Privacy and Dynamics of Social Networks

Pedram Pedarsani

pedram.pedarsani@epfl.ch

PhD Thesis
École polytechnique fédérale de Lausanne

20 December 2012

Abstract

Over the past decade, investigations in different fields have focused on studying and understanding real networks, ranging from biological to social to technological. These networks, called complex networks, exhibit common topological features, such as a heavy-tailed degree distribution and the small world effect. In this thesis we address two interesting aspects of complex, and more specifically, social networks: (1) users' privacy, and the vulnerability of a network to user identification, and (2) dynamics, or the evolution of the network over time.

For this purpose, we base our contributions on a central tool in the study of graphs and complex networks: graph sampling. We conjecture that each network snapshot can be treated as a sample from an underlying network. Using this, a sampling process can be viewed as a way to observe dynamic networks, and to model the similarity of two correlated graphs by assuming that the graphs are samples from an underlying generator graph.

We take the thesis in two directions. For the first, we focus on the privacy problem in social networks. There have been hot debates on the extent to which the release of anonymized information to the public can leak personally identifiable information (PII). Recent works have shown methods that are able to infer true user identities, under certain conditions and by relying on side information. Our approach to this problem relies on the graph structure, where we investigate the feasibility of de-anonymizing an unlabeled social network by using the structural similarity to an auxiliary network. We propose a model where the two partially overlapping networks of interest are considered samples of an underlying graph.

Using such a model, first, we propose a theoretical framework for the de-anonymization problem, we obtain minimal conditions under which de-anonymization is feasible, and we establish a threshold on the similarity of the two networks above which anonymity could be lost. Then, we propose a novel algorithm based on a Bayesian framework, which is capable of matching two graphs of thousands of nodes - with no side information other than network structures. Our method has several potential applications, e.g., inferring user identities in an anonymized network by using a similar public network, cross-referencing dictionaries of different languages, correlating data from different domains, etc. We also introduce a novel privacy-preserving mechanism for social recommender systems, where users can receive accurate recommendations while hiding their profiles from an untrusted recommender server.

For the second direction of this work, we focus on models for network growth, more specifically for network densification, by using a sampling process. The densification phenomenon has been recently observed in various real networks, and we argue that it can be explained simply through the way we observe (sample) the networks. We introduce a process of sampling the edges of a fixed graph, which results in the super-linear growth of edges versus nodes and the increase of the average degree as the network evolves.

Keywords. network dynamics, network privacy, random graphs, graph sampling, graph matching, social networks, recommender systems, de-anonymization, densification

Résumé

Au cours de la dernière décennie, les recherches dans différents domaines ont porté sur l'étude et la compréhension des réseaux réelles, de biologique à sociaux à technologiques. Ces réseaux, appelés réseaux complexes, présentent des caractéristiques topologiques communes, comme une distribution des degrés à queue lourde et l'effet petit monde. Dans cette thèse, nous abordons deux aspects intéressants de ces réseaux, et plus particulièrement les réseaux sociaux: (1) la vie privée des utilisateurs, et la vulnérabilité d'un réseau à l'identification des utilisateurs, et (2) la dynamique, ou l'évolution du réseau au fil du temps.

Pour ce faire, nous basons nos contributions sur un outil central dans l'étude des graphes et des réseaux complexes: l'échantillonnage des graphes. Nous conjecturons que chaque instantané d'un réseau peut être considéré comme un échantillon d'un réseau sous-jacent. Avec cela, un processus d'échantillonnage peut être considéré comme un moyen d'observer des réseaux dynamiques, et de modéliser la similarité de deux graphes corrélés en supposant que les graphes sont des échantillons provenant d'un graphe sous-jacent qui les génère.

Cette thèse contient deux parties. Dans une première partie, nous nous concentrons sur le problème de la vie privée dans les réseaux sociaux. Plusieurs études ont montré que la diffusion public d'informations anonymes peut divulguer des informations personnelles identifiables (PII). Des recherches récentes ont développé plusieures méthodes afin de déduire l'identité des utilisateurs en utilisant des informations auxiliaires. Notre approche à ce problème repose sur la structure du graphe, où nous étudions la possibilité de préserver l'anonymat dans un réseau sociale en utilisant la similarité structurelle à un réseau auxiliaire. Nous proposons un modèle où les deux réseaux se recouvrant partiellement sont considérés comme des échantillons d'un graphe sous-jacent.

En utilisant un tel modèle, nous proposons d'abord un cadre théorique pour le problème de de-anonymisation. Nous obtenons des conditions minimales dans lesquelles la de-anonymisation est possible, et nous établissons un seuil sur la similitude des deux réseaux au-dessus duquel l'anonymat pourrait être perdue. Ensuite, nous proposons un nouvel algorithme reposant sur un cadre bayésien, qui est capable d'apparier deux graphes de milliers de nœuds - sans aucune autre information que les structures des réseaux. Notre méthode a plusieurs applications potentielles, par exemple, révéler les identités des utilisateurs dans un réseau anonyme en utilisant un réseau public similaire, le recoupement des dictionnaires de langues différentes, ou la corrélation des données provenant de différents domaines. En outre, nous introduisons un nouveau mécanisme pour préserver la vie privée dans les systèmes de recommandation, où les utilisateurs peuvent recevoir des recommandations précises, tout en cachant leurs profils aux serveurs de recommandation non sécurisés.

Dans une deuxième partie, nous nous concentrons sur les modèles de croissance du réseau, plus spécifiquement pour la densification du réseau, en utilisant un processus d'échantillonnage. Le phénomène de densification a été récemment observé dans divers réseaux réels, et nous croyons que cela peut s'expliquer simplement par la façon dont nous observons les réseaux. Nous introduisons un processus qui consiste à prélever un échantillon des arêtes d'un graphe fixe, ce qui entraîne la croissance super-linéaire des arêtes contre des nœuds et l'augmentation du degré moyen comme le réseau évolue.

Mots Clés. dynamique de réseaux, protection de la vie privée, graphes aléatoires, echantillonnage de graphes, couplage des graphes, réseaux sociaux, systèmes de recommandation, de-anonymisation, densification

Acknowledgments

Contents

1 Introduction **1**
1.1 Privacy in Social Networks 1
1.1.1 Approximate Graph Matching 3
1.2 Dynamics of Social Networks 3
1.3 Edge Sampling 4
1.4 Contributions 6
1.5 Thesis Outline 7

I Models for Matching Large Networks **9**

2 Network De-Anonymization **11**
2.1 Network De-Anonymization: a Matching Problem 11
2.2 Related Work 12

3 A Theoretical Framework **15**
3.1 Problem Definition 16
3.2 Conditions for Perfect Matching 18
3.2.1 Allowing a Fraction of Node Mismatches 23
3.2.2 Different sampling rates 24
3.3 On the Similarity of Real Networks 26
3.4 Discussion and Summary 30

4 An Iterative Bayesian Method for Graph Matching **33**
4.1 Problem Definition 34
4.2 General Method for Graph Matching 35
4.2.1 Pair-Wise Likelihoods 35
4.2.2 Anchor Mismatch 38
4.2.3 Models for Sampled Degree and Distance 39
4.3 Matching algorithm 41
4.3.1 Complexity of the Matching Algorithm 43
4.3.2 Normalization of Likelihoods 44
4.4 Experiments 44
4.4.1 Matching Two Sampled Graphs from EPFL E-mail Network 45
4.4.2 Matching Two Snapshots from EPFL E-mail Network 46
4.4.3 Estimation of Parameters and Priors 47
4.5 Summary 49

4.A Appendix 50

II Evolution of Social Networks 51

5 Network Densification 53
5.1 Introduction 53
5.2 Motivation 55
5.3 Model 56
5.3.1 Densification Regimes and the Value of Cutoff 59
5.4 Conditions for Power-Law Densification 61
5.4.1 Necessary Condition for Power-Law Densification 61
5.4.2 Sufficient Condition for Power-Law Densification 62
5.5 Numerical Evaluations 64
5.5.1 Observation Models 69
5.6 Real Data Experiments and Variations of the Model 70
5.6.1 Simple Accumulation Model 72
5.6.2 Simple Modulation Model 73
5.7 Related Work 77
5.8 Summary 78
5.A Appendix 78

III Preserving Privacy in Recommender Systems 81

6 Preserving Privacy in Collaborative Filtering 83
6.1 Introduction 83
6.2 Problem Statement 85
6.2.1 Definitions and Notations 85
6.2.2 Problem Definition 85
6.3 Proposed Method 86
6.3.1 Sketch of the Solution 86
6.3.2 The Model 87
6.3.3 Profile Aggregation 89
6.4 Evaluation Metrics 91
6.4.1 Privacy Measurement 91
6.4.2 Recommendation Accuracy 92
6.5 Experimental Results 93
6.6 Related Work 94
6.7 Summary 96

Conclusion 99

Bibliography 103

Index 111

Curriculum Vitæ 113

1 Introduction

Over the past decade, the study of real-world networks has gained particular attention in the research community. The empirical study of various types of networks, ranging from social to technological to biological, has led to a unifying theme known as *complex networks*. Surprisingly, many of these networks exhibit common topological features, such as heavy-tailed degree distribution, high clustering, community structure, and the small-world effect. These features incite the scientific community to seek models to explain the properties and dynamics of such networks. Surveys in this area include the works of Albert and Barabasi [6], Newman [81], and Boccaletti [14], and some prominent works include those of Watts and Strogatz [102], Willinger et al. [65], and Kleinberg [48]. Thanks to the accessibility of high computational power and due to the growing availability of large-scale network data, it is now feasible to investigate and analyze large networks, and to propose models and methods in order to address different characteristics and challenges inherent to such networks.

Among various types of complex networks, social networks have gained particular attention in the research community due to their massive popularity, rapid growth and commercial potential. The emergence of immense social networks with millions of users, such as Facebook, Twitter, Flickr, and Linkedin, has led to a new era in which human communication is not restricted to one-to-one interactions. Instead, information sharing and communications flow are accomplished via a dynamic network of user relationships.

1.1 Privacy in Social Networks

With the rapid growth of social networks , the sharing of sensitive information about users and their relations in the networks has become an accelerating trend. There are many reasons social network data are shared between organizations or even released into the public domain by network owners and social network operators. First, this information is very valuable for scientific purposes: the modest number of publicly available datasets has led to a broad variety of research projects and results. For example (and without any claim to exhaustiveness), promising research directions in this area include probabilistic modeling of network properties and dynamics (as cited above), real-data measurement and analysis [72, 71, 50, 20], and developing scalable algorithms to navigate and infer data from large-scale networks [55, 60]. Obviously, this requires great care in order to avoid the accidental release of sensitive information about individual users. As AOL's public relations disaster a few years ago [1] illustrates, simply anonymizing user identities is not sufficient to prevent an attacker from identifying individual users through other means. Second, online social networks are increasingly integrated with other services on the web, which requires a certain amount of sharing between organizations (e.g., Facebook third-party applications and Facebook connect function on third-party websites). Third, it has been recognized that social network information has a strong potential for marketing purposes (e.g., for churn prediction [93] or for targeted advertisement [43]). In all of these areas lurks the risk of accidental or deliberate violations of user privacy.

Users, often willingly, share personal identifying information about themselves, without a clue of who can access this information. In information systems, privacy in general refers to the ability of the users to control the spread of their personally identifiable information (PII). PII refers to the information that can be used either by itself or in combination with other PIIs to uniquely identify an individual, such as name, birth date, contact information, social security number, driver's license number, etc. Obviously, a naïve release of network data including users' PII is a privacy violation and should be avoided. Hence, network operators generally apply privacy-preserving measures before releasing network data. In the context of social networks, several works address this privacy issue [36, 51, 21], propose mechanisms to preserve users' privacy [107, 37, 41, 73], and suggest the vulnerability of social networks to different attacks [49, 105].

Anonymity refers to the state of an individual's personally identifiable information being publicly unknown[1]. Using this notion, a well-established privacy-preserving method is *anonymization*, where user identities are replaced with random unique identifiers. Anonymization is assumed to be equivalent to hiding PII for an individual user, and thus known as a privacy-preserving technique. However, recent works [78, 76, 75] have shown that a large-scale re-identification of social networks is possible even if the data is anonymized, and that assuming a fixed set of attributes as users' PII and hiding them might not be sufficient to preserve their identities. In other words, it is hard to define a fixed set of attributes as *identifying*, and that *any information that distinguishes one person from another can be used for re-identifying anonymous data [79].* Thus, although some information might be *unidentifying* itself, it *can* be identifying *in combination with others.* This motivates a novel research direction, where re-identification is done without access to users' information, rather through the use of side information about the user. Such information spans from the local structure of social networks, to web browsing history, to search histories, etc.

Recent works have shown methods that are able to infer the true user identities under certain conditions, by relying on side information [78, 7, 105]. This process is called *de-anonymization.* Basically, an attacker relies on some side information obtained from other (e.g., publicly available) resources, in order to reveal the identities of users in an anonymized network. This is indeed an obvious privacy breach, as revealing the identities exposes personal information about the users.

Several works have recently looked into network de-anonymization and investigated the success or failure of different existing or proposed de-anonymization techniques on real-network data. Most of the works in this area focus on proposing *algorithms* and *methods* for de-anonymizing networks, tested on various real datasets of online networks [68, 77]. Works in this category propose algorithms for either de-anonymizing *specific* users in publicly available network data [7], or de-anonymizing a fraction of *all* users in a network [105]. A major challenge in all these scenarios is scalability. Recently, algorithms applicable for de-anonymizing a large proportion of users in large networks have been introduced [78, 77]. These algorithms rely on a seed set of *already revealed* node pairs and use the initial seeds to propagate among all nodes and infer all user identities.

In this thesis, our approach to the privacy problem in social networks is from an adversary's

---

[1] In many cases anonymity has been assumed to be equivalent to privacy. A formal definition of different privacy notions can be found in [87].

point of view: We look at the de-anonymization of (almost) *all users* of an anonymized network, by using no side information other than the *structural* similarity to a known labeled network. As shown in the following chapters, this is equivalent to matching the nodes of two similar (correlated) graphs, by relying only on graph structures. We refer to this as *approximate graph matching*, a concept which has been used in other fields such as pattern recognition and image processing as well.

As a complementary work, we also investigate the privacy problem in social recommender systems. This is further elaborated in the end of this chapter, and also Chapter 6.

