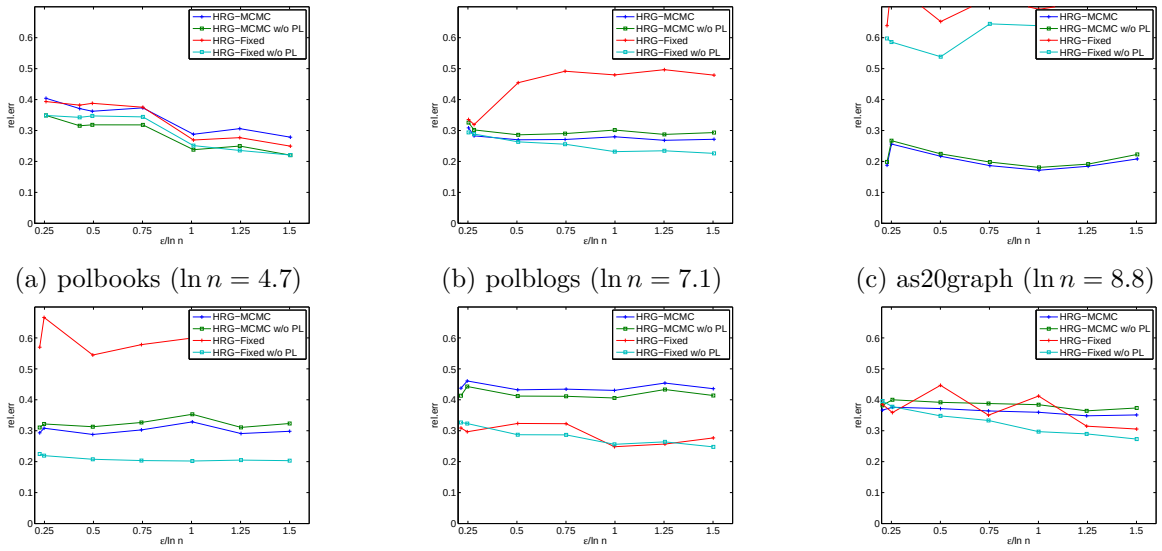
Table 4.3: Relative error of *non-private* 1K-series and HRG-based schemes

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

4.7.4 Comparative Evaluation

We report the comparisons between the competitors in Fig. 4.12. 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. 4.4), 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



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

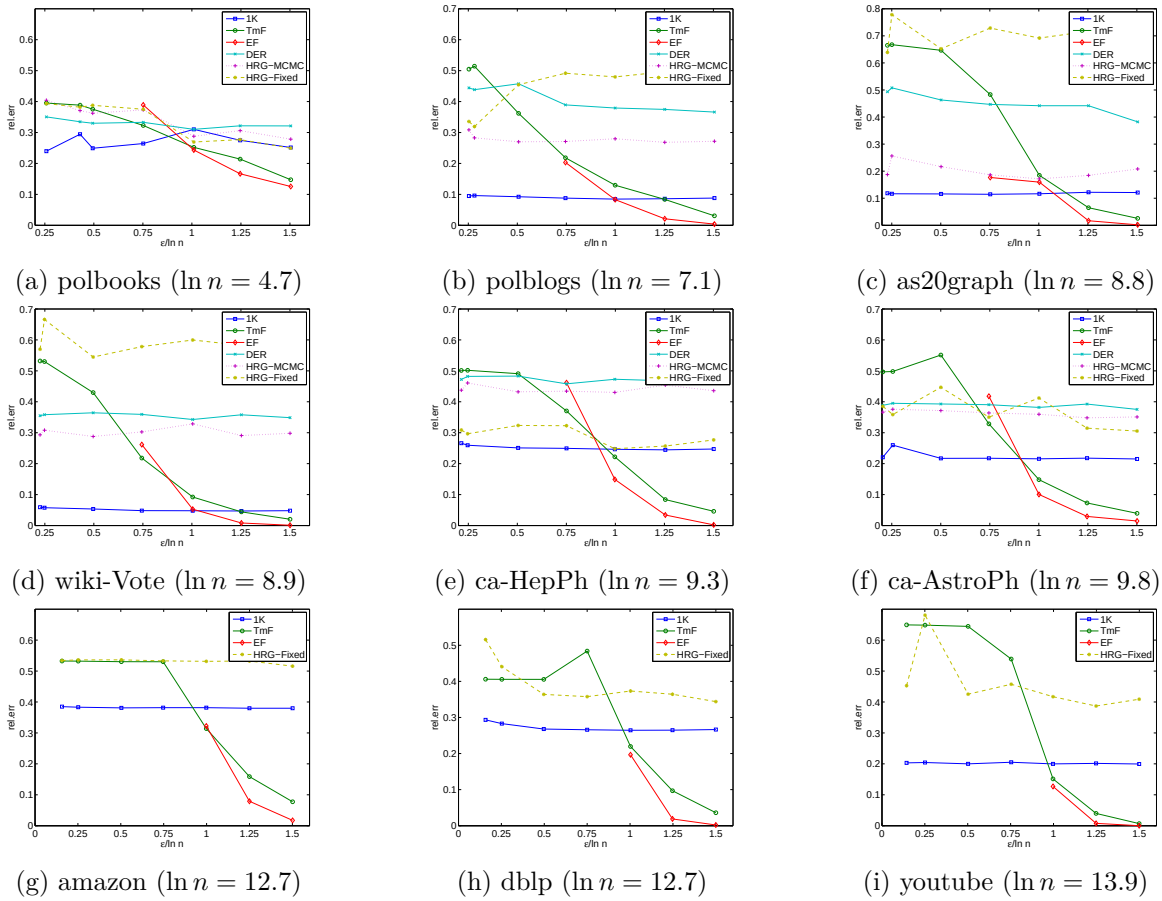
almost identical to that of the true graph while larger values of ϵ are redundant. Put differently, 1K-series benefits a lot from the five degree-based metrics (details below in Figures 4.13, 4.14 and 4.15). On the six small graphs, HRG-MCMC performs better than DER but the gap is small in *polbooks*, *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 4.13, 4.14 and 4.15. 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 4.4. As expected, TmF, EF and 1K-series run very fast,



(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 ($\ln n = 9.8$)

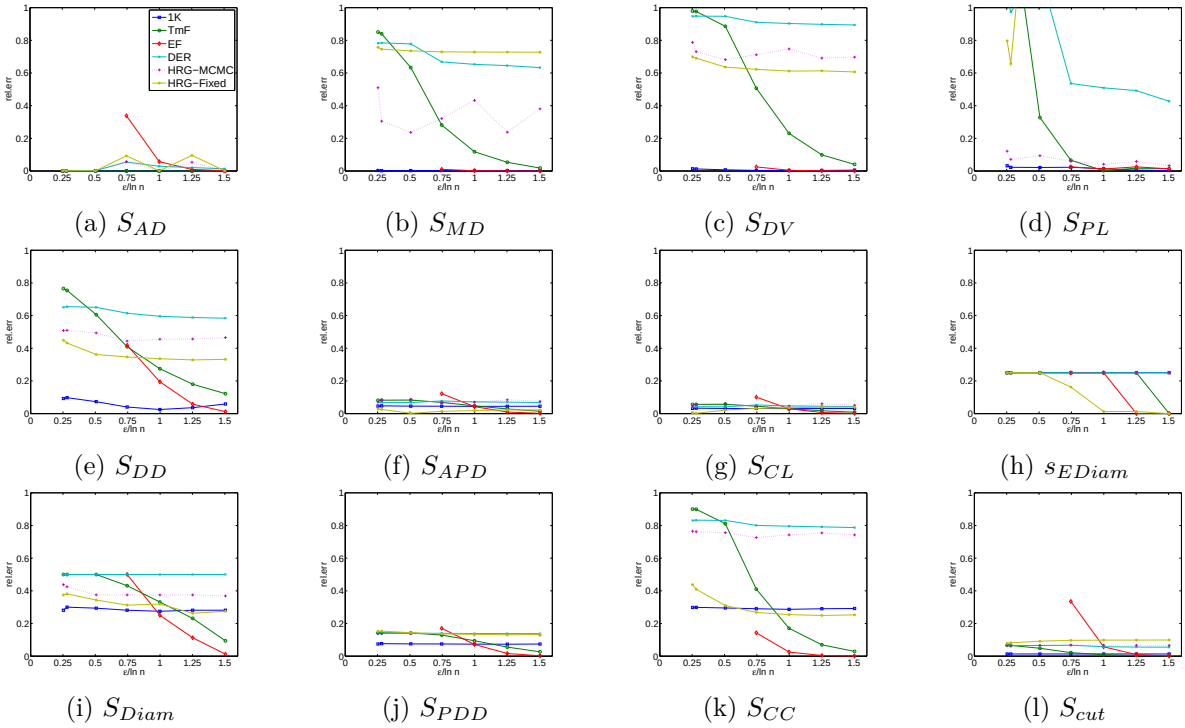
(g) amazon ($\ln n = 12.7$) (h) dblp ($\ln n = 12.7$) (i) youtube ($\ln n = 13.9$)

Figure 4.12: Comparative evaluation: the relative error is averaged on twelve utility metrics

linear in the number of edges. They run in several seconds on *youtube*, the largest graph considered in this chapter. 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 [105].

4.8 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

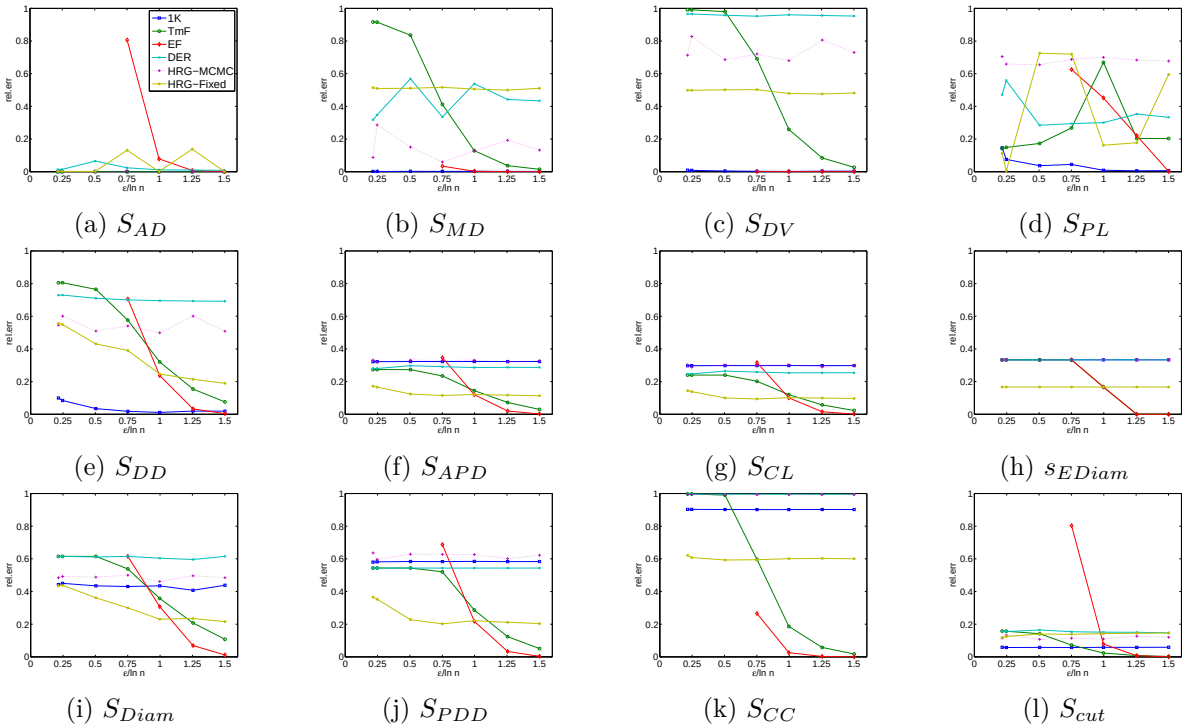


(a) S_{AD} (b) S_{MD} (c) S_{DV} (d) S_{PL}

(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 4.13: Relative errors of utility metrics on *polblogs*

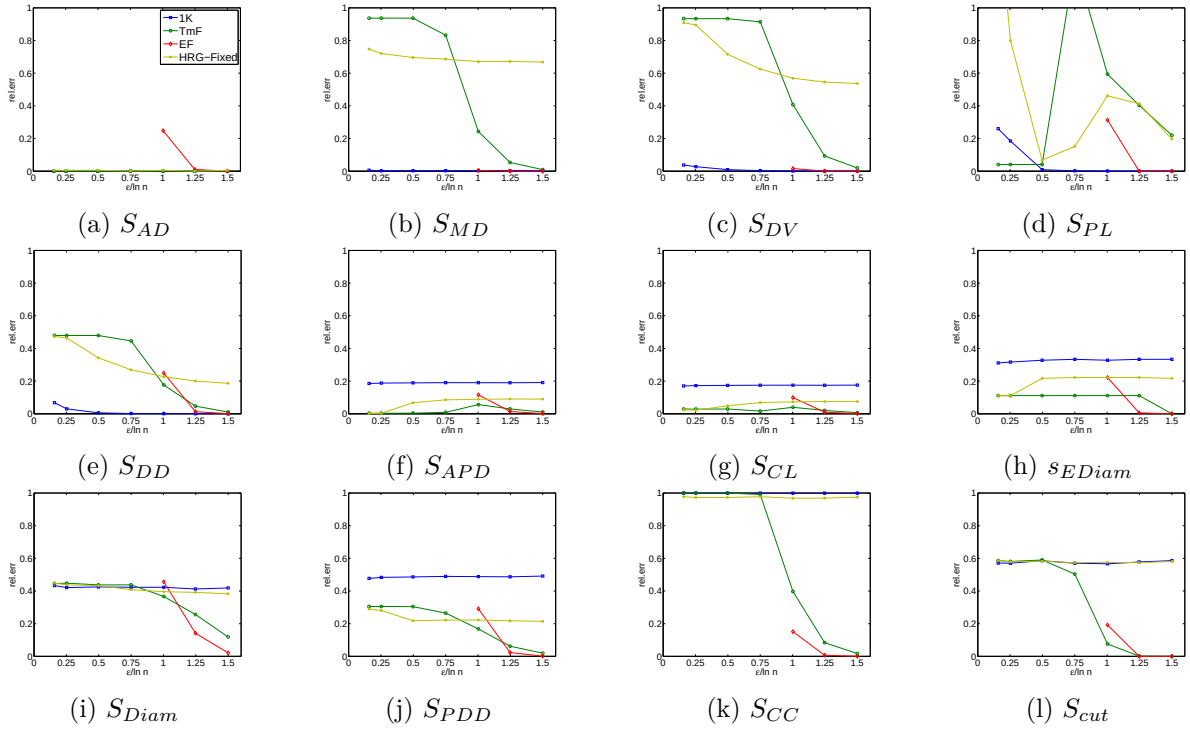


(a) S_{AD} (b) S_{MD} (c) S_{DV} (d) S_{PL}

(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 4.14: 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 4.15: Relative errors of utility metrics on *dblp*

to (1) find better consistent schemes and (2) examine summary structures for graphs other than HRG and 1K-series.

The next chapter presents another application of differential privacy. It tackles the problem of ϵ-DP community detection.

Table 4.4: 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

Chapter 5

Detecting Communities under Differential Privacy

Contents

5.1 Introduction

Graphs represent a rich class of data observed in daily life where entities are described by nodes and their connections are characterized by edges. Apart from microscopic (node level) and macroscopic (graph level) configurations, many complex networks display a *mesoscopic* structure, i.e. they appear as a combination of components fairly independent of each other. These components are called communities, modules or clusters and the problem of how to reveal them plays a significant role in understanding the organization and function of complex networks. Over the last decade, a great number of algorithms for community detection (CD) have been proposed to address the problem in a variety of settings, such as undirected/directed, unweighted/weighted networks and non-overlapping/overlapping communities (for a comprehensive survey, see [38]).

These approaches, however, are adopted in a non-private manner, i.e. a data collector (such as Facebook) knows all the contributing users and their relationships before running CD algorithms. The output of such a CD, in the simplest form, is a clustering of nodes. Even in this case, i.e. only a node clustering (not the whole graph) is revealed, contributing users' privacy may still be put at risk (cf. Section 2.3.2).

Community detection is closely related to the problems of clustering (e.g. k-Means [94]) and recommender systems [56] (see Section 5.2.2). A straightforward application of community detection is to predict missing links (or link recommendation), the focus of link prediction problem [57]. For example, given noisy output clusterings by private community detection algorithms, the analysts may predict future links by assigning higher probabilities to intra-cluster links and lower probabilities to inter-cluster links. The prediction accuracy in this case would be lower than the case of non-private high-quality clusterings but it may still be useful.

In this chapter, we address the problem of CD from the perspective of differential privacy [34]. 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 [63]. By differential privacy, we want to ensure the existence of connections between users to be hidden in the output clustering while keeping the low distortion of clusters compared to the ones generated by the corresponding non-private algorithms.

As far as we know, the problem is quite new and only mentioned in the recent work [69] where Mülle et al. use a sampling technique to perturb the input graph so that it satisfies differential privacy before running the conventional CD algorithms. This technique (we call it *EdgeFlip* afterwards) is classified as *input perturbation* in differential privacy literature (the other two categories are *algorithm perturbation* and *output perturbation*). Similarly, *TmF* approach [77] can apply to the true graph to get noisy output graphs as in the work of Mülle et al. Earlier, 1k-Series [100], *Density Explore Reconstruct* (DER) [21] and HRG-MCMC [105] are the best known methods for graph structure release under differential privacy. These methods can be followed by any exact CD algorithm to get a noisy clustering satisfying differential privacy. We choose Louvain method [9] as such a CD algorithm. However, as we will see in the experiments, the output clusterings by the aforementioned methods have very low modularity scores. This fact necessitates new methods for CD problem under differential privacy.

Our main contributions are the new schemes *LouvainDP* (input perturbation) and *ModDivisive* (algorithm perturbation) which perform much better than the state-of-the-art. LouvainDP is a high-pass filtering method that randomly groups nodes into supernodes of equal size to build a weighted supergraph. LouvainDP is guaranteed to run in linear time. ModDivisive is a top-down approach which privately divides the node set into the k-ary tree guided by the modularity score at each level. The main technique used in ModDivisive is the Markov Chain Monte Carlo (MCMC) to realize the exponential mechanism [62]. We show that ModDivisive's runtime is linear in the number of nodes, the height of the binary tree and the burn-in factor of MCMC. The linear complexity enables us to examine million-scale graphs in minutes. The experiments show the high modularity and low distortion of the output clusters by LouvainDP and ModDivisive.

Our contributions are summarized as follows:

- We analyze the major challenges of community detection under differential privacy. We explain why techniques borrowed from k-Means fail and how the difficulty of ϵ-DP recommender systems justifies a relaxation of ϵ.
- We design an input perturbation scheme LouvainDP that runs in linear time using the high-pass filtering technique from [26] and Louvain method [9].

- We propose an algorithm perturbation scheme ModDivisive as a divisive approach by using the modularity-based score function in the exponential mechanism. We prove that modularity has small global sensitivity and ModDivisive also runs in linear time.
- We conduct a thorough evaluation on real graphs of different sizes and show the outperformance of LouvainDP and ModDivisive over the state-of-the-art.

The chapter is organized as follows. Section 5.2 briefly introduces the popular Louvain method and the major challenges of ϵ-DP community detection. Section 5.3 focuses on the category of input perturbation in which we propose LouvainDP and review several recent input perturbation schemes. We describe ModDivisive in Section 5.4. We compare all the presented schemes on real graphs in Section 5.5. Finally, we present our conclusions and suggest future work in Section 5.6.

Table 5.1 summarizes the key notations used in this chapter.

Table 5.1: List of notations

Symbol	Definition
$G = (V, E_G)$	true graph with $n = \|V\|$ and $m = \|E_G\|$
$G' = (V, E_{G'})$	neighboring graph of G
$\tilde{G} = (V, E_{\tilde{G}})$	sample noisy output graph
$G_1 = (V_1, E_1)$	supergraph generated by LouvainDP
k	fan-out of the tree in ModDivisive
K	burn-in factor in MCMC-based algorithms
λ	common ratio to distribute the privacy budget
C	a clustering of nodes in G
$Q(C, G)$	modularity of the clustering C on graph G

5.2 Preliminaries

In this section, we review Louvain method and the major challenges of ϵ-DP community detection.

5.2.1 Louvain Method

Since its introduction in 2008, Louvain method [9] becomes one of the most cited methods for the community detection task. It optimizes the modularity by a bottom-up folding process. The algorithm is divided in passes each of which is composed of two phases that are repeated iteratively. Initially, each node is assigned to a different community. So, there will be as many communities as there are nodes in the first phase. Then, for each node i, the method considers the gain of modularity if we move i from its community to the community of a neighbor j (a *local change*). The node i is then placed in the community for which this gain is maximum and positive (if any), otherwise it stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first pass is then complete.

We demonstrate Louvain method in Fig.5.1 by a graph of 13 nodes and 20 edges. If each node forms its own singleton community, the modularity Q will be -0.0825. In the first pass of Louvain method, each node moves to the best community selected from its neighbors' communities. We get the partition $[\{0, 1, 2\}, \{3, 4\}, \{5, 6, 11, 12\}, \{7, 8, 9, 10\}]$ with modularity 0.46375. The

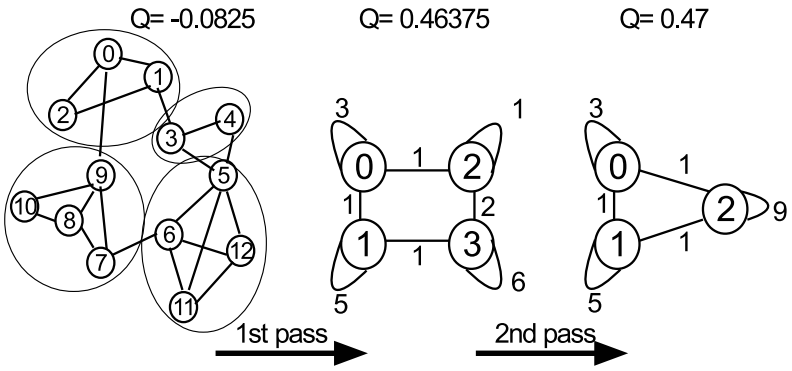


Figure 5.1: Louvain method

second phase of first pass builds a weighted graph corresponding to the partition by aggregating communities. The second pass repeats the folding process on this weighted graphs to reach the final partition $[\{0,1,2\},\{3,4,5,6,11,12\},\{7,8,9,10\}]$ with modularity 0.47.

This simple algorithm has several advantages as stated in [9]. First, its steps are intuitive and easy to implement, and the outcome is unsupervised. Second, the algorithm is extremely fast, i.e. computer simulations on large modular networks suggest that its complexity is linear on typical and sparse data. This is due to the fact that the possible gains in modularity are easy to compute and the number of communities decreases drastically after just a few passes so that most of the running time is concentrated on the first iterations. Third, the multi-level nature of the method produces a hierarchical structure of communities which allows multi-resolution analysis, i.e the user can zoom in the graph to observe its structure with the desired resolution. In addition, Louvain method is runnable on weighted graphs. This fact supports naturally our scheme LouvainDP as described in the next section.

5.2.2 Challenges of Community Detection under Differential Privacy

In this section, we explain why community detection under differential privacy is challenging. We show how techniques borrowed from related problems fail. We also advocate the choice of ϵ as a function of the graph size n.

The problem of differentially private community detection is closely related to ϵ-DP k-Means clustering and recommender systems. The ϵ-DP k-Means is thoroughly discussed in [94]. However, techniques from ϵ-DP k-Means are not suitable to ϵ-DP community detection. First, items in k-Means are in low-dimensional spaces and the number of clusters k is usually small. This contrast to the case of community detection where nodes lie in a n-dimensional space and the number of communities varies from tens to tens of thousands, not to say the communities may overlap or be nested (multi-scale). Second, items in ϵ-DP k-Means are normalized to $[-1,1]^d$ while the same preprocessing seems invalid in ϵ-DP community detection. Moreover, the output of k-Means usually consists of equal-sized balls while this is not true for communities in graphs. Considering the graph as a high-dimensional dataset, we tried the private projection technique in [50] which is followed by spectral clustering, but the modularity scores of the output are not better than random clustering.

Recent papers on ϵ-DP recommender systems [6, 43] show that privately learning the clustering of items from user ratings is hard unless we relax the value of ϵ up to $\log n$. Banerjee

et al. [6] model differentially private mechanisms as noisy channels and bound the mutual information between the generative sources and the privatized sketches. They show that in the information-rich regime (each user rates $O(n)$ items), their *Pairwise-Preference* succeeds if the number of users is $\Omega(n \log n/\epsilon)$. Compared to ϵ-DP community detection where the number of users is n, we should have $\epsilon = \Omega(\log n)$. Similarly, in D2P scheme, Guerraoui et al. [43] draw a formula for ϵ as

$$\epsilon_{D2P}^{(p,0,\lambda)} = \ln(1 + \frac{(1-p).\mathcal{N}_E}{p.|\mathcal{G}_\lambda|}) \tag{5.1}$$

where λ is the distance used to conceal the user profiles ($\lambda = 0$ reduces to the classic notion of differential privacy). $\mathcal{N}_E$ is the number of items which is exactly n in community detection. $|\mathcal{G}_\lambda|$ is the minimum size of user profiles at distance λ over all users ($|\mathcal{G}_\lambda| = o(\mathcal{N}_E)$ except at unreasonably large λ). Clearly, at $p = 0.5$ (as used in [43]), we have $\epsilon_{D2P}^{(0.5,0,\lambda)} \approx \ln n$. The sampling technique in D2P is very similar to EdgeFlip [69] which is shown ineffective in ϵ-DP community detection for $\epsilon \in (0, 0.5 \ln n)$ (Section 5.5). Note that ϵ-DP community detection is unique in the sense that the set of items and the set of users are the same. Graphs for community detection are more general than bipartite graphs in recommender systems. In addition, modularity Q (c.f. Formula 2.1) is *non-monotone*, i.e. for two disjoint sets of nodes A and B, $Q(A \cup B)$ may be larger, smaller than or equal to $Q(A) + Q(B)$.