1.1.1 Approximate Graph Matching

We comment on the notion of approximate graph matching used in this thesis. Graph matching refers to a collection of methods that compare two or several graphs by their *structure*, i.e., without reference to any vertex or edge labels. For example, in computer vision, a picture can be represented compactly by a graph describing a set of image segments and their relative positions to each other. This graph can then be used to find qualitatively similar pictures in a database, where each picture is also represented as such a segment graph [103, 104]. Such a graph is *unlabeled*, i.e., represents an equivalence class of isomorphic graphs.

The problem of graph matching is similar to the classic graph isomorphism problem. Suppose we have two unlabeled copies $G_{1,2}$ of a graph and need to identify its isomorphisms. This is equivalent to finding a bijection of their vertex sets $V_{1,2}$ under which the two edge sets are equal. This bijection is unique if and only if the graph is asymmetric. It is known from the theory of random graphs that the class of asymmetric graphs is large [16][2], which suggests that reasonably dense graphs tend to be asymmetric. But the problem of identifying the correct bijection is NP [30] in general, and therefore (almost certainly) cannot be computed in polynomial time.

In many practical applications of graph matching, we need to match similar, but not necessarily identical graphs. In the example from computer vision, the query image might be slightly different from a target image in the database, which results in small differences in the segment graphs; we do not want this to prevent a match in the database. For this reason, there has been considerable interest in *approximate* graph matching [35, 98].

In the context of social networks privacy, we show that a de-anonymization problem can be expressed as an approximate graph matching problem, with the goal being to match the nodes of two structurally similar (but not identical) *large* graphs, and thus to re-identify the nodes of the anonymized graph. This is of course a NP hard problem, and we elaborate on our approach further below.

1.2 Dynamics of Social Networks

Various works model the behavior of social networks and their underlying properties. Some significant advances in this field include Kleinberg's works on navigability, link prediction, group formation, and on anonymity in social networks [48, 66, 8, 7], the works of Leskovec et al. on densification, on the evolution of real graphs, and on the patterns of influence in recommendation networks [63, 61, 57, 54, 58, 64], Newman's work on the structure of

---

[2] In the sense that the threshold function for asymmetry in the $G(n,p)$ random graph model is only $p = \log(n)/n$ [16, 44].

growing social networks [45], the studies of online social networks by Kumar et al. [9, 47], and Barabasi's works on scale-free networks and preferential attachment model [11, 10, 47].

Most of the works on complex networks focus on static scenarios. Recently, the investigation of the dynamics of social networks and their growth over time has gained attention as well. With the increasing availability of network data, data mining and analysis has become a promising direction to investigate the dynamics of such networks [33, 72], and to answer questions such as: How does a social network evolve over time? How does the network structure change? What can we say about the dynamics of edges and nodes of the social graph? Answers to these questions shed light on the evolution of social and, in general, complex networks, and they help us understand the network dynamics over time. Besides investigating network features by using real data, another direction of work is the development of plausible mathematical models that can explain or capture the observed properties. Several models have been proposed to capture social networks evolution [56, 83, 32, 52, 26].

Our contribution to this rich body of work includes a novel explanation for the *densification* phenomenon in complex networks. Densification is a phenomenon recently observed in the evolution of social networks by Leskovec et al. in [63, 61]. They investigated several complex and social networks, and observed that as the network evolves over time, the number of edges grows faster than the number of nodes, and the average degree increases over time. They propose a growth model called *forest fire model* that explains densification as a result of network growth, and the tendency of newly arrived nodes to generate more links to the existing nodes. Densification is a surprising phenomenon: Traditionally, in the study of the evolution of networks, the average degree was assumed to be fixed, and independent of the network size, whereas this recent finding suggests that a node is likely to generate more links when it arrives later in the process. In other words, the average number of connections of an individual increases as the size of the global network increases! This being counter-intuitive, we seek an alternative explanation for densification. We elaborate on our contribution in the end of this chapter.

1.3 Edge Sampling

Graph sampling is a mathematically tractable yet efficient method for social networks analysis. Taking samples of a large graph is an established method for acquiring a representative sample of the graph. In many real scenarios, the graph is huge, and many known algorithms for computing interesting graph measures (such as shortest path, centrality, betweenness, degree distribution, and clustering coefficient) are impractical for large graphs. Hence, the need to use graph sampling in practical scenarios, in order to estimate the properties of the original graph based on the samples.

Different sampling methods are proposed in the literature, and their performance in estimating various parameters of the original graph are evaluated [59, 101, 92, 91]. Some of these methods include sampling from the edges, sampling nodes, frontier sampling, and graph traversal techniques such as random walks, breadth-first search, and depth-first search. The choice of the sampling technique heavily depends on the application. For instance, as shown in [101], neither of the state-of-the-art graph sampling algorithms estimate well both clustering coefficient and degree distribution. Moreover, they show that the efficiency of each method also depends on the social network dataset. Recently, and with the growth of social networks, researchers have adopted the traditional sampling methods and proposed sampling algorithms for unbiased sampling of online social networks [100, 31].

In this thesis, we use graph sampling as a way to model an observed instance of a network. Note that in this work we do not intend to estimate properties of the original graph from the sampled graph. Instead, we suggest that an observed graph can be a sample of a hidden fixed underlying graph. In other words, we assume that there exists a hidden fixed network of all possibly existing connections in the background, and each graph that we observe is just a sample of this underlying graph. The intuition behind our approach is that in many social networks, e.g., online social networks, e-mail networks, recommendation networks, etc., usually only a portion of the whole network is observable, obtained either through crawling techniques from the web, or as publicly available data. This can be treated as a sample of an underlying hidden network.

More specifically, we introduce a simple model called *edge sampling*: Assuming we have a fixed underlying graph, to obtain the sampled graph, we sample each edge with some fixed probability independent of all other edges. Doing this, each sampling of the underlying graph yields one realization of the sampled graph.

We use the edge sampling model introduced above in the two core parts of this thesis, namely the privacy and dynamics of social networks, as follows:

Structural Similarity We use the sampling model as a way to model the structural similarity between two graphs. As described above (and further elaborated in the contributions section), our de-anonymization problem can be expressed as the problem of matching two structurally *similar* graphs. To model the notion of similarity, we assume the two graphs are edge-sampled from an underlying graph, where the sampling probability s (for each edge) controls the structural similarity of the two graphs. In the case of a social network, the underlying graph corresponds to the true relationships between individuals, and the two sampled graphs correspond to incomplete manifestations of the underlying graph (e.g., observable interactions between people, such as phone calls and e-mails), or observations of the same network at different points in time. Notice that s controls the correlation between the two graphs: For $s = 1$, the two graphs are identical and equal to the underlying graph, whereas for smaller s, the graphs overlap, and the amount of overlap depends on s.

Network Evolution We use the edge sampling model as a way to observe dynamic networks and to model network growth. The intuition behind this choice of sampling method is the following: We argue that in many empirical studies of complex networks, it is the links (or edges) that are observed directly, and the nodes (or vertices) are *only revealed indirectly* through the observation of links. For example, most studies of e-mail networks are based on a log of e-mail messages. An e-mail message exchanged between two e-mail addresses a and b is taken as evidence of a social link (a, b). At the same time, this message *reveals* the nodes a and b if these nodes were not already known. We believe that the direct observation of edges, which at the same time gradually reveals the nodes, is a feature of most empirical studies of complex networks. The whole social communications graph is not totally revealed, and what we observe is just a sample of an underlying hidden graph.

Using this intuition, we argue that network growth is at least partially explained by the gradual observation of nodes and links that exist permanently “in the background”. In other words, there exists a fixed underlying network that is not directly observable. For example, this network might represent the people in a large organization, and the social and professional ties that bind them. An edge of this network can be observed only once this edge "fires" (is sampled), e.g., a message is sent over this edge. We believe that in many situations, it is reasonable to assume that such a hidden network exists, which changes on a time-scale much longer than the sampling process. The network growth can then be modeled as a direct consequence of the edge sampling process, i.e., each snapshot of the network corresponds to a graph sampled from the underlying graph and a different edge-sampling probability.

We further elaborate on the use of the sampling model in the corresponding chapters.

1.4 Contributions

In this thesis, we address two fundamental issues in the analysis of social and complex networks: (1) privacy, and (2) dynamics. We make the following contributions:

De-Anonymizing Social Networks We investigate the problem of de-anonymizing the users of a social network and show that anonymization is not sufficient to keep the network private. Our contributions are two-fold:

1. We answer the following question: Can we assume that a sufficiently sparse network is inherently anonymous, in the sense that even with unlimited computational power, de-anonymization is impossible? We consider the scenario where an attacker has access to (1) the target network, an unlabeled graph of social network users, and (2) an auxiliary labeled graph of the same users, where the user relationships (edges) are correlated, but not equal, with the other graph. We ascertain the *conditions* under which we can establish a perfect matching between the nodes of the two graphs, thus de-anonymize the unlabeled graph. Our approach is to introduce a random graph model for this version of the de-anonymization problem; this model is parameterized by the expected node degree and a similarity parameter that controls the correlation between the two graphs over the same vertex set. We find on these parameters simple conditions that delineate the boundary of privacy, and we show that the mean node degree need grow only slightly faster than $\log n$ with network size n for nodes to be identifiable. Our results have policy implications on sharing anonymized network information [86].
2. We consider the same settings as above and propose an efficient and scalable algorithm to find *the* correct matching between the nodes. Although approximate graph matching for small networks has received considerable attention in the literature, heuristics for large-scale matching have appeared only recently and require an initial seed set of known pairs of corresponding nodes. We present a new graph matching algorithm that relies on a probabilistic model of the two graphs to be matched. This allows us to develop a clean Bayesian framework and to express graph matching as a bipartite matching problem. We construct an algorithm that builds the map incrementally and without any initial seed set. Our method is the first to express approximate graph matching of *large* graphs in a statistical framework and can be used to de-anonymize social networks, as well as several other graph matching applications [85].

Dynamics of Social Networks We provide a novel explanation for a recently observed phenomenon in the evolution of social and complex networks: Network *densification* occurs when the number of edges grows much faster than the number of nodes as the network evolves over time. In our approach, we use an interesting observation of how real network data is gathered and used: in many empirical studies we observe edges, and nodes are inferred only indirectly through the edges. Leveraging on this, we propose a new model called *edge sampling* to explain how densification arises. Our model is innovative, as we consider a fixed underlying graph and a process that discovers this graph by probabilistically sampling its edges. We show that this model possesses several interesting features and, in particular, that edges and nodes discovered can exhibit densification. Moreover, we show that there is a direct relationship between densification and the underlying graph possessing a heavy-tailed degree distribution. In other words, that almost all real graphs densify, because they have a skewed degree distribution. Our theoretical findings are supported by numerical evaluations of the model, together with performance evaluation on real data. Our results show that densification can be explained as an observation bias, rather than being a real network property, and that edge sampling is indeed a plausible alternative explanation for densification [84].

Privacy-Preserving Social Recommender Systems We propose a novel method for privacy preservation in collaborative filtering recommendation systems. We address the problem of protecting the users' privacy in the presence of an untrusted central server, where the server has access to user profiles. To avoid privacy violation, we propose a mechanism where users store an offline profile locally, hidden from the server, and an online profile on the server from which the server generates the recommendations. The online profiles of different users are frequently synchronized with their offline versions in an independent and distributed way. Using a graph theoretic approach, we develop a model where over time each user randomly contacts other users and modifies his own offline profile through a process known as aggregation. To evaluate the privacy of the system, we apply our model to the Netflix prize dataset to investigate the privacy-accuracy tradeoff for different aggregation types. We show that such a mechanism leads to a high level of privacy through a proper choice of aggregation functions but has a marginal negative effect on the accuracy of the recommendation system [97].

1.5 Thesis Outline

The thesis has three parts.

Part I, which contains the main body of this work, is devoted to the de-anonymization of social networks. In **Chapter 2** we explain the de-anonymization problem, express de-anonymization as a graph matching problem, and survey the related work in the subject. In **Chapter 3** we present our theoretical approach to the matching problem, introduce the model for generating two similar graphs, and - via a rigorous proof - derive the conditions on the network parameters under which de-anonymization is feasible. In **Chapter 4** we propose an iterative Bayesian method for the efficient matching of large sparse graphs, and thus network de-anonymization.

In **Part II** (**Chapter 5**) we introduce our innovative model to explain network evolution, and more specifically, network densification. Finally, in **Part III** (**Chapter 6**) we propose a novel privacy-preserving algorithm for collaborative filtering social recommender systems.

Part I

Models for Matching Large Networks

2 Network De-Anonymization

The emergence of online social networks such as Facebook, Twitter, MySpace and Linkedin with hundreds of millions of users implies that an unprecedented amount of user data is now in the hands of the providers of such services. Not surprisingly, the fair use of this information, the appropriate notions of privacy and security, and the technical and legal tools to control its sharing and dissemination have become controversial and hotly debated problems in the scientific community and beyond. A major issue that arises as a result of the popularity of online social networks is users' *privacy.* Many online, and almost all offline networks (e.g, phone calls, e-mail networks, etc.) provide their users with some privacy measures. More specifically, the access to users' identities and their relationships is restricted to prevent attackers from fetching sensitive information about network users.

In order to protect users' privacy and preserve their personally identifiable information, network owners anonymize the network through replacing users' identities with random unique identifiers, before the release of the data for research or marketing purposes. However, recently it has been shown that even anonymized users could be re-identified through relying on auxiliary information obtained from an external source. It is clear that the availability of side information (e.g., class labels for users such as from demographic information, or richer link information such as directed interactions, time stamps) can further aide an attacker for the re-identification of users. This motivates us to explore the anonymity of social networks and to investigate the robustness of anonymized network data to de-anonymization attacks.

In the current chapter and also Chapters 3 and 4, we highlight our approach and contributions to the network de-anonymization problem. We investigate a realistic scenario where we assume that the only exposed information is the structure of an anonymized social network, and our goal is to de-anonymize *all* users in the network by using auxiliary information, and yet *only structural information*, from some other publicly available social graph. We elaborate on this in the next section.

2.1 Network De-Anonymization: a Matching Problem

The following important observation is behind our approach: In most real cases, although nodes are anonymized in the released data of social networks, the structure of the graph is preserved. Network owners and operators usually anonymize users' data yet preserve the relationships between anonymous users, in order that the fully anonymized *graph* can still be used for research or application purposes. This is equivalent to having access to an unlabeled graph. Furthermore, we assume that an attacker has access to a similar but *labeled* network, in which user identities are known. Such a network can be obtained for example from public data (e.g., through mining a public social network on the web), or inferred from other sources. This type of attack is also considered in [78].