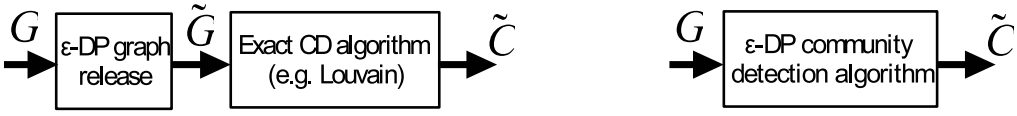
To further emphasize the difficulty of ϵ-DP community detection, we found that IDC scheme [44] using *Sparse Vector Technique* [35, Section 3.6] is hardly feasible. As shown in Algorithm 1 of [44], to publish a noisy graph that can approximately answer all cut queries with bounded error $m^{0.25}n/\epsilon^{0.5}$, IDC must have $B(\alpha)$ "yes" queries among all k queries. $B(\alpha)$ may be as low as $\sqrt{m}$ but $k = 2^{2n}$. In the average case, IDC incurs exponential time to complete.

The typical epsilon in the literature is 1.0 or less. However, this value is only applicable to graph metrics of low sensitivity $O(1)$ such as the number of edges, the degree sequence. The global sensitivity of other metrics like the diameter, the number of triangles, 2K-series etc. is $O(n)$, calling for smooth sensitivity analysis (e.g. [79]). For counting queries, Laplace/Geometric mechanisms are straightforward on real/integral (*metric*) spaces. However, *direct* noise adding mechanisms on the space $\mathcal{P}$ of all ways to partition the nodeset V are non-trivial because $|\mathcal{P}| \approx n^n$ and $\mathcal{P}$ is non-metric. Compared to ϵ-DP graph release (TmF and EdgeFlip schemes in Chapter 4) and ϵ-DP recommender systems discussed above which use $\epsilon = \ln n$ for the super-exponential spaces of size $O(2^{n(n-1)/2})$, ϵ-DP community detection clearly needs lower privacy budget.

To conclude, ϵ-DP community detection is challenging and requires new techniques. In this chapter, we evaluate the schemes for ϵ up to $0.5 \ln n$. At $\epsilon = 0.5 \ln n$, the multiplicative ratio (c.f. Definition 4.1) is $e^\epsilon = e^{0.5 \ln n} = \sqrt{n}$. We believe it is a reasonable threshold for privacy protection compared to $\epsilon = \ln n$ (i.e. $e^\epsilon = n$) in ϵ-DP graph release and ϵ-DP recommender systems.

5.3 Input Perturbation

In this section, we propose the linear scheme LouvainDP that uses a filtering technique to build a noisy weighted supergraph and calls the exact Louvain method subsequently. Then we discuss several recent ϵ-DP schemes that can be classified as input perturbation. Fig.5.2a sketches the basic steps of the input perturbation paradigm.



(a) Input perturbation

(b) Algorithm perturbation

Figure 5.2: Two categories of ϵ-DP community detection

5.3.1 LouvainDP: Louvain Method on Noisy Supergraphs

The basic idea of LouvainDP is to create a noisy weighted supergraph G_1 from G by grouping nodes into supernodes of equal size k. Then we apply the filtering technique of Cormode et al. [26] to ensure that there are only $O(m)$ noisy weighted edges in G_1. Finally, we run the exact community detection on G_1.

The high-pass filtering technique [26] was mentioned in 4.3.1. In our LouvainDP, the supergraph G_1 is an instance of sparse data with the domain size $m_0 = \frac{n_1(n_1+1)}{2}$ where n_1 is the number of supernodes and m_1 is the number of non-zero entries corresponding to non-zero superedges. We use the one-sided filtering [26] to efficiently compute G_1 with $O(m)$ edges in linear time.

Algorithm

LouvainDP can run with either geometric or Laplace noise. We describe the version of geometric noise in Algorithm 10.

Given the group size k, LouvainDP starts with a supergraph G_1 having $\lfloor \frac{|V|}{k} \rfloor$ nodes by randomly permuting the nodeset V and grouping every k consecutive nodes into a supernode (lines 1-4). The permutation prevents the possible bias of node ordering in G. The set of superedges E_1 is easily computed from G. Note that $m_1 = |E_1| \leq m$ due to the fact that each edge of G appears in one and only one superedge. The domain size is $m_0 = \frac{n_1(n_1+1)}{2}$ (i.e. we consider all selfloops in G_1). The noisy number of non-zero superedges is $m_1 = |E_1| + Lap(1/\epsilon_2)$. Then by *one-sided* filtering [26], we estimate the threshold θ (line 7) and the number of passing zero superedges s (line 8). For each non-zero superedge, we add a geometric noise and add the superedge to G_1 if the noisy value is not smaller than θ. For s zero superedges $e_1(i,j) \notin E_1$, we draw an integral weight w from the distribution $Pr[X \leq x] = 1 - \alpha^{x-\theta+1}$ and add $e_1(i,j)$ with weight w to G_1 if $w > 0$.

Complexity

LouvainDP runs in $O(m)$ because the loops to compute superedges (Line 5) and to add geometric noises (lines 9-12) cost $O(m)$. We have $s = (m_0 - m_1)\frac{\alpha^\theta}{1+\alpha} \leq \frac{m_0-m_1}{1+\alpha}\frac{(1+\alpha)m_1}{m_0-m_1} = m_1$ (see Line 7). So the processing of s zero-superedges costs $O(m)$. Moreover, Louvain method (line 17) is empirically linear in m_1 [9]. Informally, we can state that LouvainDP's runtime is $O(m)$ on any graph where Louvain takes $O(m)$ to run.

Privacy Analysis

In LouvainDP, we use a small privacy budget $\epsilon_2 = 0.1$ to compute the noisy number of non-zero superedges m_1. The remaining privacy budget ϵ_1 is used for the geometric mechanism $Geom(\alpha)$.

Algorithm 10 LouvainDP(G, s)

Input: undirected graph G, group size k, privacy budget ϵ
Output: noisy partition $\tilde{C}$

1: $G_1 \leftarrow \varnothing$, $n_1 = \lfloor \frac{|V|}{k} \rfloor - 1$, $V_1 \leftarrow \{0, 1, .., n_1\}$
2: $\epsilon_2 = 0.1$, $\epsilon_1 = \epsilon - \epsilon_2$, $\alpha = \exp(-\epsilon_1)$
3: get a random permutation V_p of V
4: compute the mapping $M : V_p \rightarrow V_1$
5: compute superedges of G_1: $E_1 = \{e_1(i,j)\}$ where $i, j \in V_1$
6: $m_1 = |E_1| + Lap(1/\epsilon_2)$, $m_0 = \frac{n_1(n_1+1)}{2}$
7: $\theta = \lceil \log_\alpha \frac{(1+\alpha)m_1}{m_0 - m_1} \rceil$
8: $s = (m_0 - m_1)\frac{\alpha^\theta}{1+\alpha}$
9: **for** $e_1(i,j)$ in E_1 **do**
10: $\quad e_1(i,j) = e_1(i,j) + Geom(\alpha)$
11: $\quad$ **if** $e_1(i,j) \geq \theta$ **then**
12: $\quad\quad$ add $e_1(i,j)$ to G_1
13: **for** s edges $e_0(i,j)$ sampled uniformly at random such that $e_0(i,j) \notin E_1$ **do**
14: $\quad$ draw w from the distribution $Pr[X \leq x] = 1 - \alpha^{x-\theta+1}$
15: $\quad$ **if** $w > 0$ **then**
16: $\quad\quad$ add edge $e_0(i,j)$ with weight w to G_1
17: run Louvain method on G_1 to get $\tilde{C}_1$
18: compute $\tilde{C}$ from $\tilde{C}_1$ using the mapping M
19: **return** $\tilde{C}$

Note that getting a random permutation V_p (line 3) costs no privacy budget. The number of nodes n is public and given the group size k, the number of supernodes n_1 is also public. The high-pass filtering technique (Lines 6-16) inherits the privacy guarantee by [26]. By setting $\epsilon_1 = \epsilon - \epsilon_2$, LouvainDP satisfies ϵ-differential privacy (see the sequential composition (Theorem 4.3)).

5.3.2 Other Input Perturbation Schemes

1K-series [100], DER [21], TmF [77] and EdgeFlip [69] are the most recent differentially private schemes for graph release that can be classified as input perturbation. While 1K-series and TmF run in linear time, DER and EdgeFlip incur a quadratic complexity. DER and EdgeFlip are therefore tested only on two medium-sized graphs in Section 5.5.

The expected number of edges by EdgeFlip is $|E_{\tilde{G}}| = (1-s)m + \frac{n(n-1)}{4}s$ (see [69]) where $s = \frac{2}{e^\epsilon + 1}$ is the flipping probability. Substitute s into $|E_{\tilde{G}}|$, we get $|E_{\tilde{G}}| = m + (\frac{n(n-1)}{4} - m)\frac{2}{e^\epsilon+1}$. The number of edges in the noisy graph $\tilde{G}$ generated by EdgeFlip increases exponentially as ϵ decreases. To ensure the linear complexity for million-scale graphs, we propose a simple extension of EdgeFlip, called *EdgeFlipShrink* (Algorithm 11).

Instead of outputting $\tilde{G}$, EdgeFlipShrink computes $\hat{G}$ that has the expected number of edges m by shrinking $E_{\tilde{G}}$. First, the algorithm computes the private number of edges $\tilde{m}$ using a small budget ϵ_2 (Lines 2-3). The new flipping probability $\tilde{s}$ is updated (Line 5). The noisy expected number of edges in the original EdgeFlip is shown in Line 6. We obtain the shrinking factor $p = \frac{\tilde{m}}{m_0}$ (Line 7). Using p, every 1-edge is sampled with probability $\frac{1-\tilde{s}}{2}p$ instead of $\frac{1-s}{2}$ as in [69]. The remaining 0-edges are randomly picked from E_G as long as they do not exist in $\hat{G}$

(Lines 14-19).

The expected edges of $\hat{G}$ is $E[\hat{G}] = E[\tilde{m}] = m$ and the running time of EdgeFlipShrink is $O(m)$.

Algorithm 11 EdgeFlipShrink(G, s)

Input: undirected graph G, flipping probability s
Output: anonymized graph $\hat{G}$

1: $\hat{G} \leftarrow \varnothing$
2: $\epsilon_2 = 0.1$
3: $\tilde{m} = m + Lap(1/\epsilon_2)$
4: $\epsilon = \ln(\frac{2}{s} - 1) - \epsilon_2$
5: $\tilde{s} = \frac{2}{e^{\epsilon}+1}$
6: $m_0 = (1 - \tilde{s})\tilde{m} + \frac{n(n-1)}{4}\tilde{s}$
7: $p = \frac{\tilde{m}}{m_0}$
8: // process 1-edges
9: $n_1 = 0$
10: **for** edge $(i, j) \in E_G$ **do**
11: add edge (i, j) to $\hat{G}$ with prob. $\frac{1-\tilde{s}}{2}p$
12: $n_1 + +$
13: // process 0-edges
14: $n_0 = \tilde{m} - n_1$
15: **while** $n_0 > 0$ **do**
16: random pick an edge $(i, j) \notin E_G$
17: **if** $\hat{G}$ does not contain (i, j) **then**
18: add edge (i, j) to $\hat{G}$
19: n_0- -
20: **return** $\hat{G}'$

5.4 Algorithm Perturbation

The schemes in the algorithm perturbation category privately sample a node clustering from the input graph without generating noisy sample graphs as in the input perturbation. This can be done via the exponential mechanism. We introduce our main scheme *ModDivisive* in Section 5.4.1 and explain how HRG-MCMC and its variants are also instances of algorithm perturbation schemes (Section 5.4.2). Fig.5.2b sketches the basic steps of the algorithm perturbation paradigm.

5.4.1 ModDivisive: Top-down Exploration of Cohesive Groups

Overview

In contrast with the agglomerative approaches (e.g. Louvain method) in which small communities are iteratively merged if doing so increases the modularity, our ModDivisive scheme is a divisive algorithm in which communities at each level are iteratively split into smaller ones. Our goal is to heuristically detect cohesive groups of nodes in a private manner. There are several technical challenges in this process. The first one is to efficiently find a good split of nodes that induces a high modularity and satisfies ϵ-DP at the same time. The second one is how to merge

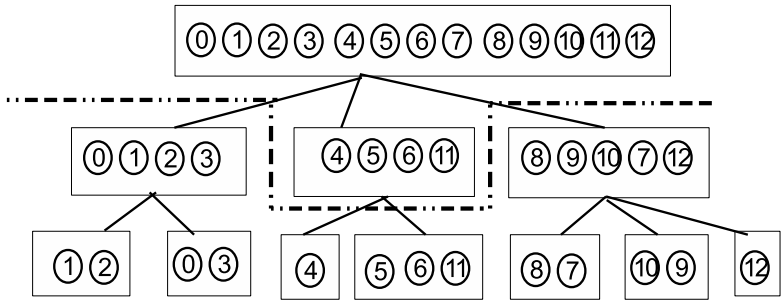


Figure 5.3: example of ModDivisive with $k = 3$. A cut C is shown by the dot-dashed line

the small groups to larger ones. We cope with the first challenge by realizing an exponential mechanism via MCMC (Markov Chain Monte-Carlo) sampling with the modularity as the score function (see Theorem 4.2), i.e. $u(G, C) = Q(C, G)$. The second challenge is solved by dynamic programming. We design ModDivisive as a *k-ary* tree (Fig.5.3), i.e. each internal node has no more than k child nodes. The root node (level 0) contains all nodes in V and assigns arbitrarily each node to one of the k groups. Then we run the MCMC over the space of all partitions of V into no more than k nonempty subsets. The resulting subsets are initialized as the child nodes (level 1) of the root. The process is repeated for each child node at level 1 and stops at level $maxL$. Fig.5.3 illustrates the idea with $k = 3$ for the graph in Fig.5.1.

Algorithm

Algorithm 12 sketches the main steps in our scheme ModDivisive. It comprises two phases: differentially private sampling a k-ary tree of depth $maxL$ which uses the privacy budget ϵ_1 and finding the best cut across the tree to get a good clustering of nodes which consumes a budget $maxL.\epsilon_m$.

The first phase (lines 1-14) begins with the creation of eA, the array of privacy budgets allocated to levels of the tree. We use the parameter $\lambda \geq 1$ as the common ratio to form a geometric sequence. The rationale behind the common ratio is to give higher priority to the levels near the root which have larger node sets. By sequential composition (Theorem 4.3), we must have $\sum_i eA[i] = \epsilon_1$. All internal nodes at level i do the MCMC sampling on disjoint subsets of nodes, so the parallel composition holds. Subsection 5.4.1 analyzes the privacy of ModDivisive in more detail. We use a queue to do a level-by-level exploration. Each dequeued node r's level will be checked. If its level is not larger than $maxL$, we will run *ModMCMC* (Algorithm 13) on it (line 9) to get a partition $r.part$ of its nodeset $r.S$ (Fig.5.3). Each subset in $r.part$ forms a child node and is pushed to the queue. The second phase (line 15) calls Algorithm 14 to find a highly modular partition across the tree.

Differentially Private Nodeset Partitioning Let $\mathcal{P}$ be the space of all ways P to partition a nodeset A to no more than k disjoint subsets, the direct application of exponential mechanism needs the enumeration of $\mathcal{P}$. The probability of a partition P being sampled is

$$\frac{\exp(\frac{\epsilon_p}{2\Delta Q}Q(P, G))}{\sum_{P' \in \mathcal{P}} \exp(\frac{\epsilon_p}{2\Delta Q}Q(P', G))} \tag{5.2}$$

However, $|\mathcal{P}| = \sum_{i=1}^{k} S(|A|, i)$ where $S(|A|, i)$ is the Stirling number of the second kind [42], $S(n, k) \approx \frac{k^n}{k!}$. This sum is exponential in $|A|$, so enumerating $\mathcal{P}$ is computationally infeasible.

Algorithm 12 ModDivisive

Input: graph G, group size k, privacy budget ϵ, max level $maxL$, ratio λ, BestCut privacy at each level ϵ_m
Output: noisy partition $\tilde{C}$

1: compute the array $eA[0..maxL-1]$ s.t. $\sum_i eA[i] = \epsilon_1$, $eA[i] = eA[i+1] * \lambda$ where $\epsilon_1 = \epsilon - maxL.\epsilon_m$
2: initialize the root node with nodeset V
3: $root = \text{NodeSet}(G, V, k)$
4: $root.level = 0$
5: queue $Q \leftarrow root$
6: **while** Q is not empty **do**
7: $\quad r \leftarrow Q.dequeue()$
8: $\quad$ **if** $r.level < maxL$ **then**
9: $\quad\quad r.part = \text{ModMCMC}(G, r.S, k, eA[r.level])$
10: $\quad\quad$ **for** subset S_i in $r.part$ **do**
11: $\quad\quad\quad P_i = \text{NodeSet}(G, S_i, k)$
12: $\quad\quad\quad P_i.level = r.level + 1$
13: $\quad\quad\quad r.child_i \leftarrow P_i$
14: $\quad\quad\quad Q.enqueue(P_i)$
15: $\tilde{C} \leftarrow \text{BestCut}(root, \epsilon_m)$
16: **return** $\tilde{C}$

Fortunately, MCMC can help us simulate the exponential mechanism by a sequence of local transitions in $\mathcal{P}$.

The space $\mathcal{P}$ is connected. It is straightforward to verify that the transitions performed in line 3 of ModMCMC are *reversible* and *ergodic* (i.e. any pair of nodeset partitions can be connected by a sequence of such transitions). Hence, ModMCMC has a unique stationary distribution in equilibrium. By empirical evaluation, we observe that ModMCMC converges after $K|r.S|$ steps for $K = 50$ (see Section 5.5.3).

Each node r in the tree is of type *NodeSet*. This structure consists of an array $r.part$ where $r.part[u] \in \{0..k-1\}$ is the group id of u. To make sure that ModMCMC runs in linear time, we must have a constant time computation of modularity $Q(P)$ (line 4 of Algorithm 13). This can be done with two helper arrays: the number of intra-edges $r.lc[0..k-1]$ and the total degree of nodes $r.dc[0..k-1]$ in each group. The modularity Q is computed in $O(k)$ (Formula 2.1) using $r.lc, r.dc$. When moving node u from group i to group j, $r.lc$ and $r.dc$ are updated accordingly by checking the neighbors of u in G. The average degree is a constant, so the complexity of ModMCMC is linear in the number of MCMC steps.

Finding the Best Cut Given the output k-ary tree R with the root node $root$, our next step is to find the best cut across the tree. A cut C is a set of nodes in R that cover all nodes in V. As an example, a cut C in Fig.5.3 returns the clustering [{0,1,2,3}, {4}, {5,6,11}, {8,9,10,7,12}]. Any cut has a modularity score. Our goal is to find the best cut, i.e. the cut with highest modularity, in a private manner.

We solve this problem by a dynamic programming technique. Remember that modularity is an *additive* quantity (c.f. Formula 2.1). By denoting $opt(r)$ as the optimal modularity for the subtree rooted at node r, the optimal value is $opt(root)$. The recurrence relation is

Algorithm 13 ModMCMC

Input: graph G, nodeset $r.S$, group size k, privacy budget ϵ_p
Output: sampled partition $r.part$
1: initialize $r.part$ with a random partition P_0 of k groups
2: **for** each step i in the Markov chain **do**
3: pick a neighboring partition P' of P_{i-1} by randomly selecting node $u \in r.S$ and moving u to another group.
4: accept the transition and set $P_i = P'$ with probability $min(1, \frac{\exp(\frac{\epsilon_p}{2\Delta Q} Q(P', G))}{\exp(\frac{\epsilon_p}{2\Delta Q} Q(P_{i-1}, G))})$
5: // until equilibrium is reached
6: **return** a sampled partition $r.part = P_i$

straightforward

$$opt(r) = \max\{Q(r), \sum_{t \in r.children} opt(t)\}$$

Algorithm 14 realizes this idea in three steps. The first step (lines 1-6) uses a queue to fill a stack S. The stack ensures any internal node to be considered after its child nodes. The second step (lines 7-17) solves the recurrence relation. Because all modularity values are sensitive, we add Laplace noise Laplace($\Delta Q/\epsilon_m$). The global sensitivity $\Delta Q = 3/m$ (see Theorem 5.2), so we need only a small privacy budget for each level ($\epsilon_m = 0.01$ is enough in our experiments). The noisy modularity mod_n is used to decide whether the optimal modularity at node r is by itself or by the sum over its children. The final step (lines 18-25) backtracks the best cut from the root node.

Complexity

ModDivisive creates a k-ary tree of height $maxL$. At each node r of the tree other than the leaf nodes, ModMCMC is run once. The run time of ModMCMC is $O(K * |r.S|)$ thanks to the constant time for updating the modularity (line 4 of ModMCMC). Because the union of nodesets at one level is V, the total runtime is $O(K*|V|*maxL)$. BestCut only incurs a sublinear runtime because the size of tree is always much smaller than $|V|$. The following theorem states this result

Theorem 5.1. *The time complexity of ModDivisive is linear in the number of nodes n, the maximum level $maxL$ and the burn-in factor K.* □

Privacy Analysis

We show that ModMCMC satisfies differential privacy. The goal of MCMC is to draw a random sample from the desired distribution. Similarly, exponential mechanism is also a method to sample an output $x \in X$ from the target distribution with probability proportional to $\exp(\frac{\epsilon u(x)}{2\Delta u})$ where $u(x)$ is the score function (x with higher score has bigger chance to be sampled) and Δu is its sensitivity. The idea of using MCMC to realize exponential mechanism is first proposed in [90] and applied to ϵ-DP graph release in [105].

In our ModMCMC, the modularity $Q(P, G)$ is used directly as the score function. We need to quantify the global sensitivity of Q. From Section 4.2.1, we have the following definition

Definition 5.1. *(Global Sensitivity ΔQ)*

$$\Delta Q = \max_{P,G,G'} |Q(P, G) - Q(P, G')| \tag{5.3}$$

Algorithm 14 BestCut

Input: undirected graph G, root node $root$, privacy budget at each level ϵ_m
Output: best cut C

stack $S \leftarrow \varnothing$, queue $Q \leftarrow$ root
while Q is not empty **do**
    $r \leftarrow Q.dequeue()$
    $S.push(r)$
    for child node r_i in $r.children$ **do**
        $Q.enqueue(r_i)$
dictionary $sol \leftarrow \varnothing$
while S is not empty **do**
    $r \leftarrow S.pop()$, $r.mod_n = r.mod + \text{Laplace}(\Delta Q/\epsilon_m)$
    if r is a leaf node **then**
        $sol.put(r.id, (\text{val} = r.mod_n, \text{self} = True))$
    else
        $s_m = \sum_{r_i \in r.children} sol[r_i.id].mod_n$
        if $r.mod_n < s_m$ **then**
            $sol.put(r.id, (\text{val} = s_m, \text{self} = False))$
        else
            $sol.put(r.id, (\text{val} = r.mod_n, \text{self} = True))$
list $C \leftarrow \varnothing$, queue $Q \leftarrow$ root
while Q is not empty **do**
    $r \leftarrow Q.dequeue()$
    if $sol[r.id].\text{self} == True$ **then**
        $C = C \cup \{r\}$
    else
        for child node r_i in $r.children$ **do**
            $Q.enqueue(r_i)$
return C

We prove that $\Delta Q = O(1/m)$ in the following theorem

Theorem 5.2. *The global sensitivity of modularity, ΔQ, is smaller than $\frac{3}{m}$* □

Proof. Given the graph G and a partition P of a nodeset $V_p \subseteq V$ (for any node of the k-ary tree other than the root node, its nodeset V_p is a strict subset of V), the neighboring graph G' has $E_{G'} = E_G \cup e$. We have two cases

Case 1. The new edge e is an intra-edge within the community s. The modularity $Q(P, G)$ is $\sum_c^k (\frac{l_c}{m} - \frac{d_c^2}{4m^2})$. The modularity $Q(P, G')$ is $\sum_{c \neq s}^k (\frac{l_c}{m+1} - \frac{d_c^2}{4(m+1)^2}) + (\frac{l_s+1}{m+1} - \frac{(d_s+2)^2}{4(m+1)^2})$.

The difference $d_1 = Q(P, G') - Q(P, G) = \frac{1}{m+1} - \frac{1}{m(m+1)} \sum_c^k l_c + \frac{2m+1}{4m^2(m+1)^2} \sum_c^k d_c^2 - \frac{d_s+1}{(m+1)^2}$

Because ΔQ is the absolute value of d_1, we consider the most positive and the most negative values of d_1. Remember that $\sum_c^k d_c \leq 2m$, so the positive bound $d_1 < \frac{1}{m+1} + \frac{(2m+1)4m^2}{4m^2(m+1)^2} < \frac{3}{m+1}$. For the negative bound, we use the constraints $\sum_c^k l_c \leq m$ and $d_s \leq 2m$, so $d_1 > \frac{1}{m+1} - \frac{m}{m(m+1)} - \frac{2m+1}{(m+1)^2} > \frac{-2}{m+1}$. As a result, $\Delta Q = |d_1| < \frac{3}{m+1} < \frac{3}{m}$.

Case 2. The new edge e is an inter-edge between the communities s and t. Similarly, we have $Q(P, G) = \sum_c^k (\frac{l_c}{m} - \frac{d_c^2}{4m^2})$ while $Q(P, G') = \sum_{c \neq s,t}^k (\frac{l_c}{m+1} - \frac{d_c^2}{4(m+1)^2}) + (\frac{l_s}{m+1} - \frac{(d_s+1)^2}{4(m+1)^2}) + (\frac{l_t}{m+1} - \frac{(d_t+1)^2}{4(m+1)^2})$.

The difference $d_2 = Q(P, G') - Q(P, G) = -\frac{1}{m(m+1)} \sum_c^k l_c + \frac{2m+1}{4m^2(m+1)^2} \sum_c^k d_c^2 - \frac{2d_s+2d_t+2}{4(m+1)^2}$. Again, we consider the most positive and the most negative values of d_2, using the constraint $\sum_c^k d_c \leq 2m$, the positive bound $d_2 < \frac{(2m+1)4m^2}{4m^2(m+1)^2} < \frac{2}{m+1}$. For the negative bound, we use the constraints $\sum_c^k l_c \leq m$ and $d_s + d_t \leq 2m$, so $d_2 > -\frac{m}{m(m+1)} - \frac{4m+2}{4(m+1)^2} > \frac{-2}{m+1}$. As a result, $\Delta Q = |d_2| < \frac{2}{m+1} < \frac{2}{m}$.

To recap, in both cases $\Delta Q < \frac{3}{m}$. □

5.4.2 HRG-MCMC and Variants

We note that HRG-based schemes in Section 4.5 are also instances of ϵ-DP community detection. Omitting the step of outputting noisy sample graphs, noisy clusterings can be computed from noisy HRG trees. Given a noisy HRG tree, we compute the modularity for all internal nodes and apply the BestCut algorithm (Algorithm 14) to find the noisy clustering with the highest modularity. HRG-MCMC [105] will be a competitor on small and medium-sized graphs while HRG-FixedTree will apply to all graphs considered in the evaluation.

5.5 Experiments and Results

In this section, our evaluation aims to compare the performance of the competitors by clustering quality and efficiency. The clustering quality is measured by the modularity Q, the average F_1-score and the Normalized Mutual Information (NMI) in which the modularity is the most important metric as we aim at highly modular clusterings. The efficiency is measured by the running time. All algorithms are implemented in Java and run on a desktop PC with *Intel*® Core i7-4770@ 3.4Ghz, 16GB memory.

Two medium-sized and three large real graphs are used in our experiments [13]. `as20graph` is the graph of routers comprising the internet. `ca-AstroPh` and `dblp` are co-authorship networks

[13] http://snap.stanford.edu/data/index.html

where two authors are connected if they publish at least one paper together. `amazon` is a product co-purchasing network where the graph contains an undirected edge from i to j if a product i is frequently co-purchased with product j. `youtube` is a video-sharing web site that includes a social network. Table 5.2 shows the characteristics of the graphs. The columns Com(munities) and Mod(ularity) are the output of Louvain method. The number of samples in each test case is 20.

Table 5.2: Characteristics of the test graphs

	Nodes	Edges	Com	Mod
as20graph	6,474	12,572	30	0.623
ca-AstroPh	17,903	196,972	37	0.624
amazon	334,863	925,872	257	0.926
dblp	317,080	1,049,866	375	0.818
youtube	1,134,890	2,987,624	13,485	0.710

The schemes are abbreviated as 1K-series (1K), EdgeFlip (EF), Top-m-Filter (TmF), DER, LouvainDP (LDP), ModDivisive (MD), HRG-MCMC and HRG-Fixed.

5.5.1 Quality Metrics

Apart from modularity Q, we use two more metrics for quality evaluation. The first metric is $\bar{F_1}$, the *average* F_1*-score* following the benchmarks in [106]. The F_1 score of a set A with respect to a set B is defined as the harmonic mean H of the precision and the recall of A against B. We define $prec(A,B) = \frac{|A \cap B|}{|A|}$, $recall(A,B) = \frac{|A \cap B|}{|B|}$

$$F_1(A,B) = \frac{2.prec(A,B).recall(A,B)}{prec(A,B) + recall(A,B)}$$

Then the average F_1 score of two sets of communities C and C' is defined as

$$F_1(A,C) = \max_i F_1(A,c_i), \quad c_i \in C = \{c_1,..,c_n\}$$

$$\bar{F_1}(C,C') = \frac{1}{2|C|}\sum_{c_i \in C} F_1(c_i,C') + \frac{1}{2|C'|}\sum_{c_i \in C'} F_1(c_i,C)$$

The second metric is the Normalized Mutual Information (NMI) based on information theory concepts [30]. Given the "real" communities C and the "found" communities C', we compute the *confusion matrix* $\mathbf{N}$ where the element N_{ij} is the number of nodes in the real community i that appear in the found community j, i.e. $N_{ij} = |C_i \cap C'_j|$. The NMI between C and C' is then

$$I(C,C') = \frac{-2\sum_{i=1}^{|C|}\sum_{j=1}^{|C'|} N_{ij}\log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\sum_{i=1}^{|C|} N_{i.}\log(\frac{N_{i.}}{N}) + \sum_{j=1}^{|C'|} N_{.j}\log(\frac{N_{.j}}{N})}$$

where $N_{i.}$ is the sum over row i and $N_{.j}$ is the sum over column j of $\mathbf{N}$. Also, $N = \sum_{i=1}^{|C|} N_{i.} = \sum_{j=1}^{|C'|} N_{.j}$. Note that $\bar{F_1}(C,C')$ and $I(C,C')$ are symmetric functions.

We choose the output clustering of Louvain method as the ground truth for two reasons. First, the evaluation on the real ground truth is already done in [83] and Louvain method is proven to provide high quality communities. Second, the real ground truth is a set of *overlap*

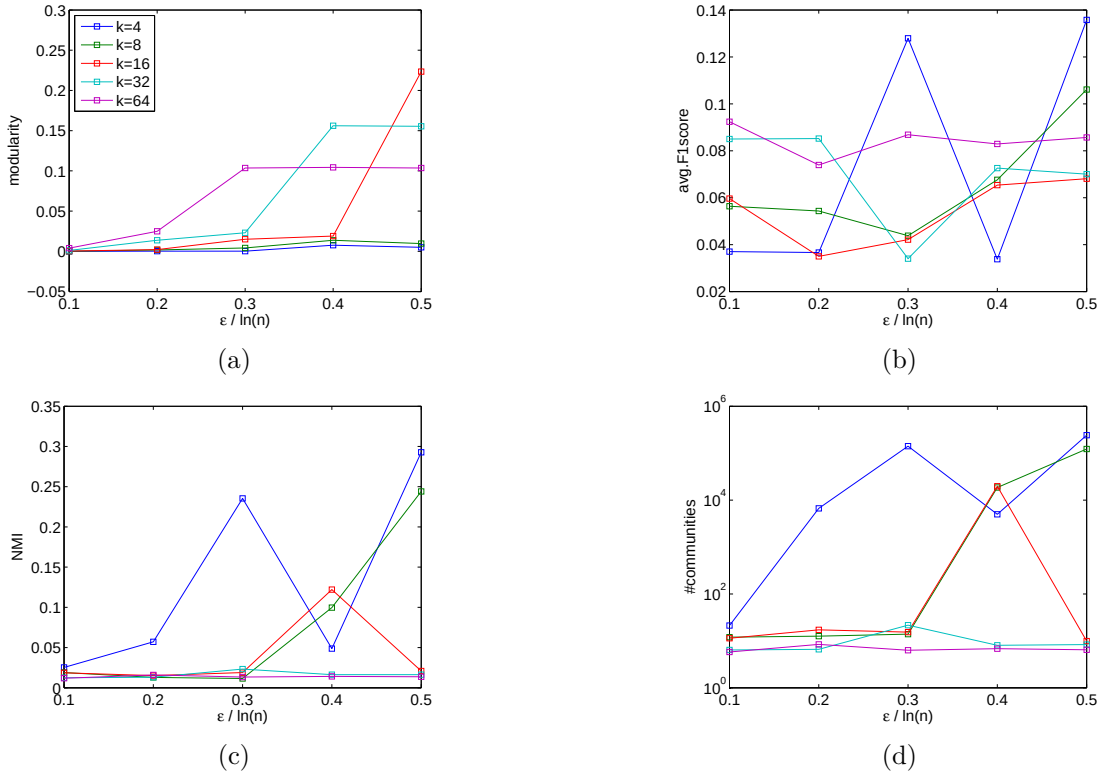


Figure 5.4: LouvainDP on `youtube`

communities whereas the schemes in this chapter output only *non-overlap* communities. The chosen values of ϵ are $\{0.1\ln n, 0.2\ln n, 0.3\ln n, 0.4\ln n, 0.5\ln n\}$. At $\epsilon = 0.5\ln n$, the multiplicative ratio (c.f. Definition 4.1) is $e^{\epsilon} = e^{0.5\ln n} = \sqrt{n}$, a reasonable threshold for privacy protection. As an instance, on `youtube` the ratio is 1065.3 at $\epsilon = 0.5\ln n$.

5.5.2 LouvainDP

We test LouvainDP for the group size $k \in \{4, 8, 16, 32, 64\}$. The results on `youtube` are displayed in Fig. 5.4. We observe the clear separation of two groups $k = 4, 8$ and $k = 16, 32, 64$. As k increases, the modularity increases faster but also saturates sooner. Similar separations appear in avg.F1Score, NMI and the number of communities.

At $\epsilon = 0.5\ln n$ and $k = 4, 8$, the total edge weight in G_1 is very low ($< 0.05m$), so many supernodes of G_1 are disconnected and Louvain method outputs a large number of communities (Fig. 5.4d). The reason is that the threshold θ in LouvainDP is an integral value, so causing abnormal leaps in the total edge weight of G_1. We pick $k = 8, 16$ for the comparative evaluation (Section 5.5.4).

5.5.3 ModDivisive

The effectiveness of ModDivisive is illustrated in Fig. 5.5 for graph `youtube` and $\lambda = 2.0$, $K = 50$. We select six pairs of $(k, maxL)$ by the set {(2,10),(3,7),(4,5), (5,4),(6,4),(10,3)}. Modularity and NMI increase steadily with ϵ while it is not always the case for avg.F1Score. Interestingly, avg.F1Score and NMI show opposite trends for different pairs of $(k, maxL)$. The

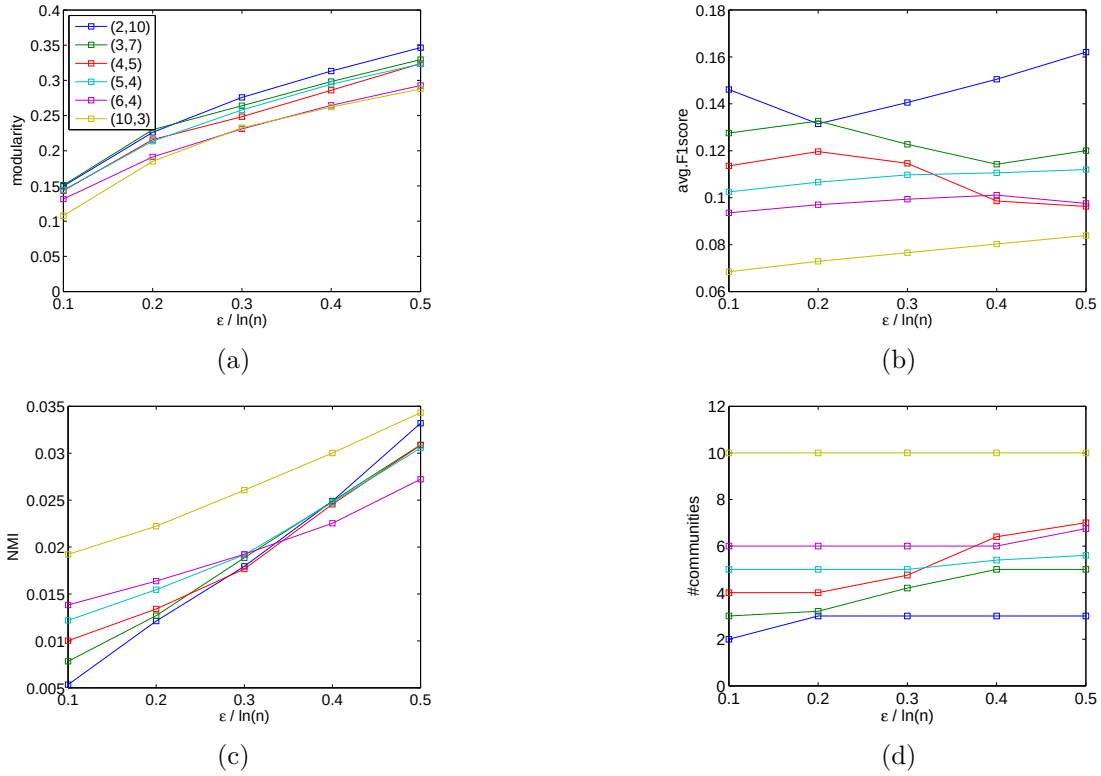


Figure 5.5: ModDivisive on `youtube` with $\lambda = 2.0$, $K = 50$

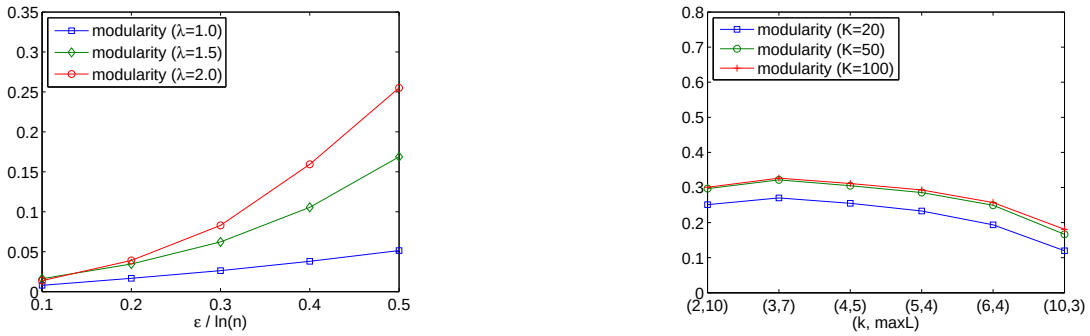
pairs that have higher modularity also provide better avg.F1Score, but lower NMI. The number of communities in the best cut is shown in Fig. 5.5d. Clearly, the small number of communities indicates that ModDivisive's best cut is not far from the root. The reason is the use of $\lambda = 2.0$, i.e. half of privacy budget is reserved to the first level, the half of the rest for the second level and so on. Lower levels receive geometrically smaller privacy budgets, so their partitions get poorer results.

We choose $\lambda = 2.0$ to obtain a good allocation of ϵ among the levels. Fig. 5.6a shows the modularity for different values of λ. Note that $\lambda = 1.0$ means ϵ is equally allocated to the $maxL$ levels. By building a k-ary tree, we reduce considerably the size of the state space $\mathcal{P}$ for MCMC. As a result, we need only a small burn-in factor K. When we look at the Fig. 5.6b, we see that larger $K = 100$ induces only tiny increase of modularity in comparison with that of $K = 50$.

5.5.4 Comparative Evaluation

In this section, we report a comparative evaluation of LouvainDP and ModDivisive against the competitors in Figures 5.7, 5.8, 5.9, 5.10 and 5.11. The dashed lines in subfigures 5.7d, 5.8d, 5.9d, 5.10d and 5.11d represent the ground-truth number of communities by Louvain method. ModDivisive performs best in most of the cases.

On `as20graph` and `ca-AstroPh`, HRG-MCMC outputs the whole nodeset V with the zero modularity while 1K-series, TmF, DER also give useless clusterings. EdgeFlip produces good quality metrics exclusively on `ca-AstroPh` while LouvainDP returns the highest modularity scores on `as20graph`. However, the inherent quadratic complexity of EdgeFlip makes Louvain method fail at $\epsilon = 0.1 \ln n$ and $0.2 \ln n$ for `ca-AstroPh` graph.



(a) λ (b) K (at $\epsilon = 0.5\ln n$)

Figure 5.6: ModDivisive: modularity vs. λ and K on `amazon`

On the three large graphs, ModDivisive dominates the other schemes by a large margin in modularity and avg.F1Score. LouvainDP is the second best in modularity only at $k = 16, \epsilon = 0.5\ln n$. It also provides the best NMI at $k = 8, \epsilon = 0.4, 0.5\ln n$ and $k = 16, \epsilon = 0.5\ln n$. Our proposed HRG-Fixed is consistent with ϵ and has good performance on `dblp` and `youtube`. Note that HRG-MCMC is infeasible on the three large graphs due to its quadratic complexity. Again, 1K-series, TmF and EdgeFlipShrink provide the worst quality scores with the exception of 1K-series's avg.F1Score and NMI on `youtube`.

The runtime of the linear schemes is reported in Fig. 5.12. EdgeFlipShrink, 1K-series, TmF and LouvainDP benefit greatly by running Louvain method on the noisy output graph $\tilde{G}$. ModDivisive and HRG-Fixed also finish their work quickly in $O(K.n.maxL)$ and $O(K.m.\log n)$ respectively.

5.6 Conclusion

We give a big picture of the problem ϵ-DP community detection within the two categories: input and algorithm perturbation. We propose LouvainDP and ModDivisive as the representatives of input and algorithm perturbation respectively. By conducting a comprehensive evaluation, we reveal the advantages of our methods. ModDivisive steadily gives the best modularity and avg.F1Score on large graphs while LouvainDP outperforms the remaining input perturbation competitors in certain settings. HRG-MCMC/HRG-Fixed give low modularity clusterings, indicating the limitation of the log-likelihood in divisive CD. The input perturbation schemes DER, EF, 1K-series and TmF hardly deliver any good node clustering except EF on the two medium-sized graphs. For future work, we plan to develop an ϵ-DP agglomerative scheme based on Louvain method and extend our work for directed graphs and overlapping community detection under differential privacy.

In the following chapter, we will propose and solve a novel problem: private link exchange over social graphs.

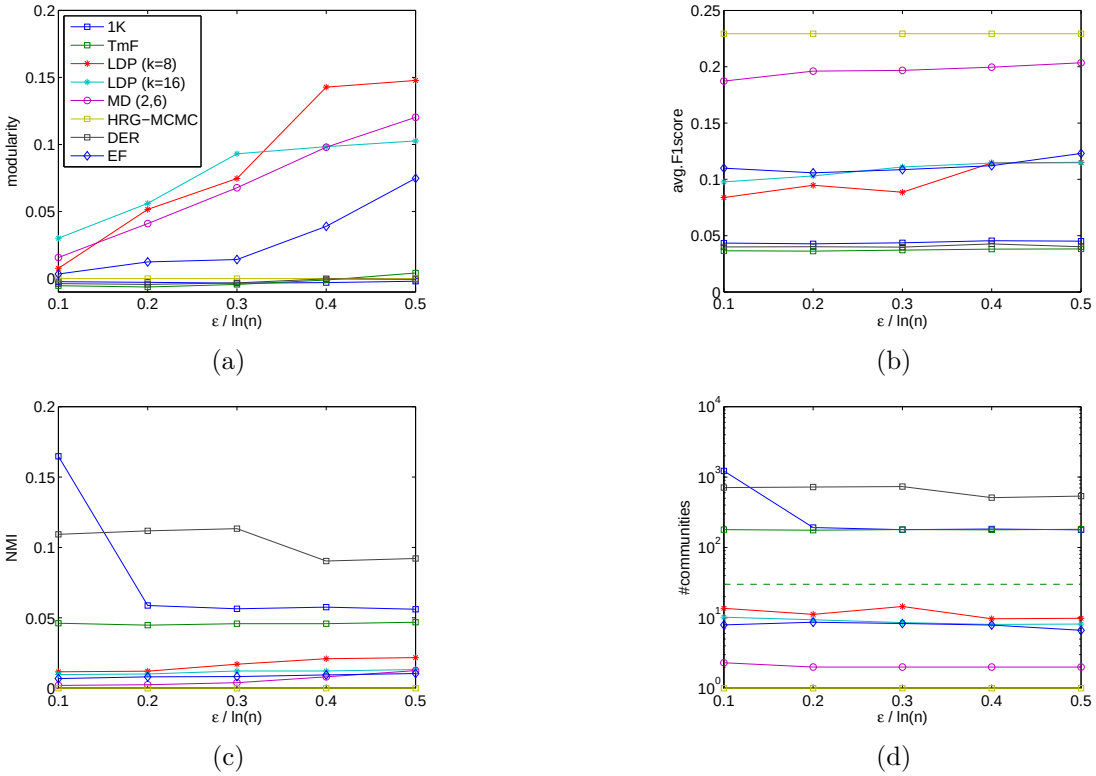


Figure 5.7: Quality metrics and the number of communities (`as20graph`)

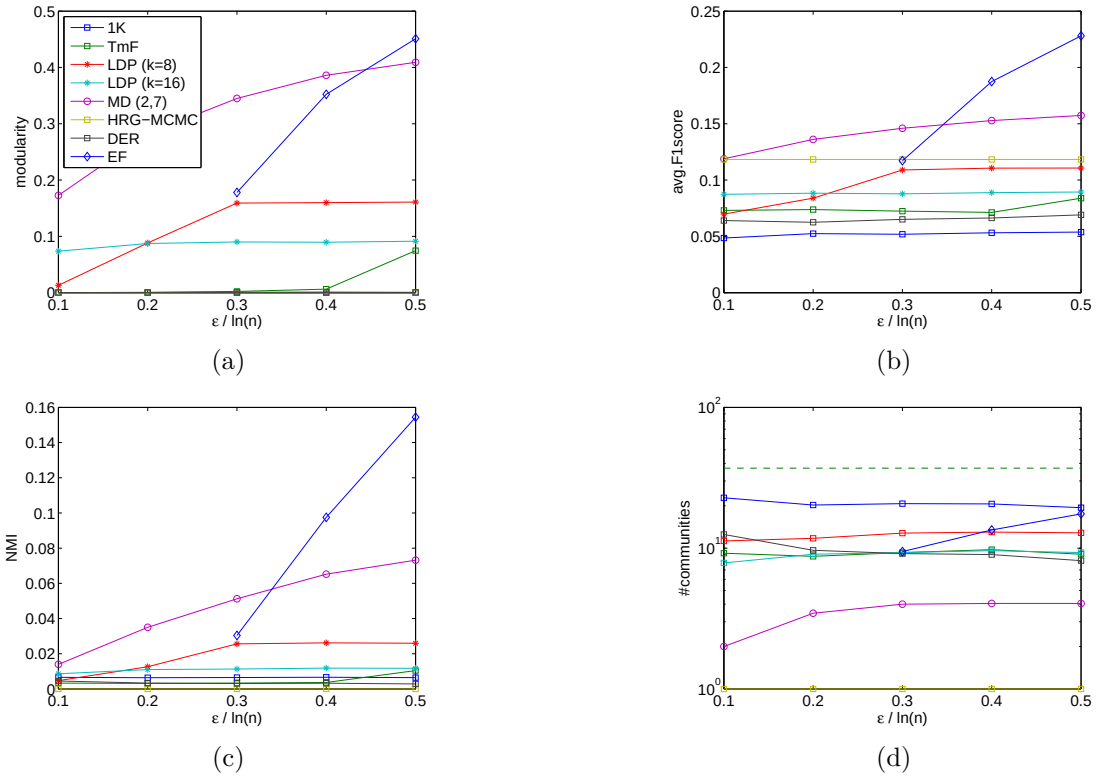


Figure 5.8: Quality metrics and the number of communities (`ca-AstroPh`)

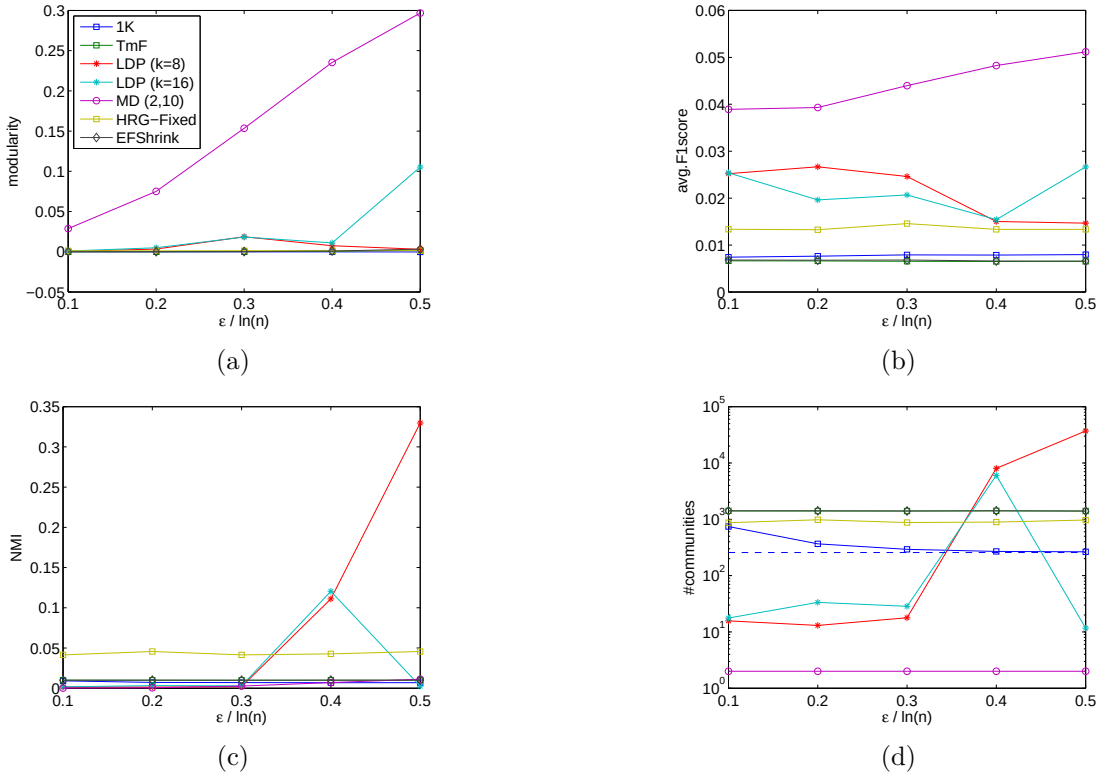


Figure 5.9: Quality metrics and the number of communities (`amazon`)