To give a concrete example, we ask whether it would be safe for an academic institution to release a database of anonymized e-mail or call logs, if an attacker has available to him a correlated but highly *incomplete* set of likely social links between the staff and students of that institution (e.g., by mining the public website of groups, departments, and so on). Could an attacker use this incomplete side information to reverse-engineer the anonymized identities in the database, and therefore the communication pattern of this university? More generally, most of us have many different online identities that are in different hands, and the social links in these different domains are probably not completely identical, but correlated. For instance, notice that in many online social networks, there are profiles that exist in several networks, e.g., a user with a profile on Facebook and Twitter, or a user is a member of two local networks such as a phone network and e-mail network in an organization. It is true that such networks exhibit similar structural properties, although the edge sets of the two graphs do not match perfectly (e.g., I do not have the exact same friend list in Twitter and Facebook, but my friend lists overlap).

Here, we assume that the attacker only has the graph structure for the re-identification of nodes. Our goal is to establish a mapping between the nodes of two structurally similar graphs, and thus introducing a way to re-identify nodes in an anonymized network by using the known identities of the other network. We can summarize our de-anonymization problem as *finding the correct mapping between the nodes of two structurally similar graphs without relying on node identities*, which we call *approximate graph matching.*

In this dissertation, we explore the problem of approximate graph matching explained above. Although we explain the problem for the specific case of network de-anonymization, our results in this thesis can be applied in general to any kind of approximate graph matching problem, in which we intend to match the nodes of two similar graphs by using only their structures. In order to model the similarity of two networks, we use the notion of *graph sampling* introduced in Chapter 1 to develop a model of *similar* or *correlated* graphs. We elaborate on this in Chapters 3 and 4.

Our contributions fall into two categories, which include the core part of the thesis in the next two chapters:

1. *Theoretical aspect of graph matching*: We assess the feasibility of graph matching by using only network structures, and we establish fundamental conditions (depending on graph structures) for matching to be feasible in theory (Chapter 3).

2. *Algorithmic aspect of graph matching*: We propose a clean statistical framework, together with an iterative algorithm for matching two large correlated sparse graphs (Chapter 4).

2.2 Related Work

We briefly summarize the related work in network de-anonymization and approximate graph matching. The works can be categorized as follows:

1. Research works relevant to network modeling with direct application for de-anonymizing users in social networks.

2. Research works in the area of graph isomorphism and approximate graph matching, mostly from applications in machine learning and pattern recognition problems.

In the first category, in [78], Narayanan et al. propose a novel algorithm, based purely on network topology, for de-anonymizing social networks. Their work is closest to our work in problem settings, where they use structural information to de-anonymize graphs of a very large size. Their method uses the cosine similarity between nodes and introduces a propagation algorithm to match all nodes. In a similar work [76], Narayanan et al. introduce a new simulated annealing-based weighted graph matching algorithm for the seeding step of de-anonymization; they also show how to combine de-anonymization with link prediction. In addition to the graph structures, both approaches above depend on the extraction of a small subset of *correctly mapped* nodes, used as *seeds* for mapping the remaining nodes. Furthermore, their method requires backtracking and revisiting the mapped nodes, through visual inspection or manual interventions.

Also in [77] and [75], Narayanan et al. present de-anonymization methodologies for sparse micro-data to de-anonymize movie viewing records in the Netflix Prize dataset and to identify authors based on their writing style through comparison with a corpus of texts of known authorship.

Backstrom et al. introduce active and passive attacks for the de-anonymization of social networks [7]. They show how target users can be identified in a very large network by identifying a neighborhood subgraph around the user using only network structure. They investigate the effectiveness of these attacks both theoretically and empirically. A limitation of active attacks is the necessity of creating fake (dummy) nodes in the social network before its release (which is of course a strong limitation in practice), while passive attacks are capable of re-identifying only a limited number of users, but without the need for fake nodes. Thus, the method works best for the de-anonymization of *specific* users within the network, or a small fraction of all users. A similar attack model is analyzed in [40], where an attacker is allowed to issue queries that reveal a k-hop subgraph around a target node; they analyze the privacy risk to the identity of the target node and to the presence of specific links, both using random graph models and real data.

Wondracek et al. [105] introduce a de-anonymization technique based on group membership information of individuals in social networks. They show that information about the group memberships of a user is often sufficient to uniquely identify this user, or at least to significantly reduce the set of possible candidates and to assess the feasibility of the attack both theoretically and empirically.

In the second category, mainly in the area of pattern recognition and machine learning, several techniques have been proposed for exact and approximate graph matching. In [82], Cordella et al. propose a so-called VF algorithm as a solution for *exact* subgraph matching, or subgraph isomorphism, exhibiting less complexity compared to the famous Ullmann backtracking algorithm [99]. In [98], Tian and Patel suggest an approximate graph matching tool (TALE) through a novel indexing method that incorporates graph structural information in a hybrid index structure. Some other methods, proposed for approximate graph matching and used mainly for matching patterns or images, include random walks on graphs [35], using EM algorithm and singular value decomposition [67], and the Bayesian edit-distance criterion [95, 74]. Hancock et al. study graph matching in pattern recognition area in several works [28, 38, 106, 27]. Although in some of these works the structural information for matching graphs is used, the approximate matching problem in such cases is generally defined as node mismatches or inconsistencies of node attributes. Also, because of the complexity of matrix manipulation and computation of complex probability distributions, such methods are generally not feasible for application to large networks.

Some of the works in the second category specifically focus on a Bayesian approach to graph matching [103, 104, 89, 12]. In most of these approaches, the goal is to match a source graph to one of the multiple correlated target graphs, thus differing from our approach to match all users of two correlated networks.

Our contribution to this existing body of work is (i) to introduce a mathematically tractable, parsimonious model for the problem of matching two similar graphs and to derive asymptotic bounds in terms of fundamental parameters for network anonymity, independently of specific algorithms, and (ii) to propose a novel algorithm for graph matching - based on a Bayesian framework for computing the likelihood of matching for two graphs - that iteratively maps nodes using only structural properties of the graphs and without any prior information of any previously mapped nodes.

3 A Theoretical Framework

In this chapter, we pursue a theoretical approach to the matching problem, and formulate the vulnerability of a network to de-anonymization as a function of its structural similarity with an auxiliary network.

In the literature, we still lack a thorough understanding of the conditions under which de-anonymization is feasible. We want to be able to ascertain when a network's anonymity can be guaranteed, given the side information and computational resources available to an attacker. In this work, we make a step in this direction. We study a challenging version (from the perspective of an attacker) of the matching problem, where the attacker has no side information about nodes in the network to be attacked, other than network structure, specifically a correlated version of the edge set of that network obtained from other sources.

To the best of our knowledge, ours is the first work to pursue a theoretical treatment of network de-anonymization problem, and in particular, to consider its feasibility for *large* networks. Our contributions in this chapter are three-fold: (1) We explore fundamental limits for de-anonymization regardless of the specific algorithm employed, and investigate the relationship between network parameters and the possibility of guaranteeing anonymity in such networks. (2) We introduce a mathematically tractable model that captures the notion of correlated networks, and uses the idea of graph sampling to control the structural similarity of two graphs. This model is based on random graphs, and can be viewed as a generalization of the classical automorphism group problem for random graphs [16, chapter 9]. Finally, (3) we prove that a surprisingly simple and mild condition on the scaling of the expected degree with the number of nodes is sufficient for de-anonymization to be feasible, with strong implications on privacy.

To summarize, in this chapter we introduce a probabilistic model based on random graphs for the de-anonymization problem and prove that using such a model, and under plausible conditions, all nodes of the network can be de-anonymized with high probability. Hereafter, we elaborate on our motivation and approach. Our key result is that under surprisingly mild conditions on the model parameters, depending on the extent of the overlap between the two graphs, it is possible to establish a perfect mapping between the nodes of the two graphs as the number of nodes grows large.

Our results for approximate graph matching not only exhibit the risk of a privacy breach in the release of even the most basic information about real networks (i.e., only anonymized users and their links), but can have useful applications as well. If matching is feasible, we can combine several "noisy" public (anonymized) versions of social networks obtained from different sources into a more precise, combined network. In another scenario, suppose we have the call graph between all the phone numbers in an organization, and the graph of e-mail exchanges between e-mail addresses in this same organization. We could then establish the correspondence between phone numbers and e-mail addresses solely through the structure of the two social networks (which we expect to be similar but not exactly equal).

We should emphasize that this chapter only addresses the *feasibility* of de-anonymization. This amounts to establishing that there exists a cost function over the two graphs, such that minimizing this function finds the correct matching with high probability. We do not address the computational complexity of this process. We defer the algorithmic aspect of the matching problem, i.e., *how* to find this correct mapping, to Chapter 4.

The remainder of this chapter is organized as follows. In Section 3.1, we formally define the de-anonymization problem, introduce a mathematical model for approximate graph matching, and define the error measure used in the matching problem. Section 3.2 is the core of this chapter where we prove that in our model, perfect matching is feasible under mild conditions on the expected degree of the graphs and on their similarity. In Section 3.3 we discuss numerical experiments using social network data to justify the assumptions in our model. Finally, in Section 3.4 we conclude with a discussion of the implications of the result.

3.1 Problem Definition

We define the problem of matching the vertex sets of two graphs. To model the structural similarity of the two graphs, we use the idea of *edge sampling* introduced in Chapter 1. Furthermore, we assume that the underlying graph is a realization of the class of random graphs $G(n, p)$, and call our model $G(n, p; s)$ model, where s corresponds to the edge sampling probability. The $G(n, p; s)$ random graph model generates two similar graphs $G_{1,2}$ over the same vertex set. As mentioned before, the goal is to match the vertices of two unlabeled graphs whose edge sets are correlated but not necessarily equal. The basis for our model is its parsimony and symmetric structure, ingredients for its mathematical tractability.

As mentioned earlier, the model assumes that the observed networks $G_{1,2}$ are incomplete manifestations of a true underlying network G of relationships. For example, the edges of G might represent the true relationships between a set of people, while $G_{1,2}$ capture the observable interactions between these people, such as communications (e-mail, phone calls, proximity, and so on), or “friend”-relationships in a social network. $G_{1,2}$ might alternatively represent observations of the same network at different points in time.

To elaborate on this, let $G = (V, E)$ be a generator graph with vertex set V and edge set E. We assume here that G is an Erdös-Rényi random graph $G(n, p)$ with n nodes, where every edge exists with identical probability p, independently of all the other edges. For a *fixed realization* of $G = G(n, p)$, we generate two graphs $G_{1,2} = (V, E_{1,2})$ by sampling the vertex set E twice. More precisely, each edge $e \in E$ is in the edge set of $E_{1,2}$ with probability s, independently of everything else. As a result, the sample graphs $G_{1,2}$ are themselves Erdös-Rényi random graph $G(n, ps)$, but their edge sets are correlated, in that the existence of an edge in E_1 implies that the existence of this edge in E_2 is more likely than unconditionally (provided $p < 1$ and $s > 0$) (see Fig. 3.1). The $G(n, p)$ model has been widely used in the study of complex and social networks [15, 46, 80, 5], which makes it a plausible candidate for the study of the approximate matching problem.

Our goal is to determine whether it is possible to find the correct mapping between the nodes of G_1 and G_2, assuming we only see unlabeled versions of these two graphs (and without access to the generator G). This is equivalent to the assumption that the two graphs have different vertex label sets that contain no information about the graphs, such as random labels allocated in an anonymization procedure. Using this model, our problem can be viewed as the generalization of the classic automorphism group problem in random graphs. We discuss

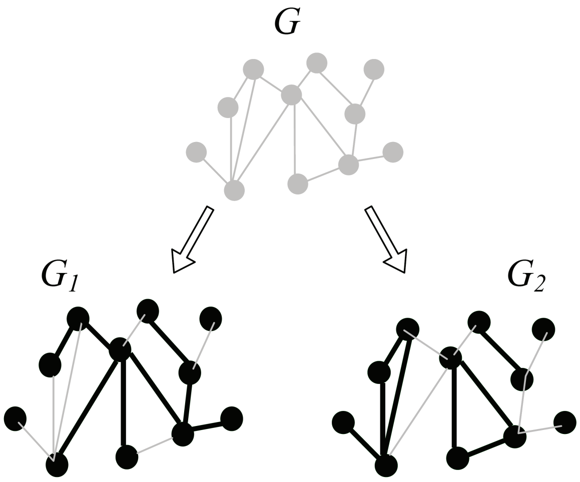


Figure 3.1: Sampling process applied to the underlying graph G, resulting in the two sampled graphs G_1 and G_2 to be matched.

this and also the effect of the choice of other graph models at the end of Section 3.2 and also in Section 3.4.

We formally define the graph matching problem as follows. We assume that $G_{1,2}$ are only available in unlabeled form (or equivalently, with two arbitrary and unknown sets of labels). Let π denote a permutation on V, i.e., one way of mapping vertices from G_1 onto G_2. The number of such permutations is $n!$. The identity permutation, denoted by π_0, is the correct mapping between the nodes of G_1 and G_2. We seek an error (cost) function over the set of permutations, which succeeds if it is uniquely minimized by π_0.

Therefore, to solve the matching problem, we are interested in showing the following:

Among all possible permutations between the two vertex sets, the identity permutation π_0 *is the permutation that minimizes a cost function, giving the node matching between the two graphs.*

The cost function should measure to what extent the structures of graphs G_1 and G_2 resemble each other under a given permutation. The structural difference can be viewed as the difference between the corresponding edge sets. This idea has also been investigated in the field of pattern recognition where the *edge-consistency* of two graph patterns (in matching a data graph to a model graph) is used to obtain the correspondence errors [74, 67].