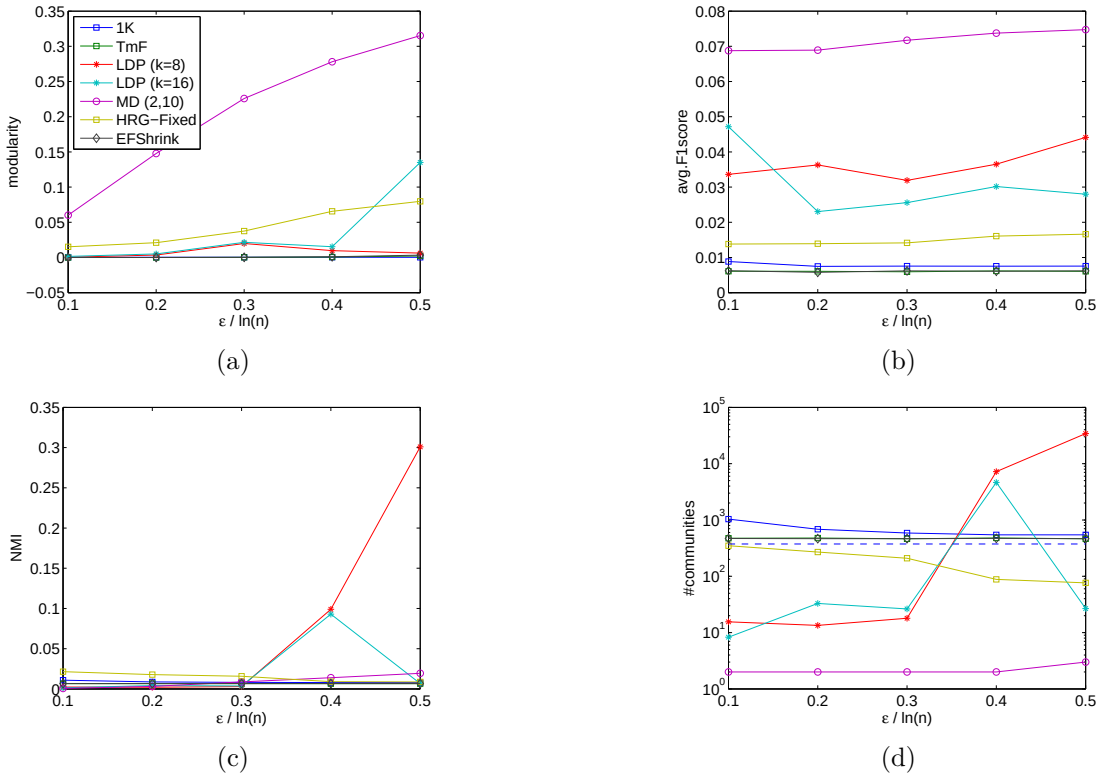


Figure 5.10: Quality metrics and the number of communities (`dblp`)

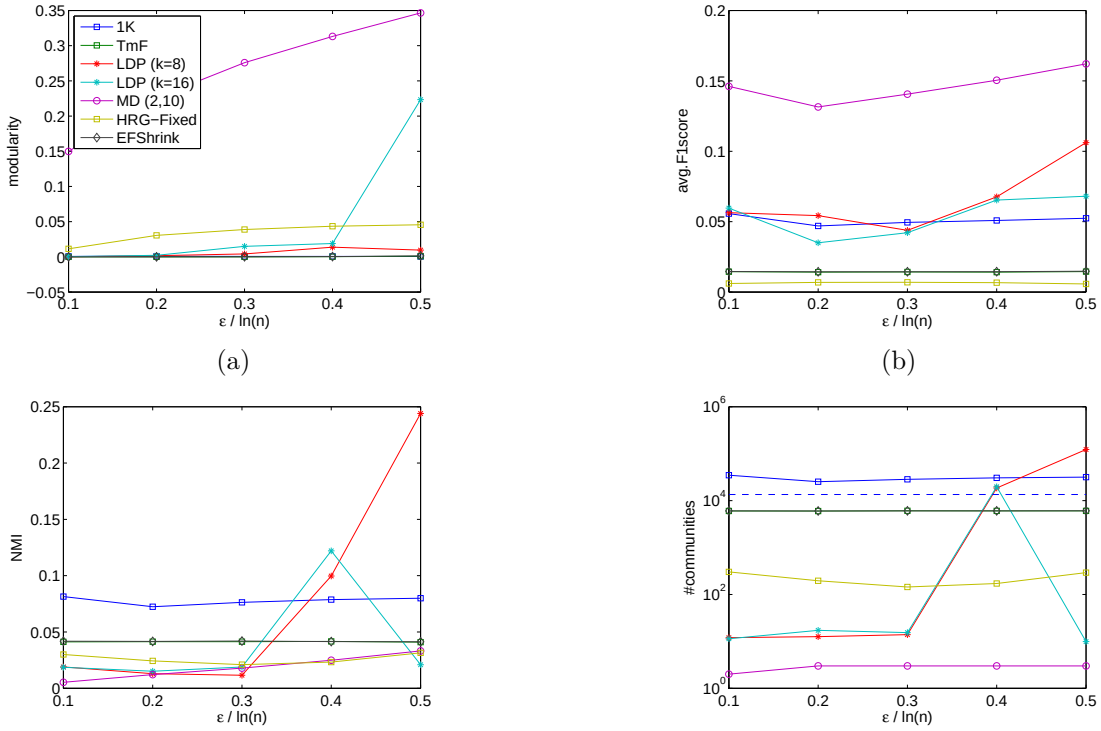
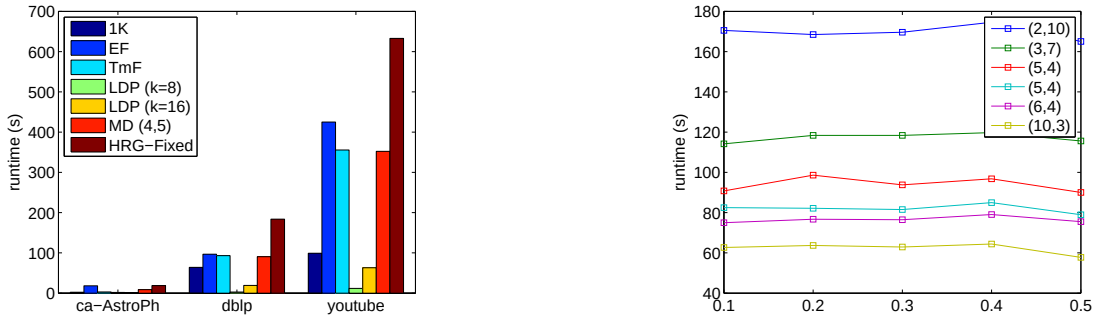


Figure 5.11: Quality metrics and the number of communities (`youtube`)



(a) All schemes ($\epsilon = 0.5 \ln n$)

(b) ModDivisive on `amazon`

Figure 5.12: Runtime

Chapter 6

Private Link Exchange over Social Graphs

Contents

6.1 Introduction

Online social networks (OSN) have grown significantly over the last ten years with billions of active users using a variety of social network services. OSNs have revolutionized the way people interact. People join social networking sites to connect and communicate with their friends in real-time. They share interests and activities across political, economic, and geographic borders. As social network sites continue to develop both in number and size, the service providers accumulate unprecedented amount of information about OSN users. As a result, social networks

are a valuable data source for research on information societies. In particular, underlying social graphs play a key role in understanding how people form communities, how the OSNs suggest friendship to two users who do not know each other but have many common friends, etc. However, social graphs are not published in clear form due to serious privacy concerns. Instead, they are anonymized in various forms and published to third party consumers such as sociologists, epidemiologists, advertisers and criminologists. Alternatively, social networking sites provide APIs [14] for data crawlers at limited rates and within privacy constraints (e.g. only public friend lists are available). Using this method, the data crawlers can collect friendship information and build a partial (local) view of the target social graph.

To overcome the constraints set by the service providers, we can start from user perspective, i.e. the contributors of OSNs. More precisely, if users cautiously collaborate with one another, they can exchange *noisy* friend lists (containing fake friendships) with their neighbors in several rounds to get better views of the true social graph. Our ideas are based on the fact that user IDs are public (e.g. Facebook profiles are searchable [1]) but the friendships are not so, except when a user leaves his friend list in public mode. Using public user IDs, any user can claim fake links from himself to the users not in his friend list.

The aggregation problem in this chapter is unique in the sense that the disseminated data over the links are the links themselves. However, there exist fundamental questions about the feasibility of this model. The first question is how to define simple and effective privacy concepts for the link exchange processes. The second question comes from the high volume of link lists in exchange which may increase exponentially round after round. While storage and computation complexity may not be big problems, communication costs are non-trivial. We address both questions by a simple (α, β)-exchange protocol with or without Bloom filters. To protect true links from inference attacks, we add fake links which are *beta*-fraction of true links. Furthermore, we realize the attenuated propagation of links via the parameter $\alpha \leq 1$.

Basically, we assume that users are *honest-but-curious* (HbC), i.e. they follow the protocol but try to figure out true friendships among noisy friend lists. To preserve link privacy, each node obfuscates its friend list by adding fake links originating from itself to a number of nodes not in its friend list. Then in exchange stage, nodes share only with their friends a fraction of noisy links they possess.

Our contributions are summarized as follows:

- We introduce a novel private link exchange problem as an alternative to social graph crawling and centralized anonymization of data. The problem is distributed and provides a privacy/utility tradeoff for all nodes.
- We present two schemes for (α, β)-exchange protocol: Baseline and Bloom filter based. We protect the true links by adding fake links and requiring the propagation probability of links to be attenuated by distance. We analyze the advantages and disadvantages of each scheme.
- We evaluate our proposed schemes on various synthetic graph models and draw a number of critical findings.

The chapter is organized as follows. Section 6.2 briefly introduces key concepts of Bloom filter and our link exchange model. In Section 6.3, we present Baseline (α, β)-exchange that realizes the exchange model by sending noisy link lists in clear form. Section 6.4 describes Bloom filter version of (α, β)-exchange with constant complexities and better privacy. We validate the

[14] https://developers.facebook.com/docs/graph-api

proposed schemes in Section 6.5. Finally, we present our remarks and suggest future work in Section 6.6.

Table 6.1 summarizes notations used in this chapter.

Table 6.1: List of notations

Symbol	Definition
$G = (V, E)$	social graph with $N = \|V\|$ and $M = \|E_G\|$
$A(G)$	adjacency matrix of G
D	degree sequence of G (column vector)
$Diam(G)$	diameter of G
$N(u)$	neighbors of node u in G, $d_u = \|N(u)\|$
T	number of exchange rounds
(v, w)	true link between v and w
$(v \to w)$	fake link generated by v
$L_u(t)$	set of links possessed by u at round t
$L_{uv}(t)$	set of links u sends to v at time t
$\propto$	uniformly at random sampling without replacement
α	fraction of links shared between a pair of nodes
β	fraction of fake links generated at $t = 0$
m	number of bits in Bloom filter
k	number of hash functions used in Bloom filter
n	number of elements in Bloom filter
$Bf_u(t)$	Bloom filter possessed by u at round t
$Bf_{uv}(t)$	Bloom filter u sends to v at time t

6.2 Preliminaries

In this section, we present the exchange model and attack model. Then we review key concepts about Bloom filter.

6.2.1 Exchange Model

We consider a distributed exchange model in which each node possesses his friend list and all nodes participate in the exchange protocol. We work on the following assumptions

- **Assumption 1** The space of node IDs is public. A node can generate fake links to any node. All friend lists (true links) are private, i.e. the existence of true link (u, v) is surely known to u and v only.
- **Assumption 2** A node exchanges messages with its neighbors only. Interacting with neighbors is based on an intuition of trusted relationships: we trust our friends more than any strangers.
- **Assumption 3** A synchronous model is guaranteed by *round-tagged* messages. It means a node prepares the message for round $t + 1$ if and only if it has received all t-th round messages from his friends.
- **Assumption 4** All nodes are honest-but-curious. They follow the protocol but try to infer true links among noisy links.

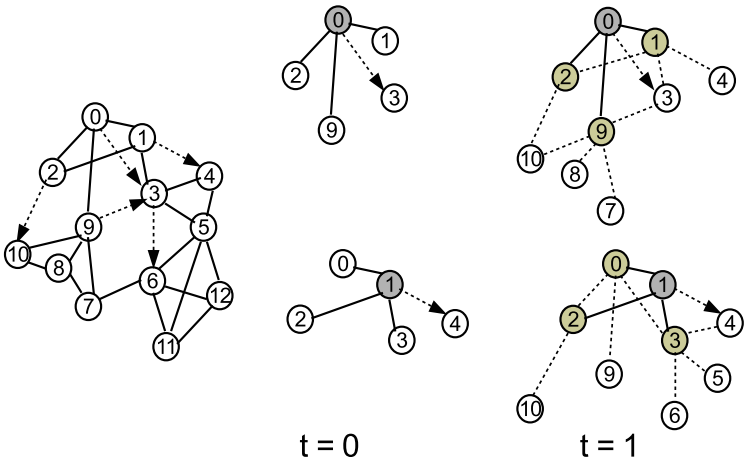


Figure 6.1: Link exchange with $\alpha = 1$, $\beta = 1/3$

Fig. 6.1 illustrates the exchange model. At round $t = 0$ (initial round), each node u prepare a noisy friend list by adding some fake links (u, v) (i.e. links from u to some people not in his friend list). This is feasible because all user IDs are public (e.g. [1]). For example, node 0 adds a fake link (0,3) and his noisy friend list {(0,1), (0,2), (0,3)} is ready to be exchanged. Similarly, the other nodes prepare their noisy friend lists as in Fig 6.1. At round $t = 1$, all nodes send and receive noisy friend lists from their neighbors. The local views of nodes 0 and 1 at $t = 1$ are shown in Fig. 6.1 where the solid lines (resp. the dashed arrows) are the true links (resp. fake links) known by the node and the dashed lines represent noisy links received at the node.

6.2.2 Attack Model

We consider honest-but-curious users (nodes) who follow the protocol but try to infer true links among noisy links. We propose a simple inference attack based on frequencies of links arriving to a node. Given a link (v, w) (a true link or a fake link) arriving to node u, if (v, w) does not exist in u's local view, it will be added. Otherwise, its frequency is increased by 1. At the end of the protocol, each node sorts all links in its local view by frequency and selects top K links as true links. How to select the value of K will be discussed later.

By splitting noisy links into two sets of links as above, the inference capability of each node is evaluated by common measures [37]: True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN). As we will see in Section 6.3, the parameter α introduces an *attenuation effect* for link propagation when $\alpha < 1$. Given a link e, nodes farther from e have lower chance of getting this link. This effect adds another dimension to our privacy model.

6.2.3 Bloom Filter

The Bloom filter is a space-efficient probabilistic data structure that supports set membership queries. It was first conceived by Burton Howard Bloom in 1970 [10]. It is used to test whether an element is a member of a set and can result in false positives (claiming an element to belong to the set when it was not inserted), but never in false negatives (reporting an inserted element not in the set).

An empty Bloom filter is an array of m bits, all set to zero. There must also be k different hash functions defined, each of which maps or hashes some set element x to one of the m array

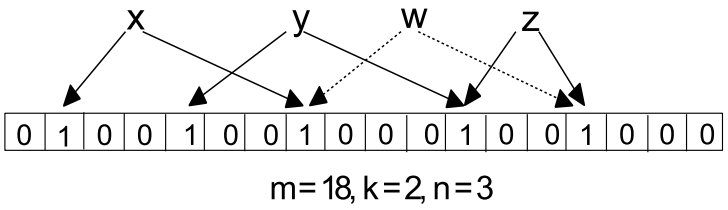


Figure 6.2: Bloom filter

positions with a *uniform* random distribution. We denote n as the number of elements inserted into the Bloom filter. Fig. 6.2 gives an example of Bloom filter with $m = 18$, $k = 2$ and $n = 3$. The MD5 hash algorithm is a popular choice for the hash functions. When an element not in the set w is looked up, it will be hashed by the k hash functions into bit positions. If one of the positions is zero, we conclude that w is not in the set. It may happen that all the bit positions of an element have been set. When this occurs, the Bloom filter will erroneously report that the element is a member of the set, also known as false positives. Fig. 6.2 shows w as a false positive.

Given the three parameters m, k and n, the false positive probability is (see [16]).

$$p = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \approx (1 - e^{-kn/m})^k \tag{6.1}$$

The false positive probability decreases as m increases or n decreases. Given m and n, the probability of false positives $(1 - e^{-kn/m})^k$ is minimized at $k = k_{opt} = \frac{m}{n} \ln 2$ (see [16]). In this case, the false positive rate $p = (1/2)^k$ or equivalently

$$k = -\log_2 p \tag{6.2}$$

6.3 Baseline (α, β)-exchange

In this section, we present the main ideas of our proposed (α, β)-exchange and the improvements using Bloom filters.

6.3.1 Overview

As shown in Section 6.2.1, the link exchange protocol is straightforward. At the beginning of the protocol, all the nodes agree on the number of rounds T and the two parameters $\alpha \in [0, 1]$, $\beta \geq 0$. Then, each node u prepares his own noisy friend list $L_u(0)$ by setting $L_u(0) = \{(u, v)|v \in N(u)\}$ and adding $\beta N(u)$ fake links in the form $(u \to w)$ where $w \notin N(u)$. At $t = 1$, each node u makes a noisy list $L_{uv}(1)$ for every neighbor v so that $L_{uv}(1)$ contains $\alpha|L_u(0)|$ links sampled from $L_u(0)$. Similarly, node v prepares a noisy list $L_{vu}(1)$ for u. All the nodes sends and receives noisy link lists. Next, each node aggregates noisy link lists by removing duplicate links (if any) and obtains his local view of graph by $L_u(1)$. The round $t = 1$ finishes.

At $t = 2$, the process repeats: all nodes u makes a noisy list $L_{uv}(2)$ for every neighbor v that contains $\alpha|L_u(1)|$ links sampled from $L_u(1)$. They exchange noisy link lists and after receiving all $L_{vu}(2)$ from his friends, node u updates his local view and gets $L_u(2)$. When $t = T$, the protocol terminates.

6.3.2 Baseline Scheme

The idea in the previous section is called Baseline (α, β)-exchange as all noisy link lists are in clear form. Algorithm 15 shows steps for Baseline (α, β)-exchange.

Algorithm 15 Baseline (α, β)-exchange

Input: undirected graph $G = (V, E)$, parameters $\alpha \in [0, 1]$, $\beta \geq 0$, number of rounds T
Output: noisy local views of graph $L_u(T), u \in V$

1: // initialization stage
2: **for** $u \in V$ **do**
3: $\quad Fa(u) = \{(u \to w) | w \notin N(u)\}$ s.t. $|Fa(u)| = \beta |N(u)|$
4: $\quad L_u(0) = \{(u, v) | v \in N(u)\} \cup Fa(u)$
5: // exchange stage
6: **for** $t = 1..T$ **do**
7: $\quad$ **for** $(u, v) \in E$ **do**
8: $\quad\quad u : L_{uv}(t) \propto L_u(t-1)$ s.t. $|L_{uv}(t)| = \alpha |L_u(t-1)|$
9: $\quad\quad v : L_{vu}(t) \propto L_v(t-1)$ s.t. $|L_{vu}(t)| = \alpha |L_v(t-1)|$
10: $\quad\quad u$ sends $L_{uv}(t)$ to v
11: $\quad\quad v$ sends $L_{vu}(t)$ to u
12: $\quad$ **for** $u \in V$ **do**
13: $\quad\quad L_u(t) = L_u(t-1) \cup \bigcup_{v \in N(u)} L_{vu}(t)$
return $L_u(T), u \in V$

Given the graph structure $G = (V, E)$, two parameters $\alpha \in [0, 1]$, $\beta \geq 0$ and the number of rounds T. The protocol takes place in two stages. In initialization stage, each node u prepares his own noisy friend list $L_u(0)$ by adding $\beta N(u)$ fake links in the form (u, w) where $w \notin N(u)$ (Lines 3 and 4). In exchange stage (Lines 6-13), at round t, each node u makes a noisy list $L_{uv}(t)$ for every neighbor v that contains $\alpha |L_u(t)|$ links sampled from $L_u(t)$. The exchange happens on every relationship (true link). Each node takes the union of all noisy links he receives before starting the next round.

Faster Simulation in A Single PC

For simulation in a single PC, storing all link lists for all nodes in clear form is a costly solution. Moreover, the union operation on lists is time-consuming. We present here a technique to reduce the memory footprint and processing time using bit sets.

Fig. 6.3 outlines the idea. We have $M' = (1 + 2\beta)|E_G|$ distinct links consisting of $|E_G|$ true links and $2\beta|E_G|$ fake links. By indexing M' links from 0 to $M' - 1$, the noisy link list at each node is stored in a bit set of size M'. The union of link lists (Line 13 Algorithm 15) is equivalent to an OR operation between bit sets. To prepare $L_{uv}(t)$ for link exchange in round t, node u must recover link IDs in its bit set.

We emphasize that indexing links and storing link IDs in bit sets are only for simulation. In reality, the number of links are unknown to all nodes, so they must run Baseline or Bloom filter (Section 6.4) protocol.

For the case $\alpha = 1$, the exchange volume is reduced further if each node u sends only "new" links, i.e. the links that do not exist in u's list in the previous round. Fig. 6.4 visualizes this idea in which "new" links are in shaded region and old links are in white region. Note that

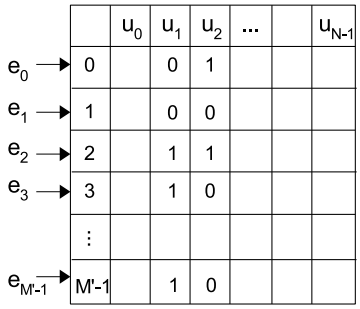


Figure 6.3: Fast simulation using bit sets (column vectors)

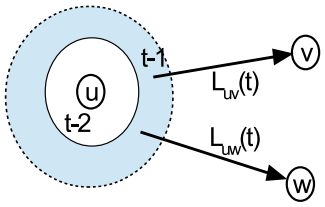


Figure 6.4: Incremental volume for $\alpha = 1$

the incremental volume is valid only for $\alpha = 1$. When $\alpha < 1$, the phenomenon of multipath propagation (Fig. 6.5) requires both new and old links to be sampled with probability α.

Utility-Oriented Initialization

Baseline scheme in the previous section lets a node u generate fake links by connecting u to a certain number of nodes not in u's friend list. This initialization may make local sub graphs at the final round have distorted path distributions due to many fake links connecting faraway nodes. Distorted path distributions reduce the "utility" perceived at each node. Based on the observation of using fake links connecting nearby nodes [76], we suggest a utility-oriented improvement by two-round initialization. We call a fake link $(u \to v)$ *distance-2 link* if $d(u, v) = 2$. For example, in Fig.6.1 $(0 \to 3)$ is a distance-2 fake link while $(2 \to 10)$ is not. Correspondingly, v is called a *distance-2 node* w.r.t u.

We introduce a new parameter $\gamma \in [0, 1]$ which stipulates that each node u create $\gamma\beta d_u$ fake links at $t = 0$ and exchange $\alpha(1 + \gamma\beta)d_u$ randomly chosen links to each of its neighbors. Node u collects node IDs and save them in the set ID_u. At $t = 1$, node u uses node IDs in ID_u to create $\alpha(1 - \gamma)\beta d_u$ fake links. Algorithm 16 implements this idea.

The number of distance-2 nodes that u collects in Line 7 of Algorithm 16 is $\alpha(\sum_{v \in N(u)} d_v - d_u - 2Tri(u))$ where $Tri(u)$ is the number of triangles with u as a vertex. Assuming that the set $Fa_0(u)$ contains no distance-2 links (Line 3 Algorithm 16). The number of non-distance-2 nodes that u collects is $\sum_{v \in N(u)} \alpha\gamma\beta d_v$. The expected number of distance-2 links that u can create is

$$L2(u) = \frac{(1 - \gamma)(\sum_{v \in N(u)} d_v - d_u - 2Tri(u))}{[\sum_{v \in N(u)} d_v - d_u - 2Tri(u)] + \sum_{v \in N(u)} \gamma\beta d_v}$$

$L2(u)$ is a decreasing function of γ. All nodes have the highest (resp. lowest) number of distance-2 fake links at $\gamma = 0$ (resp. $\gamma = 1$). The case of $\gamma = 1$ reduces to standard initialization (Lines 2-4 Algorithm 15).

Algorithm 16 Two-round Initialization

Input: undirected graph $G = (V, E)$, parameters $\alpha, \gamma \in [0, 1]$, $\beta \geq 0$
Output: each node u issues βd_u fake links

1: // t = 0
2: **for** $u \in V$ **do**
3: $\quad Fa_0(u) = \{(u \to w) | w \notin N(u)\}$ s.t. $|Fa_0(u)| = \gamma\beta|N(u)|$
4: $\quad L_u(0) = \{(u, v) | v \in N(u)\} \cup Fa_0(u)$
5: // t = 1
6: **for** $(u, v) \in E$ **do**
7: $\quad u$ and v exchange α-fraction of their links
8: **for** $u \in V$ **do**
9: $\quad u$ aggregates all links it knows into $L_u(1)$
10: $\quad ID_u = \{w | w = v_1 \wedge w = v_2, (v_1, v_2) \in L_u(1)\} \setminus \{u, N(u)\}$
11: $\quad Fa_1(u) = \{(u \to w) | w \in ID_u\}$
12: $\quad\quad$ s.t. $|Fa_1(u)| = \alpha(1 - \gamma)\beta|N(u)|$
13: $\quad L_u(1) = L_u(1) \cup Fa_1(u)$

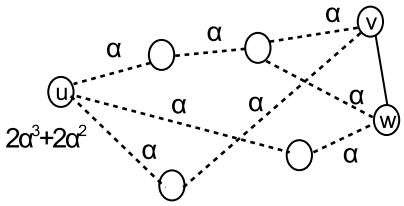


Figure 6.5: Multipath link propagation

6.3.3 Complexity Analysis

Let A be the adjacency matrix of G and D be the column vector of degree sequence of nodes, the number of links at all nodes is upper bounded by the following vector, where I_N is the identity matrix of size N.

$$LU(t) = (I_N + \alpha A)^t (1 + \beta) D \tag{6.3}$$

We say $LU(t)$ an "upper-bound" because $LU(t)$ accepts duplicate links. More precisely, let $LU_u(t)$ and $LU_{uv}(t)$ be the noisy link lists at node u and for exchange without removing duplicate links as in Line 13 Algorithm 15. We have $LU_u(t) = LU_u(t-1) + \sum_{v \in N(u)} LU_{vu}(t)$, where "+" denotes *multiset* semantics. Clearly, $L_u(t) < LU_u(t)$.

Note that the number of rounds T can be small because of the following analysis. We have four simple facts (see Fig. 6.5)

1. for $\alpha = 1$, a true link (v, w) is propagated to node u at round t iff $\min\{d(u, v), d(u, w)\} = t$.
2. for $\alpha = 1$, a fake link $(v \to w)$ is propagated to node u at round t iff $d(u, v) = t$.
3. for $\alpha < 1$, a true link (v, w) is propagated to node u at round t with probability $\min(\sum_{p_l \in P(u,v) \cup P(u,w)} \alpha^l, 1)$. Here p_l is a path of length l from u to v or w (l is in the range $[1..t]$).
4. for $\alpha < 1$, a fake link $(v \to w)$ is propagated to node u at round t with probability $\min(\sum_{p_l \in P(u,v)} \alpha^l, 1)$ where p_l is defined in the previous case.

We consider three cases.

Case 1: $\alpha = 1, \beta = 0$ In this case, there is no fake links. Using Fact 1, we have $|L_u(Diam(G) - 1)| = m$, i.e. every node u receives all true links in G after $(Diam(G) - 1$ rounds.

Case 2: $\alpha = 1, \beta > 0$ In this case, there are $2\beta m$ fake links. Using Facts 1 and 2, we have $|L_u(Diam(G))| = (1 + 2\beta)m$, i.e. every node u receives all true links and fake links in G after $Diam(G)$ rounds.

Case 3: $\alpha < 1, \beta \geq 0$ In this case, there are $2\beta m$ fake links. Using Facts 3 and 4, every node u receives all true links (v, w) in G after T rounds if

$$\sum_{t=1}^{T} [(\alpha A)^t]_{vu} + [(\alpha A)^t]_{wu} \geq 1 \tag{6.4}$$

and all fake links $(v \to w)$ if

$$\sum_{t=1}^{T} [(\alpha A)^t]_{vu} \geq 1 \tag{6.5}$$

The protocol's complexity is measured in storage, computation and communication. Because all links are stored in clear form, all complexities increase round by round (except the trivial case $\alpha = 0$). They are also upper bounded by the total links in graph, which is $(1 + 2\beta)|E_G|$. Intuitively, low-degree nodes will incur lower complexities than high-degree nodes. However, as t increases, the gap gets narrower. In Section 6.4, we will achieve constant complexities by using Bloom filters.

6.3.4 Privacy Analysis

In this section, we discuss the link inference attacks that can be mounted by nodes. Each node has knowledge about the true links connecting itself to its neighbors and the fake links it creates before the first round as well as the fake links pointing to it. The remaining links (denoted as B_u) stored at node u are subject to an inference attack by u. As discussed in Section 6.2.2, u may mount an inference attack by sorting links in B_u by weight and picks top-K links as true links.

In Baseline (α, β)-exchange, the ratio of true links over fake links is $\frac{1}{\beta}$. Each user, therefore, can set $K = \frac{|B_u|}{1+\beta}$ and divide B_u into two sets T_u (predicted true links) and F_u (predicted fake links). The numbers of true positives, true negatives, false positives and false negatives are

$$TP_u = |E_G \cap T_u| \, , FP_u = |T_u \setminus E_G| \tag{6.6}$$

$$FN_u = |E_G \cap F_u| \, , TN_u = |F_u \setminus E_G| \tag{6.7}$$

The precision, recall and F1 score are defined as

$$Prec = \frac{TP_u}{TP_u + FP_u} \tag{6.8}$$

$$Recall = \frac{TP_u}{TP_u + FN_u} \tag{6.9}$$

$$F1 = \frac{2.Prec.Recall}{Prec + Recall} \tag{6.10}$$

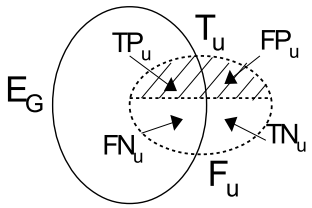


Figure 6.6: Inference attack measures

6.4 Bloom Filter Based Scheme

6.4.1 Motivation

Baseline (α, β)-exchange has several drawbacks that motivate a better approach. First, all link lists are in clear form, allowing nodes to store link frequencies for inference attack (Section 6.3.4). If we obfuscate link lists, this kind of attack may be mitigated. Hashing could be a solution. Second, sending link lists in clear form may incur high communication cost, especially at high degree nodes. Assuming that all node IDs are in range $\{0...2^{32}-1\}$, i.e. each ID needs 4 bytes, each link is encoded in 8 bytes. Given a link list, a better way to encode it is to store all links (u, v_i) incident to u by $\{u|\{v_i\}\}$. In this way, the message length for a link list can be reduced up to 50%. In average, each link costs between 32-bit to 64-bit. Using Bloom filters, the number of bits per link may be reduced. For example, with $k = 4$, the number of bits per link is $k/\ln 2 \approx 5.8$.

This section introduces a Bloom filter based approach. Compared to Baseline approach, it has several advantages and limitations. Bloom filters, by encoding links in compact forms, reduce the storage and communication costs. The computation at each node is also much simpler thanks to logical OR operation compared to set unions in Baseline.

6.4.2 Bloom Filter Based Scheme

Algorithm 17 describes steps of Bloom filter version of (α, β)-exchange. As for inputs, we add a global false positive probability p and the number of links $|E_G|$. As analyzed in [16], the number of hash functions k is set to $\lceil -\log_2 p \rceil$ (Line 2). The number of bits per link is $c = k/\ln 2$ (Line 3). The length of every Bloom filter is $m = c.|E_G|$ (Line 4). Then, each node u initializes its Bloom filter $Bf_u(0)$ by hashing all links in $L_u(0)$ using k hash functions. At the same time, all nodes send their noisy links $L_u(0)$ to the coordinator who will gather all links into the list L. This list will be used in the recovery stage.

In the exchange stage, each pair of nodes (u, v) prepare and exchange noisy link lists in encoded form $Bf_{uv}(t)$ and $Bf_{vu}(t)$ (Lines 14-18). Before the next round, each node aggregates all Bloom filters sent to it by taking the OR operation. (Lines 19 and 20). Finally, the recovery stage helps each node to obtain its noisy local view $L_u(T)$. In this stage, the coordinator sends to L to all nodes. If we omit the role of the coordinator (Lines 5,11 and 23), each node u has to try hash $\frac{N(N-1)}{2}$ possible links against its final Bloom filter $Bf_u(T)$.

Bit Erasure

Because Bloom filters store links information in encoded form, we have to simulate the α-sampling steps (Lines 8 and 9, Algorithm 15).

α-sampling is equivalent to "deletion" of $(1-\alpha)|Bf_u(t-1)|$ elements from $Bf_u(t-1)$. We can

Algorithm 17 Bloom filter (α, β)-exchange

Input: undirected graph $G = (V, E)$, parameters $\alpha \in [0, 1]$, $\beta \geq 0$, number of rounds T, false positive probability p

Output: noisy local views of graph $L_u(T), u \in V$

// initialization stage
$k = \lceil -\log_2 p \rceil$ (see equation (6.2))
$c = k/\ln 2$
$m = c.|E_G|$
$L = \emptyset$
for $u \in V$ **do**
    $Bf_u(0) = \text{BloomFilter(k,m,c)}$
    $Fa(u) = \{(u \to w) | w \notin N(u)\}$ s.t. $|Fa(u)| = \beta |N(u)|$
    $L_u(0) = \{(u, v) | v \in N(u)\} \cup Fa(u)$
    Hash all $e \in L_u(0)$ into $Bf_u(0)$
    $L = L \cup L_u(0)$
// exchange stage
for $t = 1..T$ **do**
    for $(u, v) \in E$ **do**
        u prepares $Bf_{uv}(t) = \text{BitErasure}(Bf_u(t-1), \alpha)$
        v prepares $Bf_{vu}(t) = \text{BitErasure}(Bf_v(t-1), \alpha)$
        u sends $Bf_{uv}(t)$ to v
        v sends $Bf_{vu}(t)$ to u
    for $u \in V$ **do**
        $Bf_u(t) = Bf_u(t-1) \vee \bigvee_{v \in N(u)} Bf_{vu}(t)$
// link recovery stage
for $u \in V$ **do**
    $L_u(T) = \text{Hash}(L, Bf_u(T))$
return $L_u(T), u \in V$

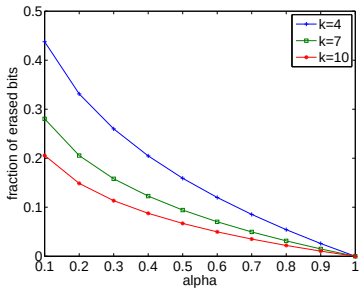


Figure 6.7: Fraction of erased bits as a function of α and k

perform this operation by recovering elements in $Bf_u(t-1)$ then explicitly keeping α-fraction of elements and hashing these elements to an empty Bloom filter. This approach, however, is costly because the node must try $\frac{N(N-1)}{2}$ possible links. As a result, an implicit removal of $(1-\alpha)$-fraction of elements is needed.

Resetting one bit causes one or several misses (false negatives) and possibly reduces false positives. For example, resetting the second bit in Bloom filter (Fig. 6.2) makes x a false negative whereas resetting the 12th-bit makes both y and z disappear. Moreover, if the 8-th bit is reset, x becomes a false negative and w is no longer a false positive.

Let m_1 be the number of 1-bits in Bloom filter $Bf_u(t-1)$ and s be the number of randomly reset bits ($s < m_1$), the probability of a true positive remaining in Bloom filter is

$$(1-\frac{s}{m_1})^k \tag{6.11}$$

If omitting the effect of false positives (which is reduced as illustrated above), the formula (6.11) is exactly the sampling fraction α. In other words,

$$\alpha = (1-\frac{s}{m_1})^k \Rightarrow s = m_1(1-\alpha^{1/k}) \tag{6.12}$$

We can see that s is a decreasing function of α and k. An illustration of this fact is shown in Fig. 6.7.

Algorithm 18 realizes α-sampling implicitly via bit erasure.

Algorithm 18 Bit Erasure

Input: Bloom filter B, parameter $\alpha \in [0,1]$, number of hashes k
Output: Bloom filter B' that contains approximately α fraction of elements in B

1: $B' = B$
2: $M_1 = \{i | B(i) = 1\}$
3: $m_1 = |M_1|$
4: $s = \lfloor m_1(1-\alpha^{1/k} \rfloor$
5: randomly reset s bits in m_1 positions of B'
6: **return** B'

Bloom Filter Compression

In Algorithm 17, all Bloom filters stored at nodes and transmitted between nodes are of length m bits where $m = |E_G|k/\ln 2$. For $p = 0.1$, we have $k = 4$ and $m \approx 5.8|E_G|$. For $p = 0.01$, we have $k = 7$ and $m \approx 10.1|E_G|$. For million-scale graphs with hundreds of millions of links,

the length of Bloom filters would be hundreds of megabytes. This is undesirable for message transmission although storage and computation are not big problems. However, we observe that as in Baseline (α, β)-exchange, not all messages have the length of $\Theta(E_G)$. Thus, lossless data compression is a useful tool for Bloom filter exchange.

Arithmetic coding [67] is such a lossless compression scheme. Arithmetic coding differs from other forms of entropy encoding, such as Huffman coding [48]. Huffman coding separates the input into component symbols with symbol probabilities approximated by negative powers of two and replaces each with a code. Arithmetic coding encodes the entire message into a single number, a fraction f where $0.0 \leq f < 1.0$. Arithmetic coding runs in linear time [67].

6.4.3 Complexity and Privacy Analysis

Thanks to constant sizes of bit arrays and constant time for OR operations, the total communication cost of Bloom Filter scheme is constant and the aggregation of noisy link lists is constant too. However, Bloom Filter scheme incurs an extra recovery step at all nodes. Each node needs to download the full noisy link set L from the coordinator. As we confirm in Section 6.5.2, the exchange time of Bloom Filter scheme is much lower than that of Baseline, but the recovery step costs higher time complexity.

As mentioned in Section 6.4.1, all link lists are obfuscated in Bloom filters, frequency-based inference attacks may be mitigated if the set of all links L is revealed to all nodes only after the final round. The ratio of true links over fake links in Bloom Filter scheme is almost identical to that of Baseline. The reason lies in the independence of all links in exchange protocols. All links have the same probability to be sampled and sent to neighbors of nodes. Interestingly, Bloom Filter helps reduce the true/fake link ratio faster than Baseline for small α (Section 6.5.1) thanks to its inherent false positives as well as false negatives caused by bit erasure.

6.5 Evaluation

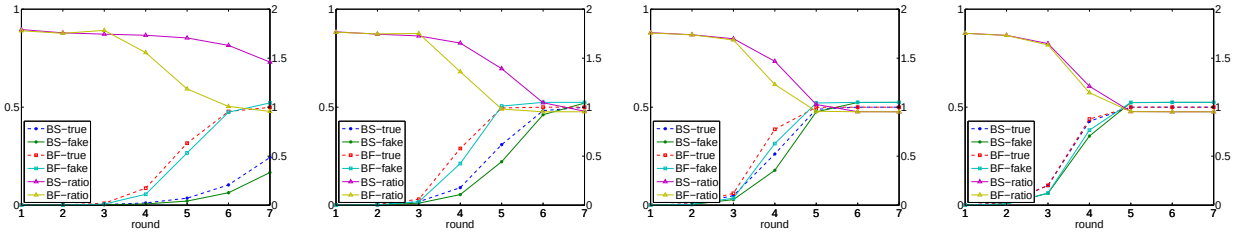
In this section, we empirically evaluate the performance of our proposed schemes on synthetic graphs. All algorithms are implemented in Java and run on a desktop PC with *Intel*® Core i7-4770@ 3.4Ghz, 16GB memory.

Two kinds of synthetic graphs are generated: Barabási-Albert power-law (PL) graphs and Erdös-Rényi (ER) random graphs [73]. Table 6.2 lists six synthetic graphs used in our experiments. Each test case is run 10 times. We abbreviate the two schemes Baseline (BS) and BloomFilter-based (BF) in the legends of the figures below.

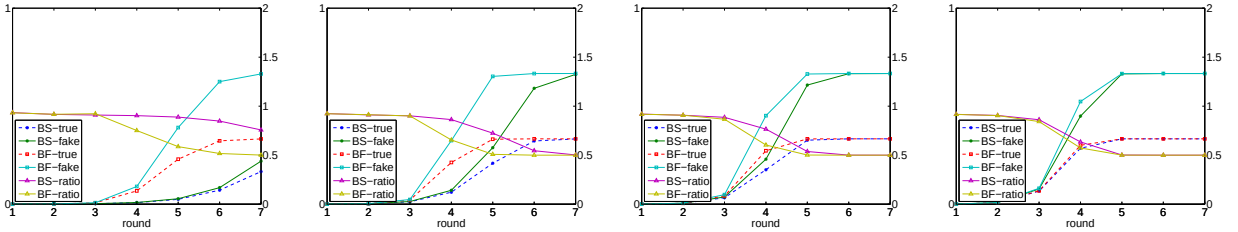
We choose $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$ and $\beta \in \{0.5, 1.0\}$. The default number of hash functions k is 4.

Table 6.2: Synthetic graphs

Graph	#Nodes	#links	Diameter
PL1	10,000	29,990	7
PL2	**10,000**	**49,970**	**6**
PL3	10,000	99,872	5
ER1	10,000	30,076	10
ER2	**10,000**	**50,424**	**7**
ER3	10,000	99,615	5



(a) $\alpha = 0.25, \beta = 0.5$ (b) $\alpha = 0.5, \beta = 0.5$ (c) $\alpha = 0.75, \beta = 0.5$ (d) $\alpha = 1.0, \beta = 0.5$



(e) $\alpha = 0.25, \beta = 1.0$ (f) $\alpha = 0.5, \beta = 1.0$ (g) $\alpha = 0.75, \beta = 1.0$ (h) $\alpha = 1.0, \beta = 1.0$

Figure 6.8: Normalized number of true/fake links (e.g. *BS-true* is the number of true links in Baseline scheme) and link ratios on ER2.

6.5.1 Message Volume and Inference Attacks

We investigate the message volume by the total number of true/fake links at all nodes after each round $t = 1..Diam(G)$. These values are normalized by dividing them by $N.M.(1 + 2\beta)$. We also estimate the inference attacks by the ratio between the number of true links and the number of fake links. Fig. 6.8 and Fig. 6.9 show two-y-axis charts. The left y-axis is for the normalized number of links. The right y-axis is for the ratios.

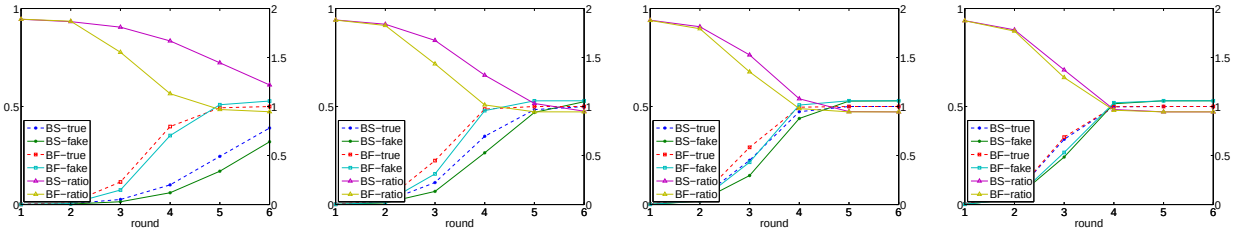
Several observations can be made clearly from Figures 6.8 and 6.9. First, the number of true/fake links increases exponentially and converges fast as all nodes reach the round at $Diam(G)$. For $\alpha = 0.25$, Baseline does not converge because not all links are propagated to all nodes. Bloom filter scheme produces higher number of true/fake links, especially at $\alpha = 0.25, 0.5$. For larger values of α, the two schemes almost coincide. Second, the ratio of true links over fake links decreases round by round and converges to $\frac{1}{2\beta}$. In early rounds, the ratios are lower than $\frac{1}{\beta}$. Higher the ratio, higher inference risk of true links. Clearly, Bloom Filter scheme reduces the risk better than Baseline for $\alpha = 0.25, 0.5$ in later rounds.

Fig. 6.10 displays the distribution of link volume collected at sample nodes. We sort V by degree and take 100 sample nodes. ER graphs which are commonly called *homogeneous* graphs show nearly uniform distributions for various values of (α, β). On the contrary, PL graphs are *heterogeneous* ones and sample nodes exhibit much more random distributions.

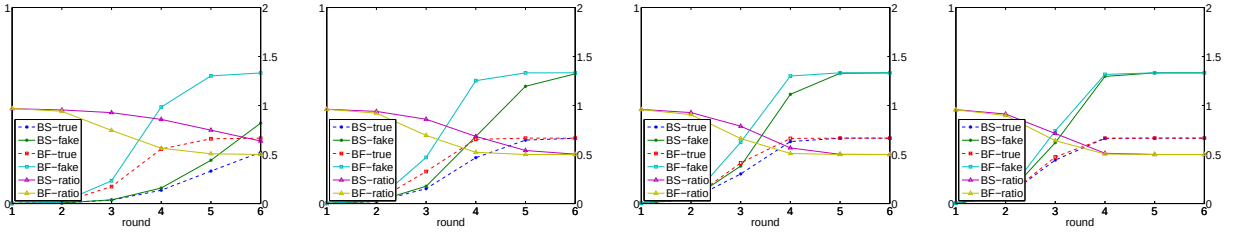
The inference attack on Baseline scheme (Section 6.3.4) is shown in Fig. 6.11. The average F1 scores for two values of β are plotted at different rounds of Baseline protocol. We observe that the scores are quite close to the theoretical values $1/(1+\beta)$ (see the dashed lines). On ER2 graph, the inference attack is more effective at latter rounds and for larger α while this is not clear on PL2.

6.5.2 Bloom Filter Scheme

In this section, we examine the performance of Bloom Filter scheme. We set the false positive rate of Bloom Filter to 0.1, 0.01 and 0.001 (the number of hash functions k is 4,7 and 10

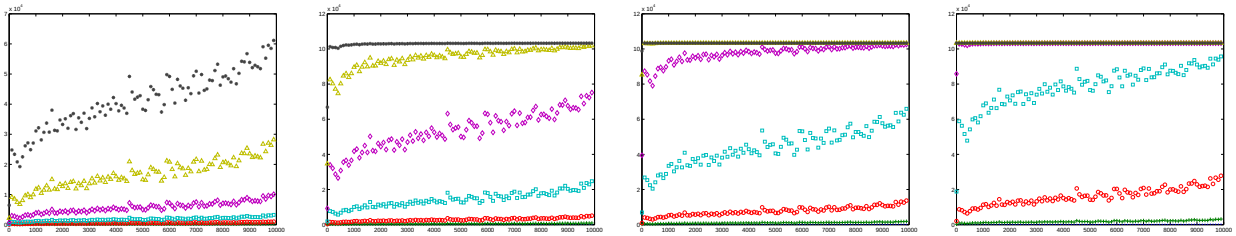


(a) $\alpha = 0.25, \beta = 0.5$ (b) $\alpha = 0.5, \beta = 0.5$ (c) $\alpha = 0.75, \beta = 0.5$ (d) $\alpha = 1.0, \beta = 0.5$

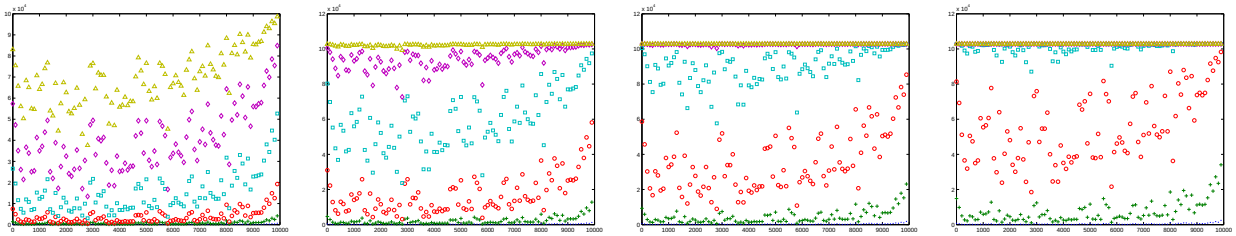


(e) $\alpha = 0.25, \beta = 1.0$ (f) $\alpha = 0.5, \beta = 1.0$ (g) $\alpha = 0.75, \beta = 1.0$ (h) $\alpha = 1.0, \beta = 1.0$

Figure 6.9: Normalized number of true/fake links (e.g. *BS-true* is the number of true links in Baseline scheme) and link ratios on PL2.