We introduce the cost function as the error measure for edge-inconsistency, considering only the structures of two graphs $G_1(V, E_1)$ and $G_2(V, E_2)$. The matching error Δ can be generally defined as

$$\Delta_\pi = \sum_{e \in E_1} \mathbf{1}_{\{\pi(e) \notin E_2\}} + \sum_{e \in E_2} \mathbf{1}_{\{\pi^{-1}(e) \notin E_1\}}, \tag{3.1}$$

where $\mathbf{1}_{\{A\}}$ denotes the indicator function. In other words, permutation π defines a mapping between the nodes of G_1 and G_2, and Δ counts the number of edges that exist in one graph with the corresponding edges not existing in the other graph under matching π. This is the simplest error function that can be assumed for such a setting when comparing the structures of two graphs. Although this cost function not necessarily optimal (depending on the graph model) nor computationally efficient, it lend itself to probablistic analysis. It is now sufficient to find the condition under which, among all possible mappings, the identity mapping π_0 minimizes the error function. This condition is indeed a threshold on sampling probability, as we would expect the matching to be easier for highly overlapping graphs, and vice versa. Specifically, we prove below that if the sampling probability s is beyond some threshold, as n grows large, the identity permutation π_0 minimizes the error function (3.1). In other words, for s larger than this threshold, if we enumerate all permutations and assume $n!$ possible mappings for the two graphs, the permutation that gives the minimum error will be the correct mapping between the nodes.

We reiterate that in this chapter we do not address the algorithmic aspects of de-anonymization, including the computational complexity of enumerating all mappings and computing their error. Instead, we next show conditions on the model parameters such that minimizing the error function is almost surely equivalent to identifying the correct mapping using only the structures of the two sampled graphs, i.e., we show that *de-anonymization is feasible*, and it is not possible to guarantee anonymity.

3.2 Conditions for Perfect Matching

Following the model introduced in Section 3.1, we state the main theorem of this chapter, followed by its proof.

Theorem 3.2.1 *For the $G(n,p;s)$ matching problem with $s = \omega(1/n)$ and $p \to 0$, if*

$$ps \cdot s^2 = 8\frac{\log n}{n} + \omega(n^{-1}), \tag{3.2}$$

then the identity permutation π_0 minimizes the error criterion (3.1) *a.a.s*[1]*, yielding perfect matching of the vertex sets of G_1 and G_2.* [2]

Proof We denote by Δ_0 the error induced by the identity permutation and Δ_π the error induced by the permutation π. Figure 3.2 depicts two possible mappings between the same G_1 and G_2 shown in Figure 3.1 corresponding to the identity mapping π_0 and a permutation π_2 (in which all nodes are fixed except two) respectively, together with their error.

To show the result, we define Π_k on V as the set of all permutations that fix $n-k$ nodes and permute k nodes, calling them an order-k permutation. The number of such permutations, referred to as "rencontre numbers", is as follows [94]:

$$|\Pi_k| = R(n, n-k) = \binom{n}{k} \cdot (!k), \tag{3.3}$$

where $!k$ is the subfactorial of k, denoting the number of permutations of k objects in which no object appears in its natural place. It is easily verified that $R(n, n-k)$ can be upper-bounded as follows:

$$|\Pi_k| = \binom{n}{k} \cdot (!k) \leq \binom{n}{k} \cdot \left(\frac{k!}{2}\right) \leq n^k. \tag{3.4}$$

---

[1] a.a.s: asymptotically almost surely, i.e., with probability going to 1 as the number of nodes n goes to infinity. In general, *asymptotic* refers to the behavior for $n \to \infty$.

[2] We use the standard asymptotic notation (o, O, ω, Ω, and θ).

The random variables introduced below are indexed by n, which we omit unless required by the context. We define

$$S_k = \sum_{\pi \in \Pi_k} \mathbf{1}_{\{\Delta_\pi \leq \Delta_0\}}.$$

S_k counts the number of order-k permutations for which the number of matching errors is at most that of the identity permutation. Thus, $S = \sum_{k=2}^{n} S_k$ is the total number of false matches. The expected number of errors can be computed as:

$$\begin{aligned} \mathsf{E}\left[S\right] &= \sum_{k=2}^{n} \mathsf{E}\left[S_k\right] = \sum_{k=2}^{n} \sum_{\pi \in \Pi_k} \mathsf{E}\left[\mathbf{1}_{\{\Delta_\pi \leq \Delta_0\}}\right] \\ &= \sum_{k=2}^{n} \sum_{\pi \in \Pi_k} \mathsf{P}\left\{\Delta_\pi - \Delta_0 \leq 0\right\}, \end{aligned}$$

where the expectation is over $G(n, p; s)$.

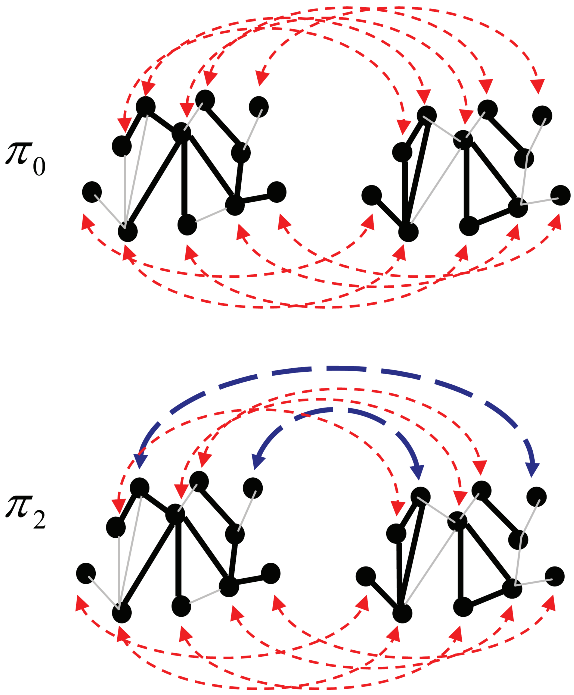


Figure 3.2: The identity permutation π_0 versus a permutation $\pi_2 \in \Pi_2$ that mismatches $k = 2$ vertices for mapping G_1 to G_2. The error in each case corresponds to the number of edges in one graph with the mapped edge not existing in the other graph. Thus, $\Delta_0 = 8$ and $\Delta_{\pi_2} = 10$, where Δ_0 is the edge difference as a result of the sampling process, and Δ_{π_2} is induced by both the sampling process and the wrong mapping of two nodes in π_2.

S counts the total number of non-identity permutations that minimize the error, and we need to show that with high probability no such permutations exist. By the First Moment Method (following Markov's inequality), since S is a non-negative integer-valued random variable, to show that $\mathsf{P}\left\{S = 0\right\} \to 1$, it suffices to show that $E[S] \to 0$.

Using this method and substituting (3.4) in the above, it is then sufficient to show that

$$\mathsf{E}\left[S\right] \leq \sum_{k=2}^{n} n^k \max_{\pi\in\Pi_k} \mathsf{P}\left\{\Delta_\pi - \Delta_0 \leq 0\right\} \to 0. \tag{3.5}$$

We bound the error probability for a fixed order-k permutation π, i.e., we bound the probability term in (3.5). For permutation π, let V_π be the set of vertices for which $v \neq \pi(v)$, and let $E_\pi = V_\pi \times V$, i.e., the set of possible edges between one or two vertices mismatched under π. Note that every edge satisfying $e \neq \pi(e)$ is in E_π. The inverse is not true, because transpositions in π (a pair (u, v) such that $\pi(u) = v$ and $\pi(v) = u$) induce invariant edges. The cardinality e_k of E_π is

$$e_k = |E_\pi| = \binom{k}{2} + k(n-k),$$

where the first term is the number of unordered node pairs both in V_π, and the second term is the number of unordered node pairs with one node in V_π.

As every edge e in the complement of E_π (i.e., in $(V \times V) - E_\pi$) is by definition invariant under π, they contribute equally to Δ_0 and Δ_π. Therefore, we can write $\Delta_\pi - \Delta_0 = X_\pi - Y_\pi$, where

$$\begin{aligned} X_\pi &= \sum_{e\in E_\pi} |\mathbf{1}_{\{e\in E_\pi^1\}} - \mathbf{1}_{\{\pi(e)\in E_\pi^2\}}|, \\ Y_\pi &= \sum_{e\in E_\pi} |\mathbf{1}_{\{e\in E_\pi^1\}} - \mathbf{1}_{\{e\in E_\pi^2\}}|, \end{aligned} \tag{3.6}$$

with $E_\pi^{1,2} = E_\pi \cap E(G_{1,2})$, i.e., the set of edges in $G_{1,2}$ incident to at least one mismatched vertex. Here, Y_π is the number of errors for the identity permutation within the set E_π, i.e., the number of sampling errors within E_π. Note that X_π and Y_π are not independent, because they are functions of the same random sets $E_\pi^{1,2}$.

Y_π counts the number of edges in E_π that are sampled in only one of $G_{1,2}$, i.e., the number of sampling errors under the identity permutation. The probability for each possible edge to be in $E(G)$ and exactly one of $G_{1,2}$ is $2ps(1-s)$. Thus Y_π is binomial with probability $2ps(1-s)$.

For X_π, we need to proceed more carefully. Assume π has $\phi \geq 0$ transpositions. First, note that each transposition in π induces one invariant edge $e = \pi(e) = \pi^{-1}(e)$ in E_π (such an edge contributes to X_π with probability $2ps(1-s)$).

The remaining $e_k - \phi$ edges are not invariant under π. Each pair of such edges $(e, \pi(e))$ contributes 1 to X_π if $e \in G_1$ and $\pi(e) \notin G_2$ or vice versa (cf. (3.6)). The probability for exactly one of two *different* edges in E_π to be sampled is $2ps(1-ps)$. Note that the terms in (3.6) are dependent, because conditional on $|\mathbf{1}_{\{e\in E_\pi^1\}} - \mathbf{1}_{\{\pi(e)\in E_\pi^2\}}| = 1$, at least one of e or $\pi(e)$ is present in the generator G. Thus, the conditional probability of an adjacent pair (either $(\pi^{-1}(e), e)$ or $(\pi(e), \pi(\pi(e)))$ contributing 1 to (3.6) is $s(1-ps)$. We conservatively ignore this positive correlation and stochastically lower-bound X_π by assuming that each pair of edges $(e, \pi(e))$ contributes an i.i.d. Bernoulli with parameter $2ps(1-ps)$ to (3.6).

Thus, X_π is stochastically lower-bounded by the sum of two independent binomials $\mathsf{Bi}\left(e_k - \phi, 2ps(1-ps)\right) + \mathsf{Bi}\left(\phi, 2ps(1-s)\right)$, where ϕ is the number of transpositions in π. By definition, a transposition can occur only between two vertices that are both in V_π. Hence, $\phi \leq \lfloor k/2 \rfloor \leq k/2$.

Thus, we have

$$\begin{aligned} X_\pi &\overset{(stoch.)}{\geq} \mathsf{Bi}\left(e_k - \lfloor k/2 \rfloor, 2ps(1-ps)\right) &(3.7)\\ Y_\pi &\sim \mathsf{Bi}\left(e_k, 2ps(1-s)\right). &(3.8)\end{aligned}$$

We upper-bound the probability of the event $\{X_\pi - Y_\pi \leq 0\}$ using the following lemma, which holds regardless of dependence between X_π and Y_π:

Lemma 3.2.1 *Let X_1 and X_2 be two binomial random variables with means λ_1 and λ_2, where $\lambda_2 > \lambda_1$. Then,*

$$P\{X_2 - X_1 \leq 0\} \leq 2\exp\left(-\frac{1}{8}\frac{(\lambda_2-\lambda_1)^2}{\lambda_2}\right). \tag{3.9}$$

Proof Let X_1 and X_2 be two binomial random variables with means λ_1 and λ_2. The probability of the event $\{X_2 - X_1 \leq 0\}$ can be upper-bounded as follows:

$$\mathsf{P}\left\{X_2 - X_1 \leq 0\right\} \leq \mathsf{P}\left\{X_1 \geq x\right\} + \mathsf{P}\left\{X_2 \leq x\right\}, \tag{3.10}$$

for any x.

We now find an upper-bound for the right-hand side of (3.10). We use the Chernoff bounds for the binomial random variables X_1 and X_2 using the following theorem [44]:

If $X \in Bi(n,p)$ and $\lambda = np$, then,

$$\begin{aligned} P\{X > \lambda + t\} &\leq \exp\left(-\frac{t^2}{2\left(\lambda + t/3\right)}\right), \quad t \geq 0; &(3.11)\\ P\{X < \lambda - t\} &\leq \exp\left(-\frac{t^2}{2\lambda}\right), \quad t \geq 0. &(3.12)\end{aligned}$$

We upper-bound $\mathsf{P}\left\{X_1 \geq x\right\}$ and $\mathsf{P}\left\{X_2 \leq x\right\}$ using (3.11) and (3.12) (for two arbitrary positive values of t_1 and t_2 respectively). We set $x = (\lambda_1 + \lambda_2)/2$, and thus $t_1 = t_2 = (\lambda_2 - \lambda_1)/2$. Using $\lambda_2 > \lambda_1$ allows to bound the two exponents as follows:

$$\begin{aligned} \mathsf{P}\left\{X_1 \geq x\right\} &\leq \exp\left(-\frac{1}{8}\frac{(\lambda_2-\lambda_1)^2}{\lambda_1 + (\lambda_2-\lambda_1)/6}\right)\\ &\leq \exp\left(-\frac{1}{8}\frac{(\lambda_2-\lambda_1)^2}{\lambda_2}\right), &(3.13)\end{aligned}$$

and

$$\mathsf{P}\left\{X_2 \leq x\right\} \leq \exp\left(-\frac{1}{8}\frac{(\lambda_2-\lambda_1)^2}{\lambda_2}\right). \tag{3.14}$$

This completes the proof. ■

Now let λ_π and λ_0 denote the means of X_π and Y_π respectively, with values,

$$\begin{aligned} \lambda_\pi &= 2ps(1-ps)(e_k - k/2) &(3.15)\\ \lambda_0 &= 2pse_k(1-s). &(3.16)\end{aligned}$$

Since $0 \leq s \leq 1$, $2 \leq k \leq n$, and $e_k \simeq k(n-k/2)$, to satisfy $\lambda_\pi > \lambda_0$ we need to have,

$$\begin{aligned} 2ps(1-ps)(k(n-k/2)-k/2) &> 2psk(n-k/2)(1-s) \\ \implies s &> \left(\tfrac{1-ps}{1-p}\right) \frac{1}{2n-k}, \end{aligned} \tag{3.17}$$

which will be satisfied for $s = \omega(1/n)$ and $p \to 0$.

Thus, using the above lemma, we obtain,

$$\mathsf{P}\left\{X_\pi - Y_\pi \leq 0\right\} \leq 2\exp\left(-\underbrace{\frac{1}{8}\frac{(\lambda_\pi - \lambda_0)^2}{\lambda_\pi}}_{f(n,p,k)}\right). \tag{3.18}$$

Since $ps \to 0$, substituting (3.15) and (3.16) in (3.18) yields:

$$\begin{aligned} f(n,p,k) &= \frac{1}{8}\frac{\left(2ps\left((e_k - k/2) - (e_k - e_k s)\right)\right)^2}{2ps\left(e_k - k/2\right)} \\ &= \frac{ps}{4}\frac{\left((k/2)\left((2n-k)s-1\right)\right)^2}{(k/2)\left(2n-k-1\right)} \end{aligned}$$

For $s = \omega(1/n)$ we have $(2n-k)s = \omega(1)$. Thus,

$$\begin{aligned} f(n,p,k) &\simeq \frac{ps}{4}\frac{(k/2)\left((2n-k)s\right)^2}{(2n-k)} \\ &\simeq \frac{ps}{4}\, s^2\, k\left(n - k/2\right). \end{aligned} \tag{3.19}$$

Using (3.4), (3.5), (3.18) and (3.19), we have,

$$\begin{aligned} \mathsf{E}\left[S\right] &\leq 2\sum_{k=2}^{n} n^k \cdot \exp(-f(n,p,k)) \\ &\overset{(a)}{\simeq} 2\sum_{k=2}^{n} n^k \exp\left(-k\left(n-\frac{k}{2}\right)\frac{ps}{4}\cdot s^2\right) \\ &\overset{(b)}{\leq} 2\sum_{k=2}^{\infty} \exp\left(k\left(\log n - \frac{nps}{8}\cdot s^2\right)\right), \end{aligned} \tag{3.20}$$

where (a) is derived using (3.19), and (b) uses $k \leq n$. The geometric series goes to zero if the first term goes to zero, which is implied by the condition in the statement of the theorem. This completes the proof. ■

A more direct approach to prove the result would be to try to condition on a property of the underlying graph G and/or of $G_{1,2}$ that is both asymptotically almost sure, and for which one could show that uniformly over all permutations π, the number of errors is higher than for the identity π_0. It is difficult to identify such a property that would make the second part of the problem tractable. Instead, we show the result using a method commonly employed in the random graph literature [16, 44], which allows us to analyze a fixed permutation π over the full probability space $G(n,p;s)$.

A remarkable aspect of our result is that for fixed similarity parameter s, the condition is $ps = 8c \log n/n$ for some $c(s) > 1$. As expected, $c(s) = 1/s^2$ is monotonically decreasing in $(0, 1)$, and $c(1) = 1$. Thus, for an overall edge sampling probability ps of a bit larger than $8 \log n/n$, with high probability the identity permutation minimizes the error function and yields the correct mapping. Note that the threshold for the connectivity of $G_{1,2} = G(n, ps)$ (and for the disappearance of isolated vertices) is $ps = \log n/n$ [16, 44]. It is obvious that it is impossible to perfectly match a pair of graphs $G_{1,2}$ when at least one of them possesses more than one isolated vertex (as these necessarily give rise to multiple permutations with equal error counts). Therefore, $ps = \log n/n$ is a lower bound for zero-error graph matching using any technique (i.e., any cost function). Our bound for $G(n, p; s)$ matching is therefore tight, up to a constant function of s.

For the case of $s = 1$, the approximate graph matching problem is equivalent to the classical automorphism group problem for random graphs [16]. Specifically, it is known that $G(n, p)$ is asymmetric (has an automorphism group of size one) for $p = \log n/n + \omega(1)$. This suggests that the constant $c(s)$ in our result can be improved upon through more refined bounding techniques. Indeed, we use relatively loose bounds in several places: in particular, we underestimate the mean of X_π quite significantly by ignoring the positive correlation (within each cycle of π) in the terms of (3.6); also, we assume the worst-case dependence between X_π and Y_π in (3.10), even though they are in reality positively correlated through the generator G. These bounds are sufficient to show the asymptotic result to within a constant, but more precise techniques akin to those used to show the classical automorphism result may allow to go further. Another obvious extension of our work would employ other generator graph structures such as random regular graphs, small world models, or scale free graphs.

In the following, we further strengthen our results for $G(n, p)$, allowing a small node mismatch to occur. We show that allowing a small error in matching nodes contributes to a lower threshold for sampling probability.

3.2.1 Allowing a Fraction of Node Mismatches

In this section, we address the same problem, with the new assumption of letting an ϵ-fraction of nodes to be mismatched, where epsilon is an arbitrary positive constant less than 1. We show that tolerating this small error in matching the two graphs, a looser bound (or lower threshold) for the sampling probability will be achieved. In the following, we show that in this case, it is sufficient for the overall sampling probability ps to be above the threshold of $8 \log n/n$ by $\omega(n^{-2})$, so that the identity permutation yields the best matching.

Corollary 3.2.1 *Consider the random graph $G(n, p)$, and sample its edges twice independently with probability s to obtain two sampled versions G_1 and G_2. For*

$$ps \cdot s^2 = 8 \frac{\log n}{n} + \omega(n^{-1}),$$

the identity permutation π_0 minimizes the error criterion (3.1) a.a.s., if we allow an ϵ-fraction of the nodes to be mismatched, where ϵ is an arbitrary constant between zero and 1.

Proof We now tolerate an ϵ-fraction mismatch in the nodes. Thus, for an order-k permutation π_k which permutes k nodes, we have $k/n < \epsilon$. So for the permutations contributing to the matching error, k should be larger than ϵn. This requires the same analysis as above, with the only difference that in the sum in (3.20), k starts from ϵn, which yields:

$$\begin{aligned}
\mathsf{E}\left[S\right] &\leq \sum_{k=\epsilon n}^{n} (n)_k \mathsf{P}\left\{X_\pi - X_0 < 0\right\} \\
&\leq 2\sum_{k=\epsilon n}^{n} n^k \exp\left(-k\left(n-\frac{k}{2}\right)\frac{ps}{4}\cdot s^2\right) \\
&\leq 2\sum_{k=\epsilon n}^{\infty} \exp\left(k\left(\log n - \frac{nps}{8}\cdot s^2\right)\right) \\
&\Longrightarrow \exp\left(\frac{\epsilon n^2}{8}\left(\frac{8\log n}{n} - ps\ s^2\right)\right) \to 0 \qquad (3.21)
\end{aligned}$$

The term above goes to zero if the condition of the corollary holds. ■

Note that the result does not depend on ϵ, and holds regardless of the fraction of nodes chosen. The above result is worth considering when compared with Theorem 3.2.1. Here we show that allowing a positive fraction of nodes to be mismatched, the threshold for matching becomes lower ($\omega(n^{-2})$ rather than $\omega(n^{-1})$). In other words, a small error in matching yields an improvement to the threshold required for sampling probability. However, this improvement is not significant as it does not change the dominant factor ($8\log n/n$).

3.2.2 Different sampling rates

Hereby, we generalize the sampling model, allowing G_1 and G_2 to be edge-sampled with different sampling rates s_1 and s_2. We would call this a $G(n,p;s_1,s_2)$ model. This would cover more realistic graphs, e.g., as in observing G_1 and G_2 from an underlying graph over different time spans (both being sampled from the same graph), resulting in one graph being denser than the other (different average degrees). We show that we can still find the conditions for perfect matching, and we derive a more general result depending on both sampling probabilities.

Theorem 3.2.2 *For the $G(n,p;s_1,s_2)$ matching problem with $\frac{s_1 s_2}{s_1+s_2} = \omega(1/n)$ and $p \to 0$, if*

$$p\left(\frac{s_1+s_2}{2}\right)\left(\frac{2s_1s_2}{s_1+s_2}\right)^2 = 8\frac{\log n}{n} + \omega(n^{-1}), \qquad (3.22)$$

then the identity permutation π_0 minimizes the error criterion a.a.s, yielding perfect matching of the vertex sets of G_1 and G_2.

Proof The proof would be very similar to the proof of (3.2.1). The only difference arises when estimating the random variables X_π and Y_π, since the probability of the possible edges would now depend on s_1 in G_1, and s_2 in G_2, respectively.

More specifically, the probability for each possible edge to be in $E(G)$ and exactly one of $G_{1,2}$ would be $ps_1(1-s_2)+ps_2(1-s_1)$. Thus Y_π is binomial with probability $p(s_1+s_2-2s_1s_2)$. For X_π, we should again consider the transposition, assuming $\phi \geq 0$ transpositions exist. Each transposition in π induces one invariant edge $e = \pi(e) = \pi^{-1}(e)$ in E_π, and such an edge contributes to X_π with probability $ps_1(1-s_2)+ps_2(1-s_1)$ in this case. As before, the remaining $e_k - \phi$ edges contribute to X_π, now with probability $ps_1(1-ps_2)+ps_2(1-ps_1)$, and we ignore the positive correlation of the edges.

Thus, X_π is stochastically lower-bounded by the sum of two independent binomials $\mathsf{Bi}\left(e_k - \phi, p(s_1+s_2-2ps_1s_2)\right) + \mathsf{Bi}\left(\phi, p(s_1+s_2-2s_1s_2)\right)$, where ϕ is the number of transpositions in π. We lower-bound ϕ by $k/2$ and have,

$$\begin{aligned} X_\pi &\overset{(stoch.)}{\geq} \mathsf{Bi}\left(e_k - \lfloor k/2 \rfloor, p(s_1+s_2-2ps_1s_2)\right) && (3.23)\\ Y_\pi &\sim \mathsf{Bi}\left(e_k, p(s_1+s_2-2s_1s_2)\right). && (3.24)\end{aligned}$$

The means of X_π and Y_π are as follows:

$$\begin{aligned} \lambda_\pi &= p(s_1+s_2-2ps_1s_2)(e_k-k/2) && (3.25)\\ \lambda_0 &= p(s_1+s_2-2s_1s_2)e_k. && (3.26)\end{aligned}$$

We will now use Lemma (3.2.1). Since $0 \leq s_1, s_2 \leq 1$, $2 \leq k \leq n$, and $e_k \simeq k(n-k/2)$, to satisfy $\lambda_\pi > \lambda_0$ we need to have,

$$\begin{aligned} p(s_1+s_2-2ps_1s_2)(k(n-k/2)-k/2) &> p(s_1+s_2-2s_1s_2)k(n-k/2)\\ \Longrightarrow \frac{s_1s_2}{s_1+s_2} &> \frac{1}{2\left((2n-k)(1-p)+p\right)}, && (3.27)\end{aligned}$$

which will be satisfied for $\frac{s_1s_2}{s_1+s_2} = \omega(1/n)$ and $p \to 0$.

Thus, using Lemma (3.2.1), we obtain,

$$\mathsf{P}\left\{X_\pi - Y_\pi \leq 0\right\} \leq 2\exp\left(-\underbrace{\frac{1}{8}\frac{(\lambda_\pi-\lambda_0)^2}{\lambda_\pi}}_{f(n,p,k)}\right). \qquad (3.28)$$

where,

$$\begin{aligned} f(n,p,k) &= \frac{1}{8}\frac{\left(p\left((s_1+s_2)(e_k-k/2-e_k)+2s_1s_2e_k\right)\right)^2}{p(s_1+s_2)(e_k-k/2)}\\ &= \frac{p(s_1+s_2)}{8}\cdot\frac{\left((k/2)\left((2n-k)\left(\frac{2s_1s_2}{s_1+s_2}\right)-1\right)\right)^2}{(k/2)\,(2n-k-1)}\end{aligned}$$

For $\frac{s_1s_2}{s_1+s_2} = \omega(1/n)$, we have $(2n-k)\left(\frac{s_1s_2}{s_1+s_2}\right) = \omega(1)$. Thus,

$$\begin{aligned} f(n,p,k) &\simeq \frac{p(s_1+s_2)}{8}\frac{(k/2)\left((2n-k)\left(\frac{2s_1s_2}{s_1+s_2}\right)\right)^2}{(2n-k)}\\ &\simeq p\left(\frac{s_1+s_2}{2}\right)\left(\frac{s_1s_2}{s_1+s_2}\right)^2 k\,(n-k/2). && (3.29)\end{aligned}$$

Finally, using (3.4), (3.5), and (3.29), we have,

$$\begin{aligned}
\mathsf{E}[S] &\leq 2\sum_{k=2}^{n} n^k \cdot \exp(-f(n,p,k)) \\
&\overset{(a)}{\simeq} 2\sum_{k=2}^{n} n^k \exp\left(-k\left(n-\frac{k}{2}\right)p\left(\frac{s_1+s_2}{2}\right)\left(\frac{s_1s_2}{s_1+s_2}\right)^2\right) \\
&\overset{(b)}{\leq} 2\sum_{k=2}^{\infty} \exp\left(k\left(\log n - \frac{np(s_1+s_2)}{4}\cdot\left(\frac{s_1s_2}{s_1+s_2}\right)^2\right)\right), \qquad (3.30)
\end{aligned}$$

where (a) is derived using (3.29), and (b) uses $k \leq n$. The geometric series goes to zero if the first term goes to zero, which is implied by the condition in the statement of the theorem. This completes the proof. ■

The above result establishes the condition for the average sampling rate of the two graphs, i.e., $p(s_1+s_2)/2$, so that perfect matching is feasible. Note that for $s_1 = s_2 = s$, the same result as in (3.2.1) holds.

3.3 On the Similarity of Real Networks

To be mathematically tractable and parsimonious, our model inevitably embodies several strong assumptions: (i) the underlying graph is a $G = G(n,p)$ random graph, and the edge sets of the "visible" graphs $G_{1,2}$ are sampled (ii) independently and (iii) with identical probability s from G. Despite these assumptions, we believe that our model and our result on anonymity conditions have implications for real networks and scenarios. Although we are unable to explore the validity of our assumptions in full generality, we wish at least to provide some evidence to justify them. First, it is fairly clear that the underlying graph of a social network would possess a structure very different from a $G(n,p)$, as demonstrated in many studies illustrating fascinating properties such as skewed degree distributions and the small-world effect. However, we conjecture that de-anonymizing two networks sampled from a random graph is harder than more "structured" networks. A random graph is in some sense "maximally uniform", and we therefore believe that for other, more realistic hidden graphs G, de-anonymization might in fact be possible under even weaker conditions. This is of course a promising and fascinating area for further research. Second, we consider de-anonymization successful if the error function $\Delta(.)$ has a unique minimum at π_0. We argue that this function is not too sensitive to a non-uniform sampling process over the edge set (i.e., assuming each edge e is sampled with its own sampling rate $s(e)$), provided the sampling process is similar in both graphs, and uncorrelated across edges. This is because the impact of this non-uniformity on X_0 and X_π above would cancel out to a certain extent. On the other hand, if the sampling process to obtain the two samples G_1 and G_2 were very different, then this could make de-anonymization much harder. For example, if the sampling rate over some subset of vertices were atypically large in G_1 compared to the rest, but atypically large for a *different* subset in G_2, then these two high-rate subsets would be likely to be falsely matched. Therefore, it appears that de-anonymization would be quite sensitive to such differences in the sampling process for G_1 and G_2.