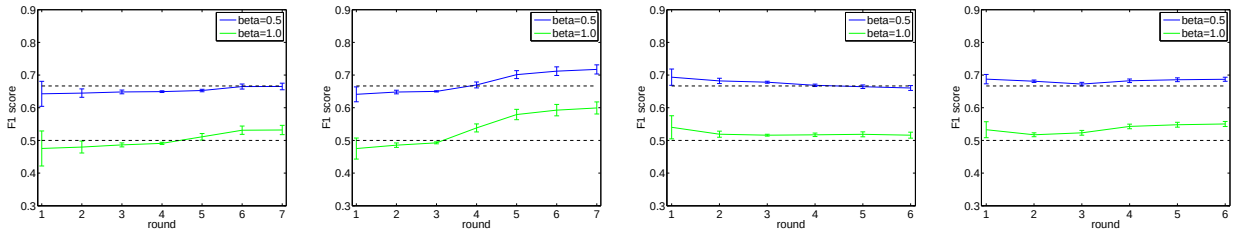


(a) $\alpha = 0.25, \beta = 0.5$ (b) $\alpha = 0.5, \beta = 0.5$ (c) $\alpha = 0.75, \beta = 0.5$ (d) $\alpha = 1.0, \beta = 0.5$



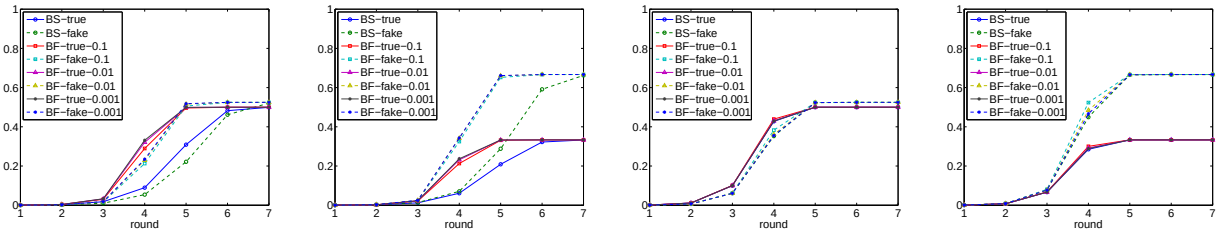
(e) $\alpha = 0.25, \beta = 0.5$ (f) $\alpha = 0.5, \beta = 0.5$ (g) $\alpha = 0.75, \beta = 0.5$ (h) $\alpha = 1.0, \beta = 0.5$

Figure 6.10: Number of edges at sampled nodes ($t = 1(.), t = 2(+), t = 3(\circ), t = 4(\square), t = 5(\diamond), t = 6(\triangle), t = 7(*)$). First row for ER2, second row for PL2.

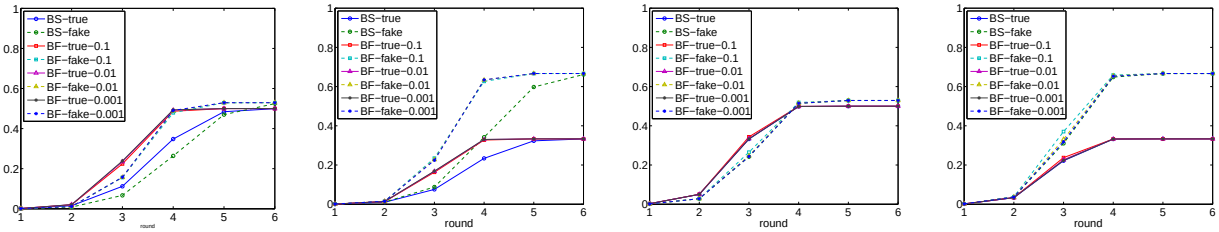


(a) ER2, $\alpha = 0.5$ (b) ER2, $\alpha = 1.0$ (c) PL2, $\alpha = 0.5$ (d) PL2, $\alpha = 1.0$

Figure 6.11: Inference attacks



(a) $\alpha = 0.5, \beta = 0.5$ (b) $\alpha = 0.5, \beta = 1.0$ (c) $\alpha = 1.0, \beta = 0.5$ (d) $\alpha = 1.0, \beta = 1.0$



(e) $\alpha = 0.5, \beta = 0.5$ (f) $\alpha = 0.5, \beta = 1.0$ (g) $\alpha = 1.0, \beta = 0.5$ (h) $\alpha = 1.0, \beta = 1.0$

Figure 6.12: Normalized number of true/fake links by different false positive rates. First row for ER2, second row for PL2.

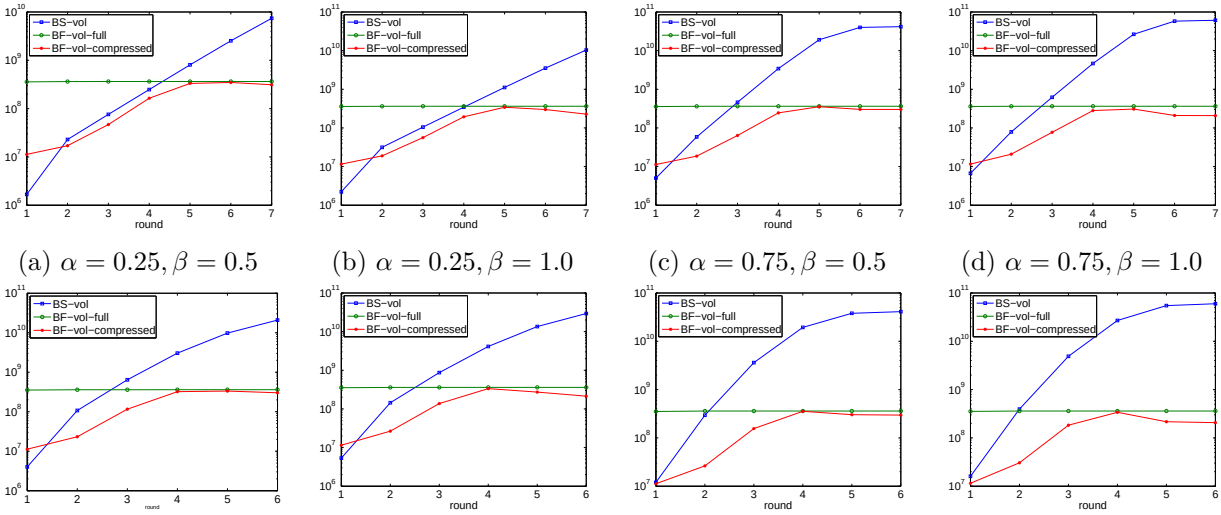
respectively). Fig. 6.12 displays the normalized number of true/fake links by Baseline and Bloom Filter with different false positive rates. We find that lower false positive rates make no improvement for $\alpha = 0.25, 0.5$. Bit Erasure (Algorithm 18) causes this effect. Lower α means more bits to be erased in Bloom filters. Consequently, the number of false positive links and false negative links is amplified round by round for small α.

We compare the communication complexity of Baseline and Bloom Filter schemes in the number of bytes transmitted among nodes in each round. Fig. 6.13 reports the total message size of Baseline and Bloom Filter (with or without compression). Baseline scheme stores links in clear form, so it incurs exponential communication complexity. As discussed in Section 6.4.1, we assume that each node ID cost 4 bytes and a link list of length l may be stored compactly in $4l$ bytes. Bloom Filter uses constant-sized bit arrays, so its communication cost is constant too. Using bit array compression (Section 6.4.2), Bloom Filter scheme reduces the message size even further, especially at early rounds.

In Fig. 6.14, we compare the runtime of Baseline and Bloom Filter simulations in a single PC. In each round, each node updates its link set (`count` operation) by aggregating noisy link lists from its neighbors. Then, each node prepares (`exchange` operation), for the next round, new noisy link lists sampled from its link set. At $\alpha < 1$, the exchange operations cost an increasing time as more rounds are considered. Higher α makes the link sampling slower. Only at $\alpha = 1$, we have fast exchange operations. In particular, the exchange runtime of Bloom Filter scheme is constant for $\alpha = 1$ and is an increasing function of round for $\alpha < 1$ due to bit erasure operations. The count operation of Bloom Filter dominates that of Baseline because each node has to hash the full link set to recover its noisy link set at each round.

6.5.3 Utility-Oriented Initialization

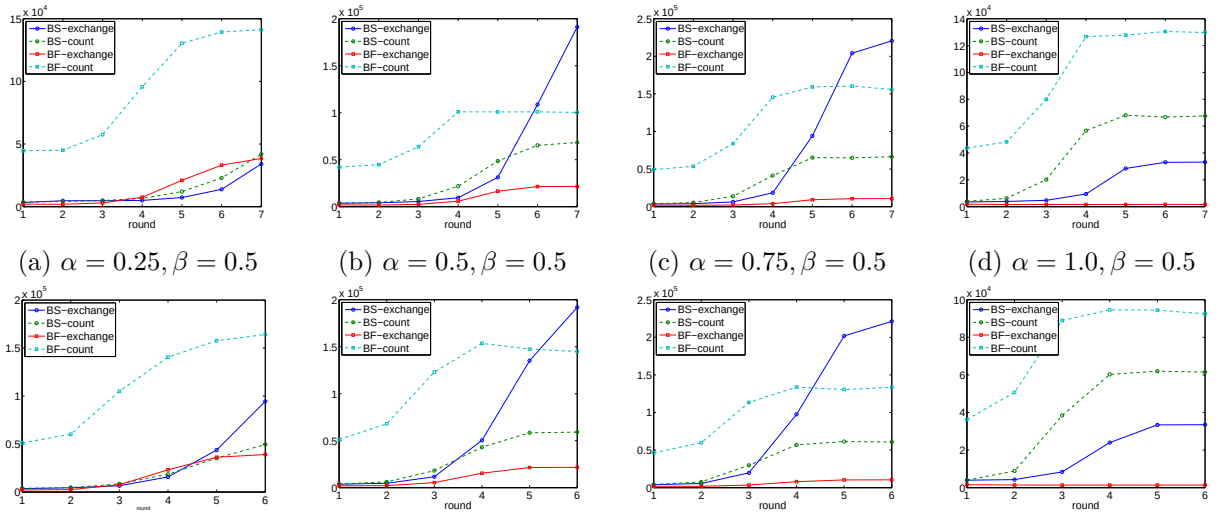
In this section, we illustrate the benefit of two-round initialization (Algorithm 16). We set $\gamma = 0.0, 0.5$ and denote the enhanced scheme as *D2*. Several utility metrics are chosen as follows.



(a) $\alpha = 0.25, \beta = 0.5$ (b) $\alpha = 0.25, \beta = 1.0$ (c) $\alpha = 0.75, \beta = 0.5$ (d) $\alpha = 0.75, \beta = 1.0$

(e) $\alpha = 0.25, \beta = 0.5$ (f) $\alpha = 0.25, \beta = 1.0$ (g) $\alpha = 0.75, \beta = 0.5$ (h) $\alpha = 0.75, \beta = 1.0$

Figure 6.13: Communication complexity. Y-axis is the number of bytes transmitted among nodes (log-scale). First row for ER2, second row for PL2.



(a) $\alpha = 0.25, \beta = 0.5$ (b) $\alpha = 0.5, \beta = 0.5$ (c) $\alpha = 0.75, \beta = 0.5$ (d) $\alpha = 1.0, \beta = 0.5$

(e) $\alpha = 0.25, \beta = 0.5$ (f) $\alpha = 0.5, \beta = 0.5$ (g) $\alpha = 0.75, \beta = 0.5$ (h) $\alpha = 1.0, \beta = 0.5$

Figure 6.14: Total simulation runtime of all nodes (in millisecond). First row for ER2, second row for PL2.

- Power-law exponent of degree sequence: PL is the estimate of η assuming the degree sequence follows a power-law $n_d \sim d^{-\eta}$ where n_d is the number of d-degree nodes.

- Clustering coefficient: $CC = \frac{3N_\Delta}{N_3}$ where N_Δ is the number of triangles and N_3 is the number of connected triples.

- Average distance: APD is the average distance among all pairs of vertices that are path-connected.

- Distance distribution: $Distance$ is the normalized node-pair shortest-path histogram.

We take 100 sample nodes by degree and compare local aggregated graphs to the ground truth. The ground truth is computed by setting $\beta = 0$ in Baseline scheme. Fig. 6.15 shows the benefit of two-round initialization (D2-0.0 and D2-0.5) on PL2 graph in early rounds. D2-0.0 and D2-0.5 schemes result in lower relative errors than Baseline and Bloom Filter in the first and second rounds, especially by CC and PL metrics. All schemes are comparable at $t = 3$, except on CC metric. Finally, Baseline and Bloom Filter are almost equivalent in terms of utility and they perform better D2 schemes at $t = Diam(G)$ on PL, APD and $Distance$ metrics.

6.6 Conclusion

We motivate the private link exchange problem as an alternative to social graph crawling and centralized anonymization of data. The problem is distributed and provides a privacy/utility tradeoff for all nodes. Our proposed problem is unique in the sense that the disseminated data over the links are the links themselves. We present two schemes for (α, β)-exchange protocol: Baseline and Bloom filter based. Experiments on synthetic graphs clarify advantages and drawbacks of both schemes. Baseline scheme keeps link lists in clear form, so its communication cost increases fast. Bloom Filter scheme incurs lower communication complexity but needs an extra recovery step in the final round. Both schemes guarantee link privacy in the range $[\frac{1}{2\beta}, \frac{1}{\beta}]$. In Baseline, the inference attack based on link counting is not much better than the random attack. For future work, we plan to investigate asynchronous models and node/links failures. We also consider community-based link exchange models in which nodes are gathered in super nodes and the link exchange takes place among super nodes only.

In the next chapter, we summarize our main contributions and discuss possible directions for future work.

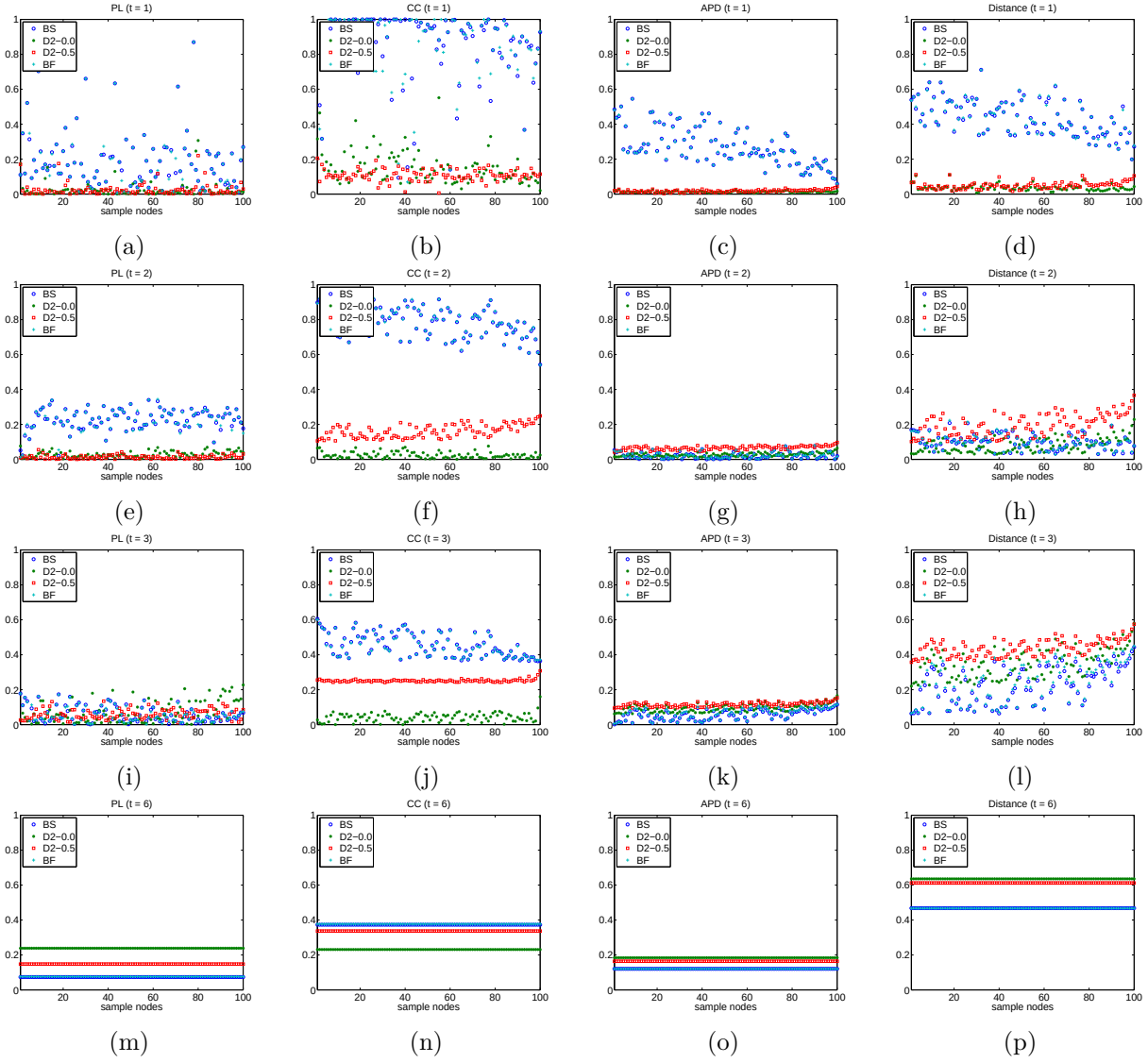


Figure 6.15: Utility relative errors on PL2 ($\alpha = 1.0, \beta = 0.5$)

Chapter 7

Conclusion and Perspectives

Contents

We conclude this thesis by summarizing the main contributions and describing possible directions for future research.

7.1 Conclusion

In this thesis, we have made significant contributions to the following problems for privacy protection in social networks.

Social Graph Anonymization Social graphs are the key underlying structure of social networking services. They are relational data in which a lot of correlation information about users is encoded. Careless publication of social graphs opens the doors to large-scale privacy violations. Graph anonymization, therefore, is an important technique to mitigate the risks. This thesis contributes to the graph anonymization toolbox with several novel schemes based on uncertainty semantics and differential privacy, two out of five main anonymization categories. in Chapter 3, we identify the shortcomings of two existing works [11, 66] and devise a unified model, Uncertain Adjacency Matrix (UAM). Based on UAM, we propose an effective scheme, Maximum Variance that formalizes the edge probability assignments in the form of a quadratic program. UAM also helps us to quantify and compare the total edge variance between the three schemes and to show how well our approach fills the existing gap between [11] and [66].

Differential privacy is a formal privacy notion that has attracted a lot of interdisciplinary research recently. Differentially private mechanisms for graph anonymization are broadly studied in Chapter 4. We identify two main classes of techniques: direct publication and model-based publication. We also analyze two radical challenges of graph release under differential privacy: huge output space and consistency. These challenges allow a relaxation of ϵ to $\ln n$ where n is the number of nodes. Our proposed scheme Top-m-Filter and an existing one EdgeFlip belong to the category of direct publication. Both of them satisfy the consistency property while the model-based schemes like 1K-series [100], HRG [105] and DER [21] cannot. Through evaluation over a wide range of real graphs, we reveal the advantages and drawbacks of each technique. Direct publication schemes require large ϵ (i.e. $\approx \ln n$) to achieve low relative errors for all metrics. Meanwhile, the results of model-based schemes depend strongly on graphs and utility

metrics. 1K-series works well for degree-based metrics while HRG is suitable for path-based metrics on graphs exposing community structure. DER performs worst and is runnable only on medium-sized graphs.

Private Community Detection Community structure is the mesoscopic structure between microscopic (node level) and macroscopic (graph level) structures. A graph is said to have community structure if it appears as a combination of components fairly independent of each other. The problem of how to identify communities plays a significant role in understanding the organization and function of complex networks. A large body of work on community detection in a variety of settings has been proposed over the last decade. However, these approaches work, by default, in non-private manner. In this thesis, we add the privacy dimension to the community detection problem because non-private output clusterings may reveal sensitive information about users' relationships and memberships. We analyze the major challenges of community detection under differential privacy. We explain why techniques borrowed from k-Means [94] fail and how the difficulty of ϵ-DP recommender systems [43] justifies a relaxation of ϵ to $0.5 \ln n$. We propose two novel schemes LouvainDP and ModDivisive. As a bottom-up approach, LouvainDP employs the high-pass filtering technique from [26] and Louvain method. ModDivisive is a top-down approach and uses the modularity-based score function in the exponential mechanism. Both schemes run in linear time and output much better clusterings than the existing techniques.

Private Link Exchange Despite their high value for research on information societies, social graphs are not published in clear-form due to serious privacy concerns. Instead, they are anonymized in various forms and published to third party consumers. Alternatively, social networking sites provide APIs for data crawlers at limited rates and within privacy constraints. To overcome the constraints set by the service providers, we propose an aggregation model start from user perspective, i.e. from the contributors of OSNs. We introduce a novel private link exchange problem as an alternative to social graph crawling and centralized anonymization of data. The problem is distributed and provides a privacy/utility tradeoff for all nodes. Our proposed problem is unique in the sense that the disseminated data over the links are the links themselves. We design two schemes for (α, β)-exchange protocol: Baseline and Bloom filter based. We protect the true links by adding fake links and requiring the propagation probability of links to be attenuated by distance. We analyze the advantages and drawbacks of each scheme. The evaluation on various synthetic graph models helps us to draw a number of critical findings. We believe that (α, β)-link exchange provides a good prototype for important extensions and real-world deployments in future.

7.2 Perspectives

As discussed in Chapter 3, our work on UAM and MarVar has several limitations that may incite several directions for future research. First, the model of UAM is currently constructed by solving a quadratic program. The exact optimization over millions of variables is non-trivial and we have to divide the graph into subgraphs of tens of thousands of nodes. Approximations based on linear programming and local search may be good solutions. In addition, we believe that each edge has only limited influence to the probabilities of nearby edges. This locality property may further simplify the approximate methods by dividing the graph to nearly disjoint communities. Second, the linear tradeoff for both $H1$ and $\sqrt{H2_{open}}$ against the relative errors is worth a theoretical explanation. Third, the extensions of UAM and MaxVar for directed and bipartite graphs are straightforward and we intend to find competitors for them and carry out extensive evaluations on real graphs.

Chapters 4 and 5 apply differential privacy to graph release and community detection. Both of them require a relaxation of ϵ to $\ln n$ and $0.5 \ln n$ respectively. As we have seen in Chapter 4, the consistency property stipulates that relative errors have to be as low as zero for large enough privacy budget. Only direct publication schemes like TmF and EdgeFlip satisfy this desideratum. All well-known model-based schemes like 1K-series [100], HRG [105] and DER [21] fail to meet this requirement. Specifically, model-based schemes cannot keep low relative errors for both degree-based metrics and path-based metrics. We think that the main reason lies in the reconstruction step used in the model-based schemes. For future work, we aim at formally quantifying these types of reconstruction error. On the other hand, finding alternative summary structures (e.g. structures of space complexity $O(n \log n)$) for graphs may lead to publication models better than $O(n)$ models like HRG, 1K-series.

Regarding ϵ-DP community detection, LouvainDP's performance depends on the initialization of supernodes. Remember that Louvain method [9], by using only local moves for searching the best community for each node, implies that supernodes are formed by connected nodes. This characteristic does not hold for LouvainDP due to its random division of the node set into groups of size k (to cost no privacy budget). An improvement for LouvainDP may stem from the good private initialization of supernodes. We may reserve a part of ϵ for this step. As a top-down approach, ModDivisive has to split the privacy budget ϵ for MCMC runs at levels of the k-ary tree. It has a limitation: the best cuts are near the root. In addition, the MCMC only simulates the exponential mechanism and its convergence is hard to prove. This is also the problem of HRG-based schemes in Chapter 4. As we discussed in Section 5.2.2, the output space of ϵ-DP community detection is immense ($O(n^n)$) and non-metric, so explicit implementations of the exponential mechanism are infeasible. The same argument holds for HRG-based schemes which have to explore the space of size $O(2^{n(n-1)/2})$. To avoid this approximation, we may think about other query-and-reconstruction schemes. For example, we may sample m random clusterings and query for their modularity scores. This operation costs $\epsilon = m.\frac{3}{m} = 3$. In the end, we keep top clusterings and fuse them to get better clusterings.

Despite its simplicity, the link exchange model in Chapter 6 provides a good starting point for several important extensions and real-world deployments. The current model assumes synchronous message passing, no node/edge failures and honest-but-curious users. First, if we switch to asynchronous models, i.e. we remove the requirement that each node waits for all t-round messages from its neighbors, the problem becomes more challenging. However, the semantics of multipath propagation for $\alpha < 1$ (Section 6.3) is still the same. Each edge (v, w) once reaches node u is forwarded to the neighbors of u with probability α. Second, node/edge failures may change significantly the distribution of final link sets at nodes, especially if these failures happen at bridges or articulation nodes. Also, when high-degree nodes (or hubs) refuse to join the protocol, a large number of propagation paths are removed, so the chance to get distant links decreases too. Third, if dishonest nodes exist in the network and form coalitions to reveal true links out of link sets, the protocol is even harder to design. We may need costly verification mechanisms in this case. Lastly, to reduce the communication cost, we may think about community-based link exchange models in which nodes are gathered in super nodes and the link exchange takes place among super nodes only.

Appendix A

Further Discussion

Contents

A.1 Chapter 4: Analysis of Expected Edit Distance

In this section, we give a deeper analysis of the expected edit distance $D(G, \tilde{G})$ and the effect of ϵ_2 on it. From Section 4.3.3,

$$D(G, \tilde{G}) = \frac{1}{2}(|E_G \setminus E_{\tilde{G}}| + |E_{\tilde{G}} \setminus E_G|) = \frac{1}{2}(m - n_1 + \tilde{m} - n_1) \tag{A.1}$$

Taking the expectation, we get $E[D(G, \tilde{G})] = m - E[n_1]$ where $n_1 = \frac{\tilde{m}}{2}(2 - e^{-\epsilon_1(1-\theta)})$ and $\theta = \frac{1}{2\epsilon_1}\ln(\frac{n(n-1)}{2\tilde{m}} - 1) + \frac{1}{2}$. Compared to Formula 4.3, the term m is replaced by $\tilde{m}$ to take into account the effect of ϵ_2.