While we have not quantified the above argument, it did lead us to explore the stability of the sampling process through some numerical experiments. In order to motivate and illustrate

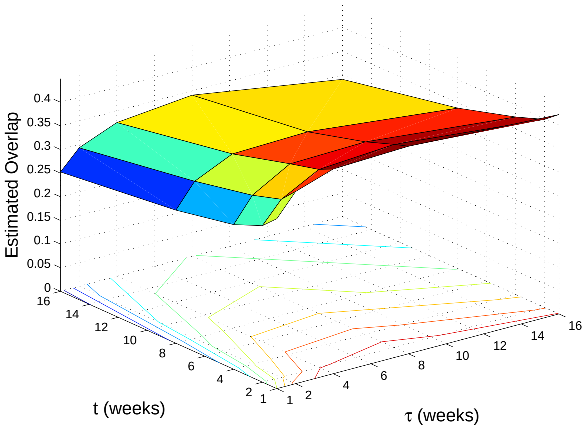


Figure 3.3: Estimated average edge overlap among overlapped nodes for EPFL internal network, as a function of window size and distance.

the concept of *similarity* between networks, and also verify the assumption of *independence* in sampling the edges, we present an example of a real social network: an e-mail graph, in which nodes represent e-mail addresses and edges represent message exchanges. The network evolves in time through the observation of new messages that are exchanged.

We consider a dataset of e-mail messages collected at the mail server of EPFL. The dataset includes logs of e-mail exchanges among users on a weekly basis for a period of 75 weeks. In our dataset, the e-mail exchanges *among EPFL users* is considered (i.e., internal EPFL network). The dataset includes snapshots of the network aggregated by week, such that timestamps are in the timescale of weeks (i.e., all messages sent in a particular week have the same timestamp). Using such dataset, we construct the e-mail network of each week for the internal EPFL network.

Having introduced the above, we investigate the similarity between different snapshots of the network, each being a sample of an underlying hidden e-mail network. Note that in order to map real data to our sampling model, the existence of a hidden underlying graph (including all possible e-mail exchanges over all times) is inevitable - to which we do not have access. However, measuring the amount of *edge overlap* between different snapshots gives us an estimation of the similarity degree between different network samples, or whether the graphs are the outcome of similar sampling processes. Also, since two network snapshots do not contain the same number of nodes necessarily, we estimate the edge overlap as *the proportion of edges among overlapped nodes that exist in both graphs.*

To accomplish the above, we need to pick two networks to be compared. We randomly choose a starting timestamp t_s (week number) in the entire dataset, and construct the first graph starting from t_s accumulated over a window size of τ weeks. For the second graph, we build it starting from timestamp $t_s + \tau + t - 1$, again accumulated over a window size of τ, where t denotes the time distance between the two graphs (in weeks). In other words, τ corresponds to the density of the graph (the larger it is, the denser the graph will be), and t implies the time distance between different samples. As an example, $\tau = 1, t = 1$ corresponds to the e-mail network of two consecutive snapshots (each consisting of e-mail exchanges over a one-week period), whereas $\tau = 2, t = 3$ corresponds to two graphs, each consisting of e-mail exchanges over a period of $\tau = 2$ weeks, with a time distance of $t = 3$ weeks. Finally, for

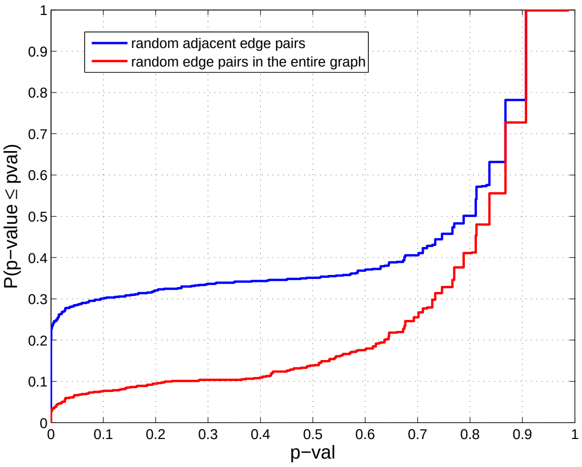


Figure 3.4: The CDF of the p-values of the Pearson's Chi-Square test for independence, for 1) random adjacent edge pairs (top curve), 2) random edge pairs in the entire graph (bottom curve). Using $\alpha = 0.05$, the test verifies the statistical equivalence of edge pairs in the EPFL dataset.

each value of τ and t, we repeat the random choice of the networks 30 times and compute the average.

Figure 3.3 depicts the estimated average edge overlap as a function of the windows size and time distance. It can be observed that the estimated edge overlap is quite significant, and it also exhibits a small increase as τ increases and t decreases, which matches intuition since it is expected that two larger and denser networks have more overlap, and as the samples are farther apart the overlap decreases. However, this change is small over a wide span of the density and distance values. Thus, the graph similarity is fairly robust over different densities and distances. The experiment shows that two graphs sampled from a hidden underlying graph (a hidden overall e-mail network in this case) are similar in structure (even if the sampling processes are non-uniform), with the sampling process being quite stable over different intervals.

Finally, we verify the assumption of independent edge sampling in our model, through looking at the correlation among the edges. In general, the emergence of an edge might correlate with the existence of other edges. In order to investigate how far the independence assumption is from reality, we examine edge correlation in the EPFL internal network. To do so, we choose a random pair of edges from the final accumulated graph (i.e., $\tau = 75$), and examine their joint appearance in 75 weekly snapshots. We use the Chi-Square test for independence to determine whether there is a significant relationship between the appearance of the two edges. We assume a null hypothesis that two randomly chosen edges e_1 and e_2 appear independently, and use the Pearson χ^2 test to decide whether we should reject the null hypothesis, separately for each set of 75 edge pair appearances. We compute the χ^2 test statistics of each sample set of 75 weeks as $X^2 = \sum \frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}$, $i, j = 0, 1$, where i denotes the existence (1) or non-existence (0) mode of e_1 (similarly $j = 0, 1$ for e_2), $O_{i,j}$ is the observed frequency count of e_1 at mode i and e_2 at mode j, and $E_{i,j} = n_i * n_j/n$, n_i being the total number of sampled observations of e_1 at mode i (similarly n_j for e_2 at mode j) and n being the total number of samples (75). The p-value is calculated as $P\left\{X^2 \leq \chi^2(1)\right\}$,

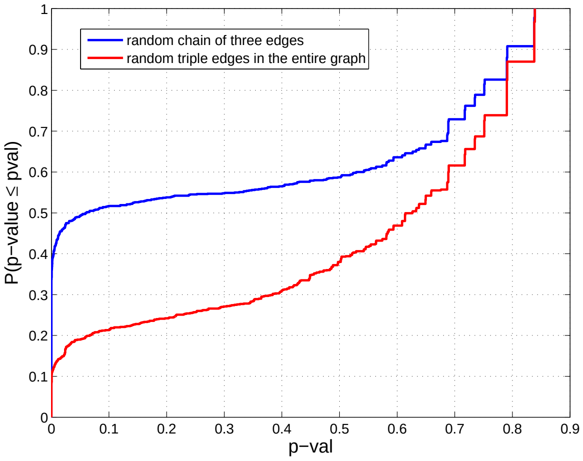


Figure 3.5: The CDF of the p-values of the Pearson's Chi-Square test for independence, for 1) random chain of three edges (top curve), 2) random triple edges in the entire graph (bottom curve). Using $\alpha = 0.05$, the test verifies the statistical equivalence of triple edges in the EPFL dataset.

where $\chi^2(1)$ is a Chi-Square random variable with one degree of freedom, as the number of bins for each categorial variable equals 2. We derive the p-value of the test and reject the independence hypothesis if the p-value is smaller than the significance level ($\alpha = 0.05$ in our tests). Repeating this for a large number of random edge pairs (800 in our experiments), we find that 93% of the edge pairs are statistically indistinguishable (p-value $> \alpha = 0.05$).

To strengthen our test even further, we do the same experiment above, by choosing random pairs of *adjacent* edges - i.e., edges incident to the same node - thinking that such edges might express a high correlation. We find that even in this case, most edge pairs (72%) are statistically independent. Figure 3.4 depicts the CDF of p-values found for each selected pair over 75 weeks, for both experiments. The plot clearly shows that in most cases, p-value is greater than α, as mentioned above.

Finally, we repeat the above experiments for triple edges , i.e. choosing three random edges in the accumulated graph, the null hypothesis being that three randomly chosen edges e_1, e_2 and e_3 appear independently. Again, we consider two cases, one where the edges are chosen randomly in the entire graph, and the other further correlated version where the edges are sampled from the set of 3-chains in the graph, i.e. paths of length 3. Figure 3.5 depicts the CDF of the p-values for each selected triple over 75 weeks, for both experiments, repeated 1000 times. It is observed that 81% of the random triple edges are independent. Further, for the correlated chain of edges, we observe that 50% of the random 3-chains are statistically indistinguishable. Further experiments show that as the number of randomly chosen edges increases, there will be a higher dependence for their joint appearance, as expected.

Our results suggest that the independence assumption clearly does not hold generally, but many small sets of edges do behave independently. To what extent the i.i.d. assumption built into our model is realistic, in the sense that it would correctly predict the boundary of privacy in real networks, is a subject of further investigation.

3.4 Discussion and Summary

In this chapter, we considered the privacy issue in social networks and investigated the possibility of de-anonymization from a mathematical perspective. We defined the problem in the context of approximate graph matching, with the goal of finding the correct mapping between the node sets of two structurally similar graphs. Using ideas from graph sampling in modeling evolution of networks, we proposed a probabilistic model to derive two sampled versions of an underlying graph as "noisy" versions of the networks to be matched. Elaborating our model for the case of random graphs, we proved that using the simplest matching criterion based only on network topology, a perfect matching between the nodes can be established with high probability as the network size grows large, under simple conditions for the sampling process. More specifically, we proved that a surprisingly mild condition on the scaling of the expected degree with the number of nodes is sufficient for de-anonymization to be feasible. For this, we expressed lower bounds for the sampling probability, or more intuitively, the extent of overlap in the edges of two graphs, so that it yields perfect matching. We also proved that a smaller threshold for sampling probability is required in the case of a fraction of node mismatches, which further strengthens our results.

Two conditions in our theorem are $s = \omega(1/n)$ and $ps \to 0$. How these parameters relate to real networks is of course a crucial and interesting question. Social networks tend to be sparse ($p \to 0$), and a reasonable assumption may be to assume a fixed average node degree ($p = c/n$), as the number of contacts is usually the result of local interactions that should not be influenced by the rest of the network[3]. The scaling of s is more debatable, as it depends on the nature of the two networks. If G_1 and G_2 capture the social interactions between a set of people using different methods (e.g., e-mail and phone calls), then it would make sense to postulate a constant s independent of the size of the network, as the choice of method (i.e., generating a link) would be a purely local one, and therefore not influenced by the rest of the network. However, more cross-domain data should be studied to verify this.

Our result shows that given a specific cost function $\Delta(.)$, a pair of correlated graphs can be perfectly matched under certain conditions. An interesting question would be the converse: can we find conditions such that no cost function could give a match? In the $G(n, p; s)$ model, it is straightforward to show such a converse of the form $ps = o(\log n/n)$, as alluded to before. In this case, G_1 and G_2 would have isolated vertices a.a.s., and obviously no method would be able to determine the correct matching among these. More precise converses, as well as variations of our model (e.g., assuming other generator graphs G) are the topic of future work.

Our work implies the feasibility of de-anonymization of a target network by using the structural similarity of a known auxiliary network, and raises privacy concerns about sharing the simplest topological information of users with partners and third-party applications. One consequence of our work might be guidelines on how to release or share only sampled versions of networks, by enforcing the sparsity constraint to guarantee anonymity. This would be promising provided such a thinned-out network would still provide enough information for the task at hand.

In future, we intend to generalize our approach to a broader class of graphs. As discussed above, we conjecture that in some sense, a random graph as the generator G may be more

---

[3] Note however that recent work on network densification [62] observes a dependence of the form $p = cn^{\alpha-2}$, for some $\alpha > 1$, i.e., an average degree that grows as $n^{\alpha-1}$ with the size of the network. While the underlying causes of densification are still being debated [84], it is still the case that $p \to 0$. Note that for fixed s, a network densifying in this way would always be non-anonymous, as $n^{\alpha-2} = \omega(\log n/n)$.

difficult than a more “structured” graph. On the other hand, the i.i.d. sampling process in our model is an idealistic assumption, and the impact of relaxing it should be explored. Finally, and perhaps most importantly, while this work proves the *existence* of the perfect matching using the proposed error function, the algorithmic complexity of searching in such a vast space should be explored, which is addressed in the following chapter.

Publication [86]

Pedram Pedarsani and Matthias Grossglauser. **On the Privacy of Anonymized Networks**. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD ’11, pages 1235–1243, 2011.

4 An Iterative Bayesian Method for Graph Matching

This chapter explores the algorithmic aspect of the approximate graph matching problem described in Chapter 3. We demonstrated that by only using the structures of two similar networks, it is *possible* to infer the correct mapping between the nodes, under mild conditions on their similarity and the graph parameters. Our proof assumed an adversary with infinite computational power, who enumerates all possible $n!$ mappings, computes a cost function for each mapping, and reports back the mapping with minimum cost as the correct mapping. We showed that for a random graph model with large number of nodes, this mapping with high probability corresponds to the correct mapping.

However, although the approach above proves the *feasibility* of matching two similar graphs, in practice we need methods for *finding* this correct mapping *efficiently.* In other words, so far, we have shown that network anonymity cannot guarantee privacy. However, we still have not demonstrated a method for building the actual mapping between the nodes of the two graphs. Notice that even if the structures of the graphs are exactly the same, the matching problem is still intractable in practice, as it is similar to the classical graph isomorphism problem.

The graph isomorphism problem is NP, and matching two similar but not necessarily identical graphs is at least as hard. This suggests that we have no hope of matching large graphs. Fortunately, it turns out that many realistic graphs have features that help graph matching. For example, in pioneering work [78], Narayanan and Shmatikov devise a graph matching algorithm that succeeds in de-anonymizing a social network with millions of nodes, based on an initial *seed set* of mapped nodes. Essentially, such methods exploit label-independent features of nodes that make these nodes easy to identify in both graphs. Such features include the node degree, the clustering coefficient, or the volume (number of edges) in a neighborhood around the node.

The algorithm in [78] starts from an initial seed set of mapped node pairs, which may be obtained manually or through heuristics (e.g., identifying cliques in both graphs). The algorithm then attempts to identify pairs of unmapped nodes that are neighbors of several seed nodes; among these pairs, it chooses the one that is best according to a heuristic, and adds it to the seed set. The algorithm then proceeds iteratively until the whole network is matched, or the process dies out. This requires (i) a seed set of sufficient size[1], and (ii) a sufficiently dense graph for the process to percolate.

In this work, we propose an algorithm for approximate graph matching that relies on a rigorous statistical model to infer the node map. The main advantages of this algorithm are the following:

---

[1] Note that in their algorithm, if the seed nodes are all at distance ≥ 3 of each other, the process cannot start, as there is no candidate node that has more than one mapped neighbor.

- **Starting from scratch:** The algorithm does not require an initial seed set of mapped node pairs as input. The same statistical framework is used to build the map from start to finish.

- **Sparse graphs:** Our approach is particularly well suited to match sparse graphs. The algorithm is able to make progress even if the mapped nodes are far from each other.

- **Confidence metric:** Our method maintains an estimate of the likelihood of every mapped node pair. While this is used by the algorithm to build the map, it is also a very useful output for some applications, as it essentially provides a level of confidence for each mapped pair.

Our statistical framework and associated inference algorithm is based on the observation that in many real networks, some nodes possess features that make them stand out from the majority of other nodes; these nodes are much easier to map correctly than an average node. This suggests an incremental mapping process, where we first try to map the easiest nodes. Each node has some fingerprint capturing label-independent metrics of interest. The matching relies on a cost function over fingerprints. Of course, errors are always possible, given that the two graphs are similar, but not identical. We deal with this by matching several node pairs at once through a maximum bipartite matching, rather than matching one node pair at a time. This is because the former takes into account the costs between all possible pairs in the set. Each iteration of the algorithm matches a set of nodes, whose size doubles in each iteration.

The second key contribution concerns the choice of fingerprint and the cost function used in the bipartite matching. We develop a probabilistic generative model for the two graphs that we are trying to match. This allows us to embed the matching problem into a clean Bayesian framework, which leads to a cost function over node pairs which can be interpreted as the log-likelihood of a correct match given the fingerprints of the two nodes.

This chapter is organized as follows. In Section 4.1, we formally define the problem. Section 4.2 describes the Bayesian framework underlying our matching algorithm, which is developed in Section 4.3. Section 4.4 reports experimental results over a social network mined from e-mail exchanges. Section 4.5 contains a short conclusion.

4.1 Problem Definition

We use a similar problem setting as in Chapter 3. We consider the problem of matching the vertex sets of two graphs. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ denote two graphs with the same number of nodes, $|V_1| = |V_2| = n$, but different edge sets. Let $\pi : V_1 \to V_2$ be a mapping that establishes a one-to-one correspondence between nodes of G_1 and G_2. Assume there exists a correct mapping π_0 between $u_1 \in V_1$ and $u_2 \in V_2$. The problem we consider is determining π_0 using as input only $G_{1,2}$ and no other labeling information.

Finding the correct mapping strongly depends on its relationship to the structure of $G_{1,2}$. For example, the correct mapping cannot be attained if one of the edge sets has no (or very few) edges, since no structural information remains to be explored. Moreover, the problem is hard even if the two graphs are structurally the same, since this reduces to finding an isomorphism between the two graphs, a well-known NP problem.

For ease of notation, and without loss of generality, we assume from now on that $V_1 = V_2 = V$, and that the true mapping is given by the identity. However, it is important to note that this notational assumption is not visible to the matching algorithm.

4.2 General Method for Graph Matching

As shown in Chapter 3, conceptually, matching two graphs can be cast as minimizing a cost function over all $n!$ possible matchings π of the vertex sets of the two graphs $G_{1,2}$ [86]. As the computational cost is prohibitive for graphs of non-trivial size, we seek a method to reduce the search space for π. We achieve this by building π in $\log_2 n$ phases. In each phase $\tau = 0, \ldots, \log_2 n - 1$, we double the number of mapped nodes, which we refer to as *anchors*, from the previous phase, until we have mapped every node. We let S_τ denote the set of anchors produced at the end of phase τ. As will become clear, this map is not necessarily monotonically increasing, in that $S_\tau \subset S_{\tau+1}$ need not be true: a node pair can be an anchor in one phase, but be unmapped at a later phase.

We reduce the complexity of the problem by summarizing the structure of the two graphs $G_{1,2}$ as a set of (label-independent) attributes for each node. Then we rely on these node attributes to match 2^τ nodes in each phase τ, used as anchors for the next phase, where this matching is only a function of the attributes. We call the set of attributes of a node its *fingerprint*, and rely on a cost function over pairs of fingerprints to compute the matching. The matching algorithm is described in detail in the next section; in this section, we define the node fingerprints, and the probabilistic model that yields the cost function over pairs of fingerprints, and its Bayesian interpretation.

To model the similarity of two graphs $G_{1,2}$, we use the graph sampling idea introduced in Chapter 1, and used in Chapter 2. The ultimate goal is to obtain a measure of similarity between two node fingerprints. We assume that the observable graphs $G_{1,2}$ are obtained through a sampling process on a (fixed) generator graph $G = (V, E)$. The choice of G is arbitrary, and we make no assumption on the structure of G. As shown later, the only information used from G are its degree and distance distributions. Our Bayesian framework is general, independent of the choice of the sampling process. However, we still need to choose a sampling process in order to rigorously measure the similarity of node fingerprints. We again consider the *edge sampling* process introduced in Chapter 1 to generate G_1 and G_2 from the fixed graph G, with sampling probabilities s_1 and s_2 respectively. We elaborate further on this choice in Section 4.2.3.

In the following, we first formalize the computation of likelihoods and describe one step of this iterative procedure assuming some nodes have already been mapped. We present the complete iterative algorithm that incrementally increases the set of mapped nodes in Section 4.3. As later discussed, the first phase of the algorithm requires no knowledge of any anchor nodes.

4.2.1 Pair-Wise Likelihoods

Consider the two observed graphs G_1 and G_2 and two nodes $u_1 \in V_1$ and $u_2 \in V_2$ with their fingerprints

$$\begin{aligned} F_{u_1} &= [X_{11}, X_{12}, \ldots, X_{1,\ell}, X_{1,\ell+1}, \ldots, X_{1,\ell+m}] \\ F_{u_2} &= [X_{21}, X_{22}, \ldots, X_{2,\ell}, X_{2,\ell+1} \ldots, X_{2,\ell+m}], \end{aligned} \tag{4.1}$$

where each component provides structural evidence for the similarity of u_1 and u_2. More specifically, the components $1, \dots, \ell$ are *absolute* attributes, in that they depend only on the graphs G_1 and G_2, respectively. They may include local graph statistics around node $u_{1,2}$, such as the node degree, the clustering coefficient, or the density of some neighborhood. For example, suppose X_{11} and X_{21} represent the degrees of nodes u_1 and u_2. If the two degrees are quite similar, this is stronger evidence that u_1 and u_2 are referring to the same underlying node $u \in G$; if they differ, $u_{1,2}$ should most likely not be matched.

A key idea in our method is to rely on anchor nodes from the previous phase to obtain additional node attributes for the current phase. In this way, the matching of nodes in later phases can build on nodes matched earlier. The components $\ell + 1, \dots, m$, with $m = |S_{\tau-1}| = 2^{\tau-1}$ are *relative* attributes, which depend both on $G_{1,2}$ and the anchors $S_{\tau-1}$ from the previous phase. They may include different distance metrics between $u_{1,2}$ and each anchor, such as hop distance or resistor distance. If two nodes $u_{1,2}$ have similar distances from a set of anchors, this is strong evidence that they match, *provided the anchors themselves are matched correctly.* Finally, we denote by $X_i = (X_{1i}, X_{2i})$ a pair of corresponding attributes in G_1 and G_2.

We now turn to the fingerprint cost function for such a pair of nodes $u_{1,2}$. Recall that the matching algorithm has no information about the meaning of a node label u_1 and u_2. We model this by assuming that the matching algorithm is looking at two randomly chosen nodes $U_{1,2}$, which are independent uniform random variables over $[1, n]$. It can only rely on the fingerprints of the two nodes $U_{1,2}$ to decide whether in fact $U_1 = U_2$ (match) or not. In other words, $U_{1,2}$ are endowed with a uniform prior distribution[2]. The cost is then the log-likelihood of $U_1 = U_2$ given the two fingerprints, which we write as

$$r(u_1, u_2) = P\left[U_1 = U_2 | F_{u_1}, F_{u_2}\right]. \tag{4.2}$$

Note that in the absence of any node attributes, the probability that two random labels in G_1 and G_2, respectively, correspond to the same node in G is given by $1/n$. Thus, the prior $P[U_1 = U_2] = 1/n$.

Using Bayes' rule and the observation above, we can rewrite (4.2) to obtain:

$$r(u_1, u_2) = \frac{P\left[F_{u_1}, F_{u_2} | U_1 = U_2\right] 1/n}{P\left[F_{u_1}, F_{u_2} | U_1 = U_2\right] 1/n + P\left[F_{u_1}, F_{u_2} | U_1 \neq U_2\right] (1 - 1/n)}. \tag{4.3}$$

Of course, different node attributes may be correlated in complicated ways, depending on the underlying generator G. For example, a high-degree node would have, on average, smaller distances to other nodes in the graph than a low-degree node. However, in large sparse graphs, these correlations can be expected to be quite weak. We therefore assume that pairs of corresponding attributes of two nodes U_1 and U_2 are conditionally independent, given either $U_1 = U_2$ or $U_1 \neq U_2$. In other words, we assume that $P\left[X_1, \dots, X_{\ell+m} | U_1 = U_2\right] = \prod_{i=1}^{\ell+m} P\left[X_i | U_1 = U_2\right]$ and that $P\left[X_1, \dots, X_{\ell+m} | U_1 \neq U_2\right] = \prod_{i=1}^{\ell+m} P\left[X_i | U_1 \neq U_2\right]$, where X_i is a pair of corresponding node attributes. For the latter case $U_1 \neq U_2$, this is because the attributes stem from two random, but different nodes in G; for the case $U_1 = U_2$, this embodies the assumption that the edge sampling process acts on each attribute pair independently (but the attributes within the pair are - possibly strongly - correlated in this case).

---

[2] Of course, other priors are possible, e.g., if additional side information about nodes is available, such as from additional node attributes.

We hereby provide a more in-depth discussion for the validity of our assumption. As will become clear later, the absolute attribute we use is node degree, and the relative attributes are the distances to anchors. The assumption states that if we know that the two nodes are the same or different, then the degree and all distance *pairs* of attributes are independent. The intuition behind this assumption is that (i) a pair of degree and pair of distance attributes can be assumed independent as they are fundamentally different structural characteristics, and (ii) two or more pairs of distance attributes can be assumed independent as the anchors are chosen from the whole node set of the graph, and based only on their priors. This assumption is reasonable especially for large networks.

The notion of pair is important here. We do not claim that individual attributes of two different nodes are independent, which would certainly not be accurate for the case $U_1 = U_2$. However, different pairs are conditionally independent. Finally, we also do not assume that attribute pairs are independent (not conditionally), which is also not accurate for the following reason.

Assume edge sampling parameter $s = 1$ such that G_1 and G_2 are identical. In this case, if we know that $X_{1k} \neq X_{2k}$ for some k, then we can conclude that $u_{1i} \neq u_{2j}$, and therefore $X_{1k'} \neq X_{2k'}$ for some other k' is very likely. If we see that $X_{1k} = X_{2k}$ for some k, then $u_{1i} = u_{2j}$ is very likely, and thus $X_{1k'} = X_{2k'}$ for some other k' is also very likely. This implies that the different attributes are strongly correlated when considered unconditionally, as knowledge of what happens with attribute k has a decisive influence on what happens with another attribute k'. Thus, we only assume conditional independence, which as discussed above, is relatively reasonable.

Under this assumption, we can factor the evidence in (4.3) and obtain

$$r(u_1, u_2) = \frac{1/n \prod_{i=1}^{\ell+m} P\left[X_i | U_1 = U_2\right]}{1/n \prod_{i=1}^{\ell+m} P\left[X_i | U_1 = U_2\right] + (1 - 1/n) \prod_{i=1}^{\ell+m} P\left[X_i | U_1 \neq U_2\right]}. \tag{4.4}$$

where $X_i = (X_{1i}, X_{2i})$ is a pair of corresponding attributes.

In order to apply equation (4.4) we need to determine $P\left[X_i | U_1 = U_2\right]$ and $P\left[X_i | U_1 \neq U_2\right]$ for all $i = 1, \dots, \ell + m$. Note that this probability depends on the sampling process, as the sampling process *transforms* attributes of G. Let $p_i(y)$ denote the probability distribution associated with attribute i in the fixed graph G, thus, $p_i(y) = P[Y_i = y]$, for some node $u \in V$. Similarly, let $q_i(x, y)$ denote the probability function of attribute i after the sampling process has been applied to G given that the same attribute has value y in G, thus, $q_i(x, y) = P[X_{1i} = x | Y_i = y]$, for some $u \in V$. Using these two models together with the fact that G_1 and G_2 are independent samples from the sampling process and by applying the law of total probability, we obtain

$$P[X_i | U_1 = U_2] = \sum_y q_i(x_{1i}, y) q_i(x_{2i}, y) p_i(y). \tag{4.5}$$

Likewise, for the case u_1 and u_2 do not correspond to the same node in G, we have,

$$P[X_i | U_1 \neq U_2] = \left(\sum_y q_i(x_{1i}, y) p_i(y) \right) \left(\sum_y q_i(x_{2i}, y) p_i(y) \right). \tag{4.6}$$

Finally, using the Bayesian framework above ((4.4)), together with the models for how the sampling process transforms given attributes ($q_i(x, y)$, and using (4.5) and (4.6)), we can precisely compute the likelihood that a given pair of nodes is correctly mapped given their fingerprints.