After a few calculations, we get $E[n_1] = m - E[X]$ where

$$X = \frac{1}{2}e^{-\epsilon_1/2}\sqrt{\frac{\tilde{m}n(n-1)}{2} - \tilde{m}^2} \tag{A.2}$$

Because $\tilde{m}$ may take non-positive values (with vanishing probabilities) which make X undefined, we bound the value of $E[X]$ in the interval $[m - \Delta, m + \Delta]$ where Δ indicates the approximation of Laplace distribution as in Fig. A.1. For $\epsilon_2 = 0.1$, the probability mass of Laplace distribution in the interval $[m - \Delta, m + \Delta]$ is 0.9999 at $\Delta = 100(\ll m)$.

By Formula 4.3, the expected of n_1 is $m - X_0$ where

$$X_0 = \frac{1}{2}e^{-\epsilon_1/2}\sqrt{\frac{mn(n-1)}{2} - m^2} \tag{A.3}$$

Our objective is to show that with high probability (w.h.p) $X \in [X_0\sqrt{(1 - \Delta/m)}, X_0\sqrt{(1 + \Delta/m)}]$.

It is straightforward to see that X is an increasing function in the interval $[m - \Delta, m + \Delta]$, so $E[X]$ is lower bounded by $\frac{1}{2}e^{-\epsilon_1/2}\sqrt{\frac{(m-\Delta)n(n-1)}{2} - (m - \Delta)^2}$ and upper bounded by $\frac{1}{2}e^{-\epsilon_1/2}\sqrt{\frac{(m+\Delta)n(n-1)}{2} - (m + \Delta)^2}$. Using the facts that $\Delta \ll m$ and $m \ll n(n-1)/2$, we get the result: $X \in [X_0\sqrt{(1 - \Delta/m)}, X_0\sqrt{(1 + \Delta/m)}]$ (w.h.p) after a few calculations.

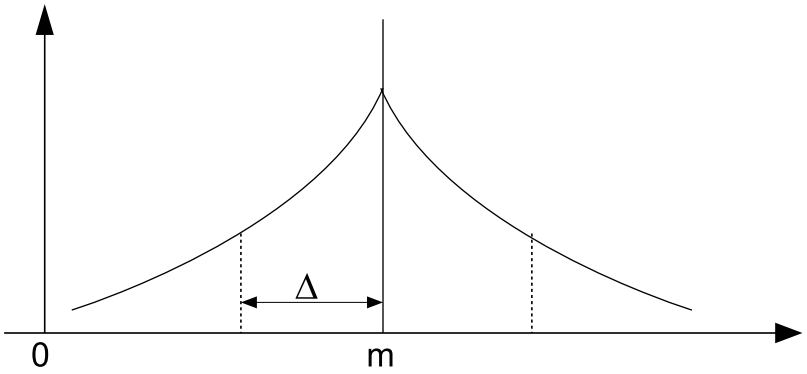


Figure A.1: Approximate distribution of $\tilde{m}$ by the interval $[m - \Delta, m + \Delta]$

As mentioned in Section 4.3.2, the High-Filtering technique by Cormode et al. [26] uses the number of non-zero entries in *non-private* manner. Their threshold θ is computed as

$$\theta = \frac{\log\left(\frac{(1+\alpha)s}{2(m_0 - m_1)}\right)}{\log \alpha} \tag{A.4}$$

where m_0 is the domain size (public), m_1 is the number of non-zero entries (private), $\alpha = e^{-\epsilon}$ is the parameter of Geometric noise and $s = \Theta(m_1)$ is the expected number of passing entries. We can hide the true value of m_1 in this formula by $\tilde{m_1} = m_1 + Lap(1/\epsilon_2)$ where ϵ_2 is subtracted from ϵ as we have done in TmF.

Appendix B

Résumé étendu

Contents

B.1 Motivation

B.1.1 Vie Privée dans les Réseaux Sociaux

Un réseau social est une structure sociale composée d'un ensemble d'acteurs sociaux, de relations et autres modes d'interactions sociales entre les acteurs. Avec l'émergence des réseaux sociaux en ligne (OSN) [15], les usagers disposent de médias sociaux d'une puissance inédite. Facebook [16], le plus grand OSN, a plus de 1,65 milliard d'utilisateurs mensuels actifs avec une augmentation de 15 pour cent par an. La plupart des OSNs offrent des services gratuits en échange d'informations collectées sur l'utilisateur et qui sont exploitées par les fournisseurs de services pour de la publicité ciblée. Parce que la protection des données de l'utilisateur n'est pas toujours garantie, la vie privée des utilisateurs du réseau social est souvent mise en danger. La fuite d'informations peut avoir de nombreuses causes: la négligence des utilisateurs, le partage de données à des tiers par les fournisseurs de services et les attaques brutales par les cybercriminels [17].

[15] https://en.wikipedia.org/wiki/Social_networking_service
[16] https://zephoria.com/top-15-valuable-facebook-statistics/
[17] https://heimdalsecurity.com/blog/10-surprising-cyber-security-facts-that-may-affect-your-online-safety/

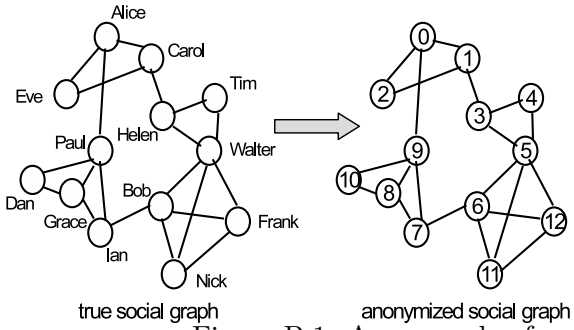
Les OSNs sont également une source importante de données pour le Big Data, une technologie qui est en train de changer le monde. Les réseaux sociaux, ainsi que d'autres réseaux complexes, fournissent des données utiles pour les études en sciences sociales, statistiques et théorie des graphes. Aujourd'hui, le public s'intéresse à la façon dont les acteurs économiques peuvent maximiser les avantages de l'exploitation des masses de données tout en minimisant les risques sur la vie privée. Les utilisateurs d'OSNs, de leur côté, souhaitent toujours plus d'avantages que de risques. Ceci explique le développement rapide des politiques de confidentialité de plus en plus complexes par les fournisseurs de services (comme Facebook). Dans la section suivante, nous motivons quelques approches de minimisation des risques sur les réseaux sociaux en fonction de leurs graphes sociaux.

B.1.2 Menaces sur la Vie Privée liées aux Publications des Graphes Sociaux

Li et al. [54] présentent les deux types principaux de divulgation d'information étudiés dans la littérature: la *divulgation de l'identité* et la *révélation d'un attribut*. Notons que la découverte de l'identité conduit souvent à la révélation d'un attribut. Chaque fois qu'un attaquant révèle le lien de certaines données à une entité spécifique du monde réel, nous disons qu'une révélation d'identité a lieu. La révélation d'un attribut implique une inférence réussie d'attributs sensibles appartenant à un utilisateur ciblé. En outre, la littérature sur la vie privée identifie deux classes principales de mécanismes de requêtes à la disposition d'un attaquant: le mode *interactif* et le mode *non-interactif*. Dans le cadre interactif, un attaquant est autorisé à poser des questions à une base de données et le propriétaire de la base de données répond avec des réponses bruitées. Sur la base des informations recueillies à partir des réponses précédentes, l'attaquant peut poser des requêtes *adaptatives* pour éviter des réponses inutiles. Dans un cadre non-interactif, le propriétaire de la base de données publie une version anonymisée de la base de données pour répondre à certaines exigences de la vie privée (par exemple les noms d'utilisateur dans la base de données publiée peuvent être remplacés par des numéros fictifs). Les recherches antérieures sur la confidentialité des données sont principalement axées sur des données structurées en *tables* dont les lignes représentent des enregistrements indépendants et identiquement distribuées (i.i.d) et dont les colonnes représentent les attributs [54,59,95]. Cependant, les données du monde réel sont souvent *relationnelles* et les enregistrements d'une table sont liés les uns aux autres ou à ceux provenant d'autres tables. Cela soulève des défis considérables pour la préservation de la vie privée des utilisateurs.

Chaque utilisateur des réseaux sociaux en ligne est représenté par un profil riche d'un ensemble d'attributs (par exemple le sexe, la date de naissance, les loisirs, l'état marital et le lieu) et par ses relations (les liens d'amitié et les adhésions à des groupes d'intérêt). En particulier, les graphes sociaux qui représentent la structure des réseaux sociaux sont des données relationnelles qui contiennent beaucoup d'informations et peuvent être exploitées par des analystes de données, ainsi que par des attaquants. En général, les utilisateurs d'OSNs ont le sentiment que les fournisseurs de services gardent leurs informations privées de manière sécurisée [4] et que leurs identités sont dissimulées parmi celles des autres utilisateurs dans des bases de données anonymisées. Cependant, une forte corrélation entre les utilisateurs dans une même base de données et avec des utilisateurs d'autres bases de données rend les attaques par ré-identification à grande échelle possibles [70–72,111].

Nous illustrons une attaque simple par ré-identification dans Figure B.1. Un graphe social de treize utilisateurs est naïvement anonymisé en remplaçant les noms d'utilisateurs par des numéros fictifs. Un attaquant peut réidentifier les utilisateurs en comptant le nombre d'amis dont chaque utilisateur dispose. Ceci est tout à fait possible parce que les OSNs comme Facebook



Degree	Users
2	Eve, Dan, Tim
3	Alice, Carol, Helen, Frank, Nick, Ian
4	Bob, Paul, Grace
5	Walter

Figure B.1: An example of re-identification attack

permettent aux utilisateurs de laisser leur nombre d'amis en mode public. En supposant que l'attaquant tente de lier le nœud 5 dans le graphe anonymisé à un utilisateur dans le vrai graphe social, il itère sur l'OSN des recherches d' utilisateurs ayant cinq amis. Ainsi, l'utilisateur Walter sera révélé. Puis des informations sur Walter peuvent être obtenues par d'autres attaques utilisant des inférences, par exemple en examinant les groupes que Walter rejoint [111].

B.1.3 Nouveaux Mécanismes pour Préserver la Vie Privée dans les Graphes Sociaux

Avec l'émergence de réseaux de plus en plus complexes [73], la communauté scientifique a besoin de graphes de grandes tailles précis pour mener des études approfondies. Toutefois, cette exigence est souvent en conflit avec la confidentialité des données des entités contribuantes. Les approches naïves comme la suppression des IDs de l'utilisateur à partir d'un graphe social ne sont pas efficaces, laissant les utilisateurs sujets à des risques sur leur vie privée, en particulier à des attaques par ré-identification [4, 46]. Par conséquent, de nombreuses techniques élaborées d'anonymisation ont été proposées [23, 55, 96, 103, 112, 113].

Étant donné un graphe non-orienté non-étiqueté, les méthodes d'anonymisation existantes se répartissent en cinq catégories principales.

- La première utilise l'ajout *aléatoire*, la suppression et la commutation de liens [13, 46, 107, 109] pour empêcher la ré-identification des nœuds ou des liens.

- La seconde vise à obtenir la k-anonymité [95] par additions ou suppressions *déterministes* de liens [23, 55, 96, 103, 112, 113], en supposant que l'attaquant connait certaines propriétés de ses nœuds ciblés.

- La troisième repose sur la *généralisation* [18, 46, 97] et regroupe les nœuds dans des super-nœuds de taille au moins k, où k est un paramètre de confidentialité.

- Les méthodes de la quatrième catégorie attribuent des probabilités aux liens pour ajouter de l'incertitude au graphe original. Les probabilités peuvent être calculées explicitement comme dans [11] ou implicitement par Random-Walk [66]. Notez que cette classe de systèmes et ceux de la troisième catégorie induisent des modèles de *mondes possibles*, c'est-à-dire, les graphes produits par la méthode sont obtenus par échantillonnage du graphe incertain.

- Enfin, il existe des techniques pour l'anonymisation de graphes basées sur la notion de *vie privée différentielle* [33].

Nous observons quelques problèmes des quatrième et cinquième catégories dans l'état-de-l'art. La quatrième catégorie exploite la sémantique de liens probabilistes pour injecter de l'incertitude dans un graphe déterministe en le transformant en un graphe probabiliste avant de publier des graphes échantillonnés. L'approche par *(k,ϵ)-obscurcissement* [11] n'offre pas un bon compromis entre la protection de la vie privée et l'utilité, tandis que l'approche *RandWalk* [66] souffre de limites inférieures pour l'utilité, malgré son excellent compromis entre vie privée et utilité.

La cinquième catégorie aborde la publication du graphe par la technique de vie privée différentielle (ϵ-DP). Pour une introduction brève, voir la section B.4. Par la vie privée différentielle, nous voulons assurer que les liens entre les utilisateurs restent cachés dans le graphe publié tout en conservant une information structurelle importante pour permettre l'analyse du graphe [21,89,100,101,105]. Cependant, le problème est difficile en raison de l'énorme espace de graphes bruités possibles en sortie. La plupart des systèmes existants ne permet pas de relâcher le paramètre, appelé budget, qui contrôle la vie privée, ni de déterminer sa borne supérieure. En outre, certains d'entre eux ont un problème de flexibilité, et généralement une complexité quadratique.

En dehors de la publication du graphe social, d'autres opérations sur les graphes ont également un grand intérêt, comme par exemple la *détection de communautés*. De nombreux réseaux complexes exposent une structure *mésoscopique*, c'est-à-dire qu'ils apparaissent comme une combinaison de composants relativement indépendants les uns des autres. Ces composants sont appelés *communautés*, *modules* ou *clusters* et c'est un problème important de les découvrir pour la compréhension de l'organisation et le fonctionnement des réseaux complexes. Au cours de la dernière décennie, un grand nombre d'algorithmes de détection de communautés (CD) ont été proposés pour résoudre ce problème dans une variété de contextes, tels que les réseaux non-orientés ou orientés, avec chevauchement ou non, pour des graphes pondérés ou non [38]. Ces approches sont traitées d'une manière non privée, c'est-à-dire un collecteur de données (comme Facebook) connaît tous les utilisateurs contribuants et leurs relations avant d'exécuter des algorithmes de CD. La sortie d'un tel CD, sous la forme la plus simple, est un regroupement de nœuds. Même dans ce cas, lorsque un seul des clusters de nœuds (et non tout le graphe) est révélé, la vie privée des utilisateurs peut encore être mise en danger.

Nous formulons comme dernière contribution un problème original et intéressant: l'échange de liens privés. Le problème est motivé par le fait que la plupart des OSNs gardent leurs données secrètes et de manière centralisée. Traditionnellement, les données bruitées sont publiées pour la recherche publique et des données plus précises sont réservées à la recherche interne. Alternativement, les chercheurs sont autorisés à collecter les graphes sociaux (et les données), mais avec de fortes limitations, ce qui aboutit à des vues partielles des vrais graphes sociaux. Pour surmonter cet obstacle, nous pouvons nous placer du point de vue des utilisateurs, les contributeurs des OSNs. Plus précisément, si les utilisateurs collaborent avec précaution ils peuvent échanger des listes d'amis bruitées avec leurs voisins en plusieurs tours pour obtenir une meilleure vue du vrai graphe social. Nous partons de l'hypothèse qu'un utilisateur fait plus confiance à ses amis qu'à des étrangers. Les utilisateurs veulent donc que les informations sur leur liste d'amis soient réduites lorsque la distance de propagation dans le graphe social augmente. Cependant, des questions fondamentales demeurent sur la faisabilité de ce modèle. La première question est de savoir comment définir des concepts de confidentialité simples et efficaces pour les processus d'échange de lien. La deuxième question est de savoir traiter le volume élevé des listes de liens échangés qui peut augmenter de façon exponentielle tour après tour. Bien que le coût du stockage et la complexité de calcul peuvent être gérés, les coûts de communication ne sont pas négligeables. Ainsi, des protocoles d'échange efficaces sont nécessaires.

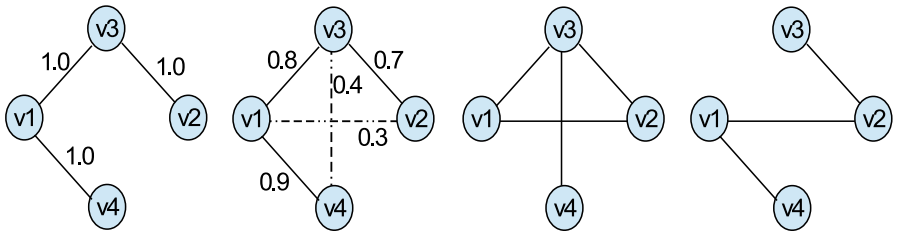


Figure B.2: La sémantique de l'incertitude des liens. De gauche à droite: graphe d'origine, graphe incertain et deux graphes échantillonnés de sortie

B.2 Bilan des problèmes

Dans cette section, nous décrivons les solutions pour résoudre les trois problèmes abordés dans la thèse: l'anonymisation de graphes sociaux, la détection de communautés privées et l'échange de liens privés.

B.2.1 Anonymisation de Graphes Sociaux

Nous étudions l'anonymisation de graphes sociaux sous deux angles: la sémantique de l'incertitude et la vie privée différentielle.

L'intuition derrière l'utilisation de la sémantique d'incertitude pour anonymiser les graphes sociaux est illustrée sur la Figure B.2. Le vrai graphe est transformé en un graphe incertain en ajoutant de nouveaux liens (appelés liens potentiels) et en attribuant des probabilités de liens. Les graphes de sortie sont produits à partir du graphe incertain en échantillonnant indépendamment les liens. Les liens probabilistes peuvent être calculées explicitement comme dans [11] ou implicitement par marches aléatoires [66]. Les performances des différents schémas sont quantifiées par le niveau de vie privée et les métriques d'utilité. Les métriques de confidentialité peuvent être des mesures de quantité d'information comme dans l'approche par (k, ϵ)-obscurcissement [11] ou d'incorrection en fonction du degré (Chapitre 3). Les métriques d'utilité couramment utilisées sont basées sur le degré de nœuds et les statistiques de chemin [12,100,107]. De toute évidence, plus il y a de liens potentiels et plus il y a de liens incertains (c'est-à-dire la probabilité des liens se rapproche de 0.5), plus la vie privée est protégée (par exemple par la baisse des risques de ré-identification) mais l'utilité décroit (la structure du graphe est déformée). Tous les systèmes d'anonymisation visent à optimiser le compromis entre ces deux aspects concurrents.

La vie privée différentielle [33] propose une définition formelle de la vie privée avec beaucoup de propriétés intéressantes: pas d'hypothèses sur la capacité de calcul et les informations possédées par les attaquants, agnosticité par rapport aux types de données, composabilité [63]. Les algorithmes de vie privée différentielle associent une quantité de bruit à la *sensibilité* du calcul i.e., la variation du résultat du calcul suivant une (petite) variation de l'entrée (en ajoutant par exemple un lien au graphe d'entrée), d'où le terme "différentiel". Une faible sensibilité implique moins de bruit à ajouter et vice-versa. Parce que les liens dans les graphes non-orientés et simples sont généralement supposés indépendants, le mécanisme standard de Laplace est applicable (en ajoutant par exemple un bruit de Laplace à chaque cellule de la matrice d'adjacence). Cependant, cette approche peut sérieusement détériorer la structure du graphe. Des méthodes récentes pour publier des graphes dans l'approche de la vie privée différentielle tentent de réduire la sensibilité du graphe de différentes façons. Les schémas de [89, 100] utilisent *dK-série* [60] pour résumer le graphe par une distribution des corrélations de degré. La sensibilité globale de

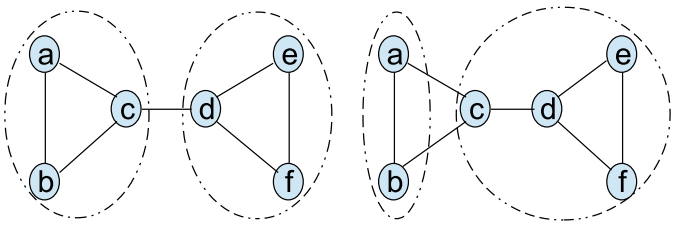


Figure B.3: Détection de communautés: un bon regroupement (gauche), un regroupement bruité (droit)

1K-série (resp. 2K-série) est 4 (resp. $O(n)$). Une sensibilité inférieure à $O(\sqrt{n})$ est proposée dans [101] par une analyse spectrale des graphes. Les travaux les plus récents *Density Explorer Reconstruct* (DER) [21] et HRG-MCMC [105] réduisent même la sensibilité du graphe à $O(\log n)$. Toutefois, les deux admettent une complexité quadratique $O(n^2)$, limitant leur intérêt aux seuls graphes de tailles moyennes.

B.2.2 Détection de Communautés Privées

Les structures communautaires sont très fréquentes dans les réseaux réels. Les réseaux sociaux proposent des groupes communautaires en fonction de la localisation géographique, des intérêts, de la profession, etc... Des communautés apparaissent dans les réseaux d'information (comme World Wide Web) sous la forme de groupes de pages se citant mutuellement. Les réseaux métaboliques font émerger des communautés sur la base de groupements fonctionnels de protéines. Réseaux de citations possèdent des communautés par sujet de recherche.

Pour la détection de communautés non-privées, nous disposons du vrai graphe et nous pouvons exécuter des algorithmes de détection de haute qualité afin d' obtenir des groupes de nœuds de bonne qualité. Les bons regroupements peuvent indiquer une modularité élevée, une faible conductance, des nœuds semblables, etc. [38]. La Figure B.3 montre un exemple de détection de communautés avec un bon regroupement et un regroupement bruité.

Dans le cadre de la détection de communautés privée, les données sont fournies sous forme d'un graphe bruité sur lequel est appliqué un algorithme de détection. Le résultat est un ensemble de communautés bruitées (et donc avec une modularité différente ou d'autres métriques sur la qualité des regroupements considérés). Une alternative consiste à envoyer au gestionnaire de données un certain nombre de requêtes sur le vrai graphe et à obtenir des réponses bruitées. Le nombre de requêtes dépend du budget de vie privée qui nous est alloué (par exemple la valeur de ϵ dans l'approche par la vie privée différentielle). Sur la base des réponses bruitées nous calculons un regroupement bruité. Dans cette thèse, nous considérons la détection de communautés (sans chevauchement) privée par la technique de vie privée différentielle.

B.2.3 Echange de Liens Privés

Les graphes sociaux sont une source précieuse pour l'étude des sociétés de l'information, mais ces graphes ne sont pas publiés en clair pour des raisons de confidentialité. Au contraire, les graphes sociaux sont publiés sous diverses formes anonymisées avant d'être donnés aux tiers consommateurs. Dans l'architecture client-serveur classique des OSNs, le serveur conserve l'ensemble du graphe social. Pour fournir des graphes échantillonnés bruités, le serveur peut exécuter des programmes d'anonymisation sur le graphe social qu'il conserve. Si le graphe social est partagé entre plusieurs serveurs, nous pouvons recourir à l'anonymisation distribuée, comme par exemple dans [97]. Alternativement, les sites de réseaux sociaux fournissent des APIs (Facebook,

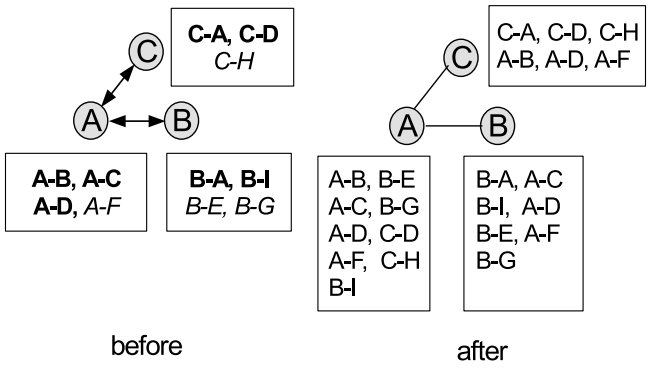


Figure B.4: Echange de liens privés. Liens vrais sont en gras. Liens faux sont italiques

Twitter) pour les robots d'exploration de données avec des limitations sur leur vitesse et des contraintes sur la vie privée (par exemple uniquement les listes d'amis publics sont disponibles). En utilisant cette méthode, les robots d'exploration de données peuvent recueillir des informations privées et reconstruire une vue partielle (locale) du graphe social.

Pour surmonter les contraintes imposées par les fournisseurs de services, nous partons du *point de vue de l'utilisateur*, c'est-à-dire d'un contributeur de l'OSN. Plus précisément, si les utilisateurs collaborent avec précaution, ils peuvent échanger des listes d'amis bruitées avec leurs voisins en plusieurs itérations pour obtenir une meilleure vue du graphe social réel. Nos idées sont basées sur le fait que les identités des utilisateurs sont publiques (par exemple les profils Facebook sont consultables [1]), mais les liens d'amitié ne le sont pas, sauf si un utilisateur met sa liste d'amis en mode public. Dans cette thèse, nous introduisons un modèle différent dans lequel les nœuds créent des listes de liens bruitées et les échangent avec leurs voisins en plusieurs passes. Le problème est original dans le fait que les données diffusées sur les liens sont les liens eux-mêmes. A la fin de la procédure, nous obtenons n graphes locaux où n est le nombre de nœuds. Nous supposons que tous les nœuds sont honnêtes-mais-curieux, c'est-à-dire ils suivent les protocoles bien définis et essaient d'en déduire les liens véritables à partir des listes de liens bruitées qui leur sont envoyées. La Figure B.4 représente l'idée de base de l'échange de lien. Chaque nœud ajoute du bruit à ses listes d'amis (en gras) en créant de faux liens (en italique). Ensuite, les paires de nœuds connectés effectuent l'échange de liens et chaque nœud supprime automatiquement les liens en double.

B.3 Contributions

Les trois problèmes ci-dessus sont présentés en détail successivement dans les Chapitres 3, 4, 5 et 6. Nous avons obtenu les contributions clés suivantes.

- **Chapitre 3:** Nous proposons un modèle général appelé *matrice d'adjacence incertaine* (UAM) pour l'anonymisation de graphe via la sémantique de liens incertains. La propriété clé de ce modèle est que les degrés attendus de tous les nœuds doivent être préservés dans le graphe anonymisé. Nous montrons que *(k, ϵ)-obf* [11] et *RandWalk* [66] s'adaptent bien à notre modèle, puis nous analysons leurs désavantages. Nous introduisons la technique de *Variance Maximizing* (MaxVar) qui satisfait toutes les propriétés de l'UAM. Elle réalise un bon compromis vie privée-utilité grace à deux observations clés: les liens potentiels sont créés entre deux noeuds proches dans le graphe réel (et sont donc plausibles et utiles) et la variance totale du degré d'un nœud est maximisée par un programme quadratique simple

(améliorant l'incertitude sur le nombre de liens et donc l'anonymat). Afin d'établir une méthodologie de comparaison équitable entre les schémas d'anonymisation de graphes, cette thèse introduit un cadre de quantification générique mettant en avant la mesure de distorsion (également appelé *incorrection* dans [92]) pour mesurer les risques de ré-identification des nœuds. Concernant le score d'utilité, des mesures typiques de graphes [11, 107] sont choisies. Nous menons une étude comparative des approches mentionnées sur trois grands graphes du monde réel et montrons l'efficacité de nos solutions.

- **Chapitre 4:** Nous analysons les deux défis clés pour la publication de graphes avec la technique de vie privée différentielle: l'immense espace des résultats possibles et la notion de cohérence. Nous justifions aussi la relaxation de ϵ à $\ln n$ en utilisant le concept de ρ-identifiabilité [52]. Nous prouvons une borne supérieure du budget de vie privée ϵ que tout système différentiellement privé (pour des graphes) ne doit pas dépasser. La limite supérieure est validée par notre schéma linéaire Top-m-Filter (TmF) et le schéma EdgeFlip [69] obtenu par d'autres auteurs. Par une analyse théorique plus profonde, nous prouvons la convergence rapide d' EdgeFlip et révélons ses limites. TmF et EdgeFlip présentent un comportement cohérent pour des régimes ayant un budget de vie privée élevé. Nous introduisons *HRG-FixedTree* pour réduire le temps d'exécution du schéma *HRG inférence* [105] par plusieurs ordres de magnitude, ce qui rend possible d'effectuer l'inférence sur de grands graphes. Nous présentons également le schéma en temps linéaire 1K-série qui est basé sur le modèle de configuration [73]. Nous procédons à une évaluation approfondie sur des graphes de tailles petites, moyennes et grandes pour étudier quelle méthode fonctionne le mieux selon les différents régimes de vie privée.

- **Chapitre 5:** Nous analysons les défis majeurs de la détection de communautés dans la vie privée différentielle [78]. Nous expliquons pourquoi des techniques empruntées à *k-Means* échouent et comment la difficulté rencontrée avec les systèmes de recommandation différentiellement privés justifie une relaxation de ϵ. Nous concevons un algorithme de perturbation d'entrée *LouvainDP* qui fonctionne en temps linéaire en utilisant la technique de filtrage passe-haut de [26] et la méthode Louvain [9]. Nous proposons aussi un schéma de perturbation d'algorithme *ModDivisive*, qui procède par division en utilisant la fonction de score basée sur la modularité et le mécanisme exponentiel. Nous prouvons que la modularité a une faible sensibilité globale et ModDivisive fonctionne aussi en temps linéaire. Nous procédons à une évaluation approfondie sur de vrais graphes de différentes tailles et montrons le gain de performance de LouvainDP et ModDivisive sur l'état de l'art.

- **Chapitre 6:** Nous introduisons un problème d'échange de liens privés comme une alternative à l'exploration de graphe social et l'anonymisation de données centralisée. Le problème est distribué et offre un compromis de vie privée-utilité pour tous les nœuds. Notre problème est unique en ce sens que les données diffusées sur les liens sont les liens eux-mêmes. Nous présentons deux schémas pour (α, β)-échange: l'un avec des techniques de base et l'autre basé sur les filtres de Bloom. Nous protégeons les liens privés en ajoutant de faux liens et exigeons que la probabilité de propagation d'un lien soit atténuée avec la distance entre l'origine et la destination du message. Nous analysons les avantages et les désavantages de chaque schéma. Nous évaluons nos propositions de systèmes sur différents modèles de graphes synthétiques et tirons un certain nombre de conclusions critiques.

Publications Certaines parties de cette thèse ont été publiées et soumises aux conférences et revues suivantes:

- H. H. Nguyen, A. Imine, and M. Rusinowitch. A Maximum Variance Approach for Graph Anonymization (FPS 2014)

- H. H. Nguyen, A. Imine, and M. Rusinowitch. Anonymizing Social Graphs via Uncertainty Semantics (ASIACCS 2015)

- H. H. Nguyen, A. Imine, and M. Rusinowitch. Differentially Private Publication of Social Graphs at Linear Cost (ASONAM 2015)

- H. H. Nguyen, A. Imine, and M. Rusinowitch. Network Structure Release under Differential Privacy (Transactions on Data Privacy, under 1st revision)

- H. H. Nguyen, A. Imine, and M. Rusinowitch. Detecting Communities under Differential Privacy (WPES 2016)

- H. H. Nguyen, A. Imine, and M. Rusinowitch. Private Link Exchange over Social Graphs (in preparation)

B.4 Vie Privée Différentielle: Une Introduction Brève

B.4.1 Motivation

Aujourd'hui, les données sont produites et collectées à un rythme phénoménal. L'analyse des énormes masses de données apporte des avantages sans précédents, mais menace également la vie privée des personnes. Nous devons faire face au défi paradoxal d'apprendre des informations utiles sur une population [35] sans rien apprendre sur un individu. Beaucoup de solutions ont été proposées pour résoudre ce paradoxe, mais pour chaque solution proposée de nouvelles vulnérabilités ont été trouvées. A la différence des paradigmes précédents, la vie privée différentielle nous fournit une avancée prometteuse: peu importe ce que l'adversaire sait sur vous à partir de sources d'information auxiliaires, votre *participation* à un ensemble de données ne pourra etre déduite (d'où le nom vie privée "différentielle").

La vie privée différentielle est un modèle formel de confidentialité initialement développé pour des données tabulaires et offre des garanties de confidentialité solides sans dépendre de la connaissance de l'adversaire, de sa puissance de calcul ou de son comportement ultérieur [33, 35]. Parce que la protection absolue de *vie privée* est impossible (voir [32] et [35, Section 1.1]), la vie privée différentielle offre une protection relative (*vie privée relative*) qui se révèle utile pour de nombreuses applications dans un large éventail de tâches d'analyse de données. De plus la vie privée différentielle a des relations étroites avec d'autres domaines tels que la cryptographie, les statistiques, la complexité, la combinatoire, la conception de mécanismes (théorie des jeux) et l'optimisation. Des résultats importants sur la vie privée différentielle sont présentés systématiquement dans [35] où les auteurs expliquent les aspects essentiels de ce concept de protection de la vie privée.

Le principe de la vie privée différentielle est vérifié si deux bases de données D et D' tel que $D' = D \cup \{X\}$, c'est-à-dire D et D' ne diffèrent que par un seul élément, les distributions de probabilités sur les résultats de D et D' fournies par la vie privée différentielle sont "essentiellement les mêmes". Plus formellement,

Definition B.1. *Un algorithme randomisé $\mathcal{A}$ est (ϵ, δ)-differentially private ((ϵ, δ)-DP en raccourci) si, pour deux ensembles quelconques de données voisins D and D', et pour tout résultat $O \in Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(D) \in O] \leq e^{\epsilon} Pr[\mathcal{A}(D') \in O] + \delta$$

Si $\delta = 0$, nous avons la notion de ϵ-DP qui est prouvée plus stricte que (ϵ, δ)-DP [31], c'est-à-dire (ϵ, δ)-DP nécessite moins de distorsion (bruit aléatoire) que ϵ-DP.

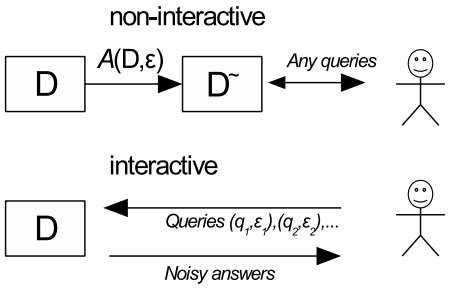


Figure B.5: Interactif vs. Non-interactif

La vie privée différentielle assure simplement que la participation à un ensemble de données ne sera pas révélée. Il est très possible que les conclusions tirées de l'ensemble de données reflètent des informations statistiques sur un individu. Par exemple, étant donné un ensemble de données de 100 personnes, 80 d'entre eux ont une certaine propriété P. Les réponses bruitées aux requêtes sur la taille de l'ensemble de données et sur le nombre d'utilisateurs ayant la propriété P sont, par exemple, 101.3 et 78.6 respectivement. De ces résultats, un analyste (ainsi qu'un attaquant) peut estimer que tout utilisateur a la propriété P avec une probabilité de 77.6 %, très proche des statistiques réelles de 80 %. Cependant, ceci est une information statistique, qui n'est pas liée à la participation d'un individu particulier à l'ensemble des données, comme le garantit la vie privée différentielle.

Il existe deux approches connues pour publier des données dans le cadre de la vie privée différentielle: le mode *interactif* et le mode *non-interactif* comme illustré sur la Figure B.5. Dans un cadre non-interactif, le gestionnaire des données publie un ensemble de données bruitées $\tilde{D}$ et s'arrête. Les utilisateurs peuvent effectuer toutes les opérations qu'ils souhaitent sur $\tilde{D}$. Dans les dispositifs interactifs, les utilisateurs sont autorisés à présenter un certain nombre de requêtes (q_i, ϵ_i) où q_i est la requête sur D et ϵ_i est le budget de q_i tel que $\sum_i \epsilon_i \leq \epsilon$. Cela signifie qu' après une requête, le budget de vie privée est réduit et quand il atteint zéro, il n'existe plus de requêtes autorisées.

4.2 Compositions Rendant la Vie Privée Différentielle Programmable

Non seulement la vie privée différentielle est formellement définie mais elle est également équipé de propriétés de composition intéressantes: la composition séquentielle et la composition parallèle (voir la Section 4.2 pour plus de détails). La composition séquentielle signifie que lorsque nous exécutons une série d'algorithmes randomisés $\mathcal{A}_i(\epsilon_i)$ sur le même ensemble de données D, le budget total de vie privée ϵ de l'opération sera la somme des ϵ_i. La composition parallèle stipule que lorsque nous exécutons une série d'algorithmes randomisés $\mathcal{A}_i(\epsilon_i)$ sur les sous-ensembles *disjoints* D_i de D, le budget effectif de vie privée de l'opération sera le maximum des ϵ_i.

Ces compositions permettent de décomposer de nombreux algorithmes complexes en étapes dans lesquelles les mécanisme de Laplace et le mécanisme exponentielle jouent le rôle de blocs de construction. Dans la plupart des cas, la preuve de ϵ-DP pour un algorithme donné est obtenu automatiquement. La tâche qui reste pour les concepteurs d'algorithmes est de réduire la variance des résultats aléatoires afin d'obtenir une meilleure utilité. À notre connaissance, aucun des concepts de confidentialité antérieurs comme la k-anonymité [95], l-diversité [59], t-proximité [54] et leurs variantes possède ces propriétés de composition.

Bibliography

[1] Facebook Directory. https://www.facebook.com/directory.

[2] J. Abawajy, M. I. Ninggal, and T. Herawan. Privacy preserving social network data publication. 2016.

[3] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914. ACM, 2013.

[4] 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.

[5] N. T. Bailey et al. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[6] S. Banerjee, N. Hegde, and L. Massoulie. The price of privacy in untrusted recommender systems. *Selected Topics in Signal Processing, IEEE Journal of*, 9(7):1319–1331, 2015.

[7] 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.

[8] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.

[9] 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.

[10] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

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

[12] 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.

[13] F. Bonchi, A. Gionis, and T. Tassa. Identity obfuscation in graphs through the information theoretic lens. In *ICDE*, pages 924–935. IEEE, 2011.

[14] C. Borgs, J. Chayes, and A. Smith. Private graphon estimation for sparse graphs. In *Advances in Neural Information Processing Systems*, pages 1369–1377, 2015.

[15] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. *On modularity-np-completeness and beyond*. Citeseer, 2006.

[16] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[17] A. Campan, Y. Alufaisan, T. M. Truta, and T. Richardson. Preserving communities in anonymized social networks. *Transactions on Data Privacy*, 8(1):55–87, 2015.

[18] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *In Privacy, Security, and Trust in KDD Workshop (PinKDD)*, 2008.

[19] T. Chakraborty, S. Srinivasan, N. Ganguly, A. Mukherjee, and S. Bhowmick. On the permanence of vertices in network communities. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1396–1405. ACM, 2014.

[20] K. Chatzikokolakis, C. Palamidessi, and M. Stronati. Constructing elastic distinguishability metrics for location privacy. *Proceedings on Privacy Enhancing Technologies*, 2015(2):156–170, 2015.

[21] 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.

[22] S. Chen and S. Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664. ACM, 2013.

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

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

[25] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

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

[27] M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011.

[28] J. Creusefond, T. Largillier, and S. Peyronnet. On the evaluation potential of quality functions in community detection for different contexts. In *International Conference and School on Network Science*, pages 111–125. Springer, 2016.

[29] N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, pages 1–12. ACM, 2007.

[30] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[31] A. De. Lower bounds in differential privacy. In *Theory of Cryptography*, pages 321–338. Springer, 2012.

[32] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210. ACM, 2003.

[33] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

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

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

[36] A. M. Fard, K. Wang, and P. S. Yu. Limiting link disclosure in social network analysis through subgraph-wise perturbation. In *EDBT*. ACM, 2012.

[37] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

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

[39] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *Computers, IEEE Transactions on*, 52(2):139–149, 2003.

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

[41] G. Giakkoupis, R. Guerraoui, A. Jégou, A.-M. Kermarrec, and N. Mittal. Privacy-conscious information diffusion in social networks. In *Distributed Computing*, pages 480–496. Springer, 2015.

[42] R. L. Graham. *Concrete mathematics:[a foundation for computer science; dedicated to Leonhard Euler (1707-1783)]*. Pearson Education India, 1994.

[43] R. Guerraoui, A.-M. Kermarrec, R. Patra, and M. Taziki. D2p: distance-based differential privacy in recommenders. *Proceedings of the VLDB Endowment*, 8(8):862–873, 2015.

[44] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *Theory of Cryptography*, pages 339–356. Springer, 2012.

[45] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178. IEEE, 2009.

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

[47] 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.

[48] D. A. Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[49] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography*, pages 457–476. Springer, 2013.

[50] K. Kenthapadi, A. Korolova, I. Mironov, and N. Mishra. Privacy via the johnson-lindenstrauss transform. *arXiv preprint arXiv:1204.2606*, 2012.

[51] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204. ACM, 2011.

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

[53] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM TKDD*, 1(1):2, 2007.

[54] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

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

[56] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[57] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[58] 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.

[59] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

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

[61] F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013.

[62] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE, 2007.

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

[64] A. Medus, G. Acuna, and C. Dorso. Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications*, 358(2):593–604, 2005.

[65] 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.

[66] P. Mittal, C. Papamanthou, and D. Song. Preserving link privacy in social network based systems. In *NDSS*, 2013.

[67] A. Moffat, R. M. Neal, and I. H. Witten. Arithmetic coding revisited. *ACM Transactions on Information Systems (TOIS)*, 16(3):256–294, 1998.

[68] Y. Moreno, R. Pastor-Satorras, and A. Vespignani. Epidemic outbreaks in complex heterogeneous networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 26(4):521–529, 2002.

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

[70] A. Narayanan, E. Shi, and B. I. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1825–1834. IEEE, 2011.

[71] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[72] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*, pages 173–187. IEEE, 2009.

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

[74] M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[75] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

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

[77] H. H. Nguyen, A. Imine, and M. Rusinowitch. Differentially private publication of social graphs at linear cost. In *ASONAM 2015*, 2015.

[78] H. H. Nguyen, A. Imine, and M. Rusinowitch. Detecting communities under differential privacy. In *WPES*. ACM, 2016. to appear.

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

[80] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *KDD*, pages 81–90. ACM, 2002.

[81] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.

[82] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005.

[83] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey. High quality, scalable and parallel community detection for large real graphs. In *WWW*, pages 225–236. ACM, 2014.

[84] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[85] S. Raskhodnikova and A. Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *arXiv preprint arXiv:1504.07912*, 2015.

[86] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[87] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[88] 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.

[89] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *SIGCOMM*, pages 81–98. ACM, 2011.

[90] E. Shen and T. Yu. Mining frequent graph patterns with differential privacy. In *KDD*, pages 545–553. ACM, 2013.

[91] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *Proceedings of the third ACM conference on Recommender systems*, pages 157–164. ACM, 2009.

[92] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *SP*, pages 247–262. IEEE, 2011.

[93] G. Smith. On the foundations of quantitative information flow. In *Foundations of Software Science and Computational Structures*, pages 288–302. Springer, 2009.

[94] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin. Differentially private k-means clustering. *arXiv preprint arXiv:1504.05998*, 2015.

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

[96] 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.

[97] T. Tassa and D. J. Cohen. Anonymization of centralized and distributed social networks by sequential clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 25(2):311–324, 2013.

[98] A. Vázquez. Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations. *Physical Review E*, 67(5):056104, 2003.

[99] S. Voulgaris, D. Gavidia, and M. Van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005.

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

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

[102] L. Wu, X. Ying, and X. Wu. Reconstruction from randomized graph via low rank approximation. In *SDM*, volume 28, pages 60–71. SIAM, 2010.

[103] W. Wu, Y. Xiao, W. Wang, Z. He, and Z. Wang. k-symmetry model for identity anonymization in social networks. In *EDBT*, pages 111–122. ACM, 2010.

[104] X. Wu, X. Ying, K. Liu, and L. Chen. A survey of privacy-preservation of graphs and social networks. In *Managing and mining graph data*, pages 421–453. Springer, 2010.

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

[106] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, pages 745–754. IEEE, 2012.

[107] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, volume 8, pages 739–750. SIAM, 2008.

[108] X. Ying and X. Wu. On link privacy in randomizing social networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 28–39. Springer, 2009.

[109] X. Ying and X. Wu. On link privacy in randomizing social networks. *Knowledge and information systems*, 28(3):645–663, 2011.

[110] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Private release of graph statistics using ladder functions. In *SIGMOD*, pages 731–745. ACM, 2015.

[111] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540. ACM, 2009.

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

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

Résumé

La vie privée est une préoccupation des utilisateurs des réseaux sociaux. Les réseaux sociaux offrent des services gratuits en échange d'informations personnelles. Les réseaux sociaux sont aussi une source de données précieuses pour des analyses scientifiques ou commerciales. Le problème de la protection des données privées au sein des réseaux sociaux demande des solutions efficaces aussi bien dans des contextes centralisés que décentralisés. Cette thèse aborde trois problèmes de confidentialité des réseaux sociaux: l'anonymisation de graphes sociaux, la détection de communautés privées et l'échange de liens privés.

Nous abordons le problème d'anonymisation de graphes via deux sémantiques différentes: la sémantique de l'incertitude et l'intimité différentielle. Pour la sémantique de l'incertitude, nous proposons un modèle général appelé Uncertain Adjacency Matrix (UAM) qui préserve dans le graphe anonymisé les degrés des noeuds du graphe non-anonymisé. Nous analysons deux schémas proposés récemment et montrons leur adaptation dans notre modèle. Nous rappelons également les inconvénients de chaque schéma et présentons notre approche dite MaxVar. Les expériences sur de grands graphes sociaux démontrent l'efficacité de MaxVar. Pour la technique d'intimité différentielle, le problème devient difficile en raison de l'énorme espace des graphes anonymisés possibles. Un grand nombre de systèmes existants ne permettent pas de relâcher le paramètre (appelé budget) contrôlant la vie privée, ni de déterminer sa borne supérieure. Dans notre approche nous pouvons calculer cette borne. Nous introduisons le nouveau schéma Top-m-Filter de complexité linéaire et améliorons la technique récente EdgeFlip. L'évaluation de ces algorithmes sur une large gamme de graphes donne un panorama de l'état de l'art.

Nous présentons le problème original de la détection de la communauté dans le cadre de l'intimité différentielle. Un grand nombre d'algorithmes pour la détection de communautés ont été proposées. Cependant, ils ne protègent pas la vie privée. Nous analysons les défis majeurs du problème et nous proposons quelques approches pour les aborder sous deux angles: par perturbation d'entrée et par perturbation d'algorithme. Pour la perturbation d'entrée, nous proposons la méthode LouvainDP qui applique l' algorithme Louvain sur un super-graphe bruité. Dans la seconde approche nous proposons ModDivisive qui utilise le mécanisme exponentiel avec la modularité comme score. Nous avons évalué nos techniques sur des graphes réels de différentes tailles et montré leur performance par rapport à l'état-de-l'art.

Certains réseaux sociaux en ligne connus conservent des données privées de manière centralisée, ce qui empêche les utilisateurs d'avoir une vue globale même approximative du graphe social. Nous montrons comment traiter ce problème dans le cas des listes d'amis avec des utilisateurs qui collaborent avec précaution, en échangeant des listes d'amis bruitées avec leurs voisins en plusieurs étapes. La solution est donc un protocole d'échange simple que nous avons pu évaluer sur différents modèles de graphes synthétiques. Le problème est unique en ce sens que les données diffusées sur les liens sont les liens eux-mêmes.

Mots-clés: Réseaux sociaux, Incertaine matrice d'adjacence, Maximum Variance, Vie privée différentielle, Top-m-Filte, détection de communautés, LouvainDP, ModDivisive, Echange intime des liens, (α, β)-échange

Abstract

Privacy is a serious concern of users in daily usage of social networks, especially when online social networks offer free services in exchange for large collection of user information. At the same time, social networks are a valuable data source for large-scale studies on social organization and evolution and are usually published in anonymized forms. On the other side, by participating

to social networks, users keep their own data and they may use the infrastructure of service providers to gather local views of the network to some extent not only restricted to 1-hop friends, for example by exchanging noisy links. To this end, the problems of privacy protection for social network data are still calling for effective and efficient approaches both in centralized and decentralized settings.

This thesis addresses three privacy problems of social networks: graph anonymization, private community detection and private link exchange. The main goal is to provide new paradigms for anonymization of social graphs, private community detection as well as distributed aggregation of graphs via noisy link exchange processes. We start the thesis by giving the big picture of data privacy in social networks and clarify the categories to which our work belongs. Then we go to the three contributions as follows.

First, we tackle the problem of graph anonymization via two different semantics: uncertainty semantics and differential privacy. As for uncertainty semantics, we propose a general obfuscation model called Uncertain Adjacency Matrix (UAM) that keep expected node degrees equal to those in the unanonymized graph. We analyze two recently proposed schemes and show their fitting into the model. We also point out disadvantages in each method and present our scheme Maximum Variance (MaxVar) to fill the gap between them. Experiments on large social graphs demonstrate the effectiveness of MaxVar.

Using differential privacy, the problem is very challenging because of the huge output space of noisy graphs. 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 thesis, such a bound is provided. We introduce the new linear scheme Top-m-Filter (TmF) and improve the existing technique EdgeFlip. 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.

Second, we present the problem of community detection under differential privacy. A great number of algorithms for community detection have been proposed to deal with the increasingly complex networks. However, the problem of doing this in a private manner is rarely considered. We analyze the major challenges behind the problem and propose several schemes to tackle them from two perspectives: input perturbation and algorithm perturbation. For input perturbation, we propose the method LouvainDP which runs Louvain algorithm on a noisy super-graph. For algorithm perturbation, we design ModDivisive using exponential mechanism with the modularity as the score. We have thoroughly evaluated our techniques on real graphs of different sizes and verified their outperformance over the state-of-the-art.

Finally, we propose protocols for private link exchange over social graphs. It is motivated by the fact that current online social networks (OSN) keep their data secret and in centralized manner. To overcome this constraint, we may start from user perspective. More precisely, if users cautiously collaborate with one another, they can exchange noisy friend lists with their neighbors in several rounds to get better views of the true social graph. The problem is unique in the sense that the disseminated data over the links are the links themselves. We propose simple (α, β)-exchange protocols and evaluate them on various synthetic graph models.

Keywords: Social networks, Uncertain Adjacency Matrix, Maximum Variance, Differential privacy, Top-m-Filter, Community detection, LouvainDP, ModDivisive, Private link exchange, (α, β)-exchange