4.2.2 Anchor Mismatch

The approach above allows for the set of anchor nodes mapped in the previous phase to be used to compute likelihoods for the fingerprints of unmapped nodes. However, anchors are subject to matching errors. If we do not take this into account, we may overestimate the likelihoods. Therefore, we refine our probabilistic model to account for the possibility of false matches among anchor pairs.

Recall that $X_{\ell+1}, \ldots, X_{\ell+m}$ depend on the anchor pairs. For example, $X_{\ell+i}$ is the distance from u_1 and u_2 to the i-th anchor pair w_{1i} and w_{2i}, respectively.

Let M_i be an indicator random variable taking value 1 if anchor pair $\ell+i$ has been correctly mapped and 0 otherwise. We now revisit (4.5), and assume that if $M_i = 0$, the distribution of the corresponding attribute pair is as if $U_1 \neq U_2$. In other words, we assume that the distance pair from a node ($u_1 = u_2$ or $u_1 \neq u_2$) to two *different* anchors is independent, in analogy to the assumption on the distance pair from two different nodes $u_1 \neq u_2$ to a correctly matched anchor pair. Thus, we have

$$P[X_{\ell+i}|U_1 = U_2, M_i = 0] = \left(\sum_y q_{\ell+i}(x_{1(\ell+i)}, y)p_{\ell+i}(y)\right)\left(\sum_y q_{\ell+i}(x_{2(\ell+i)}, y)p_{\ell+i}(y)\right) \quad (4.7)$$

Note that this is the same as the conditional probability given that $U_1 \neq U_2$ (see (4.6)). We also assume that a correctly or incorrectly mapped anchor does not change the conditional probability when $U_1 \neq U_2$. Thus,

$$P[X_{\ell+i}|U_1 \neq U_2, M_i] = P[X_{\ell+i}|U_1 \neq U_2] \quad (4.8)$$

as in (4.6).

In order to compute the overall pairwise likelihood, we redefine the likelihood function of equation (4.2) to also depend on anchors being correctly mapped. Thus,

$$\begin{aligned} r(u_1, u_2) &= P[U_1 = U_2|X_1, \ldots, X_{\ell+m}, M_1, \ldots, M_m] \quad (4.9)\\ &= P\left[U_1 = U_2|X_1, \ldots, X_\ell, (X_{\ell+1}, M_1), \ldots, (X_{\ell+m}, M_m)\right]. \end{aligned}$$

As before, we apply Bayes' rule together with conditional independence between attributes pairs to obtain

$$r(u_1, u_2) = \frac{A}{A+B} \quad (4.10)$$

where,

$$A = 1/n \prod_{i=1}^{\ell} P[X_i|U_1 = U_2] \prod_{\substack{i=\ell+1 \\ \text{s.t. } M_i=1}}^{\ell+m} P[X_i|U_1 = U_2], \quad (4.11)$$

and

$$B = (1 - 1/n) \prod_{i=1}^{\ell} P[X_i|U_1 \neq U_2] \prod_{\substack{i=\ell+1 \\ \text{s.t. } M_i=1}}^{\ell+m} P[X_i|U_1 \neq U_2]. \quad (4.12)$$

Note that all the factors, where $M_i = 0$, appear both in the numerator and denominator (in A and B) and therefore cancel out. Moreover, equation (4.10) is for a given vector of M_i values.

Finally, note that $M_i, i = 1, \ldots, m$ are not observable random variables, since we do not know if an anchor was correctly or incorrectly mapped. Therefore, we marginalize all variables M_i to obtain

$$\begin{aligned} P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}] = & \sum_{b=(0,1)^m} \prod_{i=1}^{m} (P[M_i = b_i]) \\ & \cdot P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}, M_1, \ldots, M_m], \end{aligned} \tag{4.13}$$

where b is a bit vector of size m, b_i corresponds to the i-th bit in the vector, and the sum is over all bit vectors of size m. Note that the sum is over an exponentially large set, which is prohibitive. Fortunately, a crude sampling-based estimator of (4.13) does sufficiently well for our purposes (See Appendix 4.A).

Finally, we require the prior on M_i in (4.13). Fortunately, this can be obtained as a byproduct of the previous phase: for an anchor pair (w_{1i}, w_{2i}) matched in the previous phase, we equate its posterior $r(w_{i1}, w_{i2})$ with the prior $P[M_i = 1]$, as this is the likelihood that anchors w_{1i} and w_{2i} were correctly mapped. We discuss this procedure in more detail in Section 4.3.

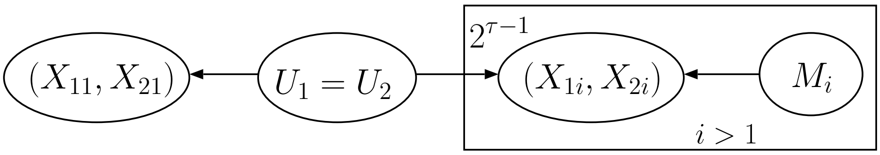


Figure 4.1: Bayesian network for graph matching at phase τ.

4.2.3 Models for Sampled Degree and Distance

The methodology described so far is general, in that we have not yet specified the distributions of specific attributes that determine the fingerprints of nodes. However, these models are needed in order to compute the likelihood function. From now on, we will use the node degree as the only absolute attribute, and hop distance as the relative attribute.

In the following, $X_1 = (X_{11}, X_{21})$ is the degree pair of $u_{1,2}$, and $X_i = (X_{1i}, X_{2i}),\ i > 1$ is the distance pair to anchor pair i[3]. Figure 4.1 depicts the Bayesian network of the formulation as described in Section 4.2.2.

The aforementioned methodology requires a probabilistic model for the node attributes that determine the fingerprint of nodes. In particular, we need probabilistic models that represent the node attributes in G and also after the sampling process, that is, in G_1 and G_2. Note that these models are represented in equations (4.5) and (4.6). As we will use only degree and distance in the fingerprint of a node, we require models only for these two attributes.

As mentioned earlier, we consider the *edge sampling* process. Under the edge sampling process, we can now determine a model for degree and distance on the sampled graph given their values in the original graph G. Recall that $q_i(x, y)$ denotes the probability that attribute i has value x in the sampled graph, given that it has value y in the original graph G. Let

---

[3] More precisely, with (w_{1i}, w_{2i}) the i-th anchor pair, X_{1i} is the graph distance in G_1 to w_{1i} (and analogously for X_{2i}).

$i = 1$ denote the node degree. As the edge sampling process samples each edge independently with probability s, $q_1(x, y)$ follows a binomial distribution with parameter s and y, i.e.,

$$q_1(x, y) \quad = \quad \mathsf{Bi}\,(x, y, s) = \binom{y}{x} s^x (1-s)^{y-x}$$

Finally, note that $p_1(y)$ denotes the degree distribution of G, i.e., the probability of sampling a degree y from G. We comment on the choice of the underlying degree distribution in Section 4.4.

The model for distances under edge sampling requires a more careful treatment as this has not been studied in the literature. Consider two adjacent nodes u and v in G. We will assume that the distance between u and v after sampling follows a geometric distribution. In particular, let $Z_{u,v}$ denote a geometric random variable with parameter α for the distance between nodes u and v after sampling. We provide an intuitive argument for this model. In a well-connected graph G, (i) paths of different lengths $1, 2, 3, \ldots$ tend to exist between two adjacent nodes, and (ii) sampling is likely to preserve a path of length ℓ between two nodes with the same probability α for all path lengths. Note that the number of paths of length ℓ between two nodes is likely to increase with ℓ. Thus, although a path of higher length is more likely to be destroyed by the edge sampling process, the larger number of paths might offset this probability.

Using the above assumption, a distance 1 would be preserved in the sampled graph with probability α, a distance two would emerge between two adjacent nodes in G with probability $\alpha(1-\alpha)$, and so forth. Thus, the sampled distance $Z_{u,v}$ between two adjacent nodes u and v in G has a probability distribution of the form $P[Z_{u,v} = \ell] = \alpha(1-\alpha)^{(\ell-1)}, \quad \ell = 1, 2, \ldots$ which is a geometric distribution with parameter α. The same argument can now be extended to nodes at any distances. Let u and v have a shortest path of length y in G with the path $(u = u_0, u_1, \ldots, u_i, u_{i+1}, \ldots, u_y = v)$. Thus, their distance after sampling can be written as $Z_{u,v} = Z_{u_0,u_1} + \cdots + Z_{u_i,u_{i+1}} + \cdots + Z_{u_{y-1},u_y}$, where the random variable $Z_{u_i,u_{i+1}}$ denotes the distance after sampling for each adjacent pair in the path. We now make an assumption that $Z_{i,i+1}$ are independent for all i. Thus, we have the sum of independent geometric random variables with parameter α, which can be expressed as a negative binomial random variable with parameters y and α, denoted by $\mathsf{NBi}\,()$.

Let X_{1i} for $i > 1$ denote the distance from node u to an anchor node $i-1$ in G_1. Then $q_i(x, y)$ is the probability of observing distance x between a node and the $(i-1)$-th anchor node in the sampled graph, given that their distance in the underlying graph is y. Thus, we have

$$q_i(x, y) = \mathsf{NBi}\,(x, y, \alpha) = \binom{x-1}{y-1} \alpha^y (1-\alpha)^{(x-y)}, \quad x \geq y,\, j = 1, 2. \tag{4.14}$$

We set the parameter $\alpha = s$, where s is the sampling probability of the edge sampling process. As described above, edge sampling tends to preserve paths of different lengths with the same probability, thus using the probability of preserving a single edge (s) as α could be a reasonable assumption. Finally, $p_i(y)$ for $i > 1$ corresponds to the distance distribution of G, i.e., the probability that two given nodes are at distance y in G. We comment on the choice of the underlying distance distribution in Section 4.4.

Note that we investigated the assumption of the geometric distance distribution for adjacent nodes, and negative binomial distribution for larger distances after sampling, via experiments on real data. We did several experiments on an e-mail network dataset, as we will introduce in Section 4.4. Our findings suggest that this assumption holds for real social networks, and that paths of different lengths exist between adjacent nodes and the choice of s for the geometric parameter is indeed reasonable.

4.3 Matching algorithm

The matching algorithm receives as input two graphs G_1 and G_2, with the same number of nodes and models for distance and degree as described in Section 4.2.3. The output of the algorithm is a matching π between all nodes such that $\pi(u_1) = u_2$ where $u_1 \in V_1$ and $u_2 \in V_2$. The algorithm is iterative and at each phase it considers a set of candidate nodes to be matched, denoted by

$$\begin{aligned} V_1^\tau &= \{u_{11}, u_{12}, \ldots, u_{1n_\tau}\}, \quad u_{1i} \in V_1 \\ V_2^\tau &= \{u_{21}, u_{22}, \ldots, u_{2n_\tau}\}, \quad u_{2i} \in V_2 \end{aligned}$$

where n_τ is the size of the candidate set in the τ-th phase, $\tau = 0, 1, \ldots, \log_2 n - 1$, and more specifically, $n_\tau = 2^{\tau+1}$. In particular, in phase τ, the set of candidate nodes are simply the n_τ nodes with the largest degrees in their respective graph, thus $d(u_{ki}) \geq d(u_{kj}), i \leq j$, where $d(u)$ is the degree of node u.

We elaborate on one phase of the algorithm. At phase τ, we consider a set of $m = 2^{\tau-1}$ anchors (previously mapped node pairs) $S_{\tau-1} = \{(w_{11}, w_{21}), \ldots, (w_{1m}, w_{2m})\}$. Note that we set $m = 0$ in the first phase ($\tau = 0$), and the algorithm requires no prior knowledge of anchor nodes. For each of the n_τ^2 node pairs (u_{1i}, u_{2j}) in the candidate set, we use the Bayesian framework described in Section 4.2 to compute their matching likelihood $r(u_{1i}, u_{2j})$ by using their fingerprints. In particular, as discussed in Section 4.2.3, we use as evidence the degree pair $(d(u_{1i}), d(u_{2j}))$ and m distance pairs $(x(u_{1i}, w_{1k}), x(u_{2j}, w_{2k}))$, $k = 1, \ldots, m$, where $x(u, v)$ is the distance between nodes u and v in their respective graph.

Assuming that pair-wise matchings of nodes are independent, the likelihood terms $r(u_{1i}, u_{2j})$ can be factorized, and the probability of a particular mapping π is given by the product of all pair-wise likelihoods under that mapping, i.e.,

$$P_\pi(V_1^\tau, V_2^\tau) = \prod_i^{n_\tau} r(u_{1i}, \pi(u_{1i})), \tag{4.15}$$

We seek the most likely mapping $\pi_* = \arg\max_\pi P_\pi(V_1^\tau, V_2^\tau)$, i.e., the mapping that maximizes the product in (4.15) in the τ-th phase. Taking the logarithm of each likelihood term $r(u_{1i}, u_{2j})$, the problem reduces to maximizing the sum of log-likelihoods for all node pairs:

$$\pi_* = \arg\max_\pi \sum_i^{n_\tau} \log r(u_{1i}, \pi(u_{1i}))$$

This problem can then be solved by using the Hungarian algorithm for maximum weighted bipartite matching [30].

Finally, the matching results from phase τ are used to construct the set of anchor node pairs S_τ (the nodes that were mapped with highest likelihoods) for the next phase, where $|S_\tau| = 2^\tau$. In other words, half of the mapped node pairs are considered as anchors for the next phase $\tau + 1$. This increases the size of the fingerprint of each node in the candidate set with more distance attributes. As $n_\tau = 2^{\tau+1}$, the size of the candidate set of nodes is also doubled in subsequent phases.

The intuition for choosing half of mapped pairs as anchors is that matching errors tend to increase in the ordered (high-to-low likelihood) sequence of mapped pairs. In particular, it is not even guaranteed that a candidate node $u_{1i} \in V_1^\tau$ has a correct match in V_2^τ, and this gives rise to more uncertainty in matching low-likelihood pairs. Thus, we use only the top half of matching pairs as anchors of the next phase, aiming to increase the confidence of the matching. A detailed description of the matching algorithm follows:

1. Input: two graphs G_1 and G_2, with node sets V_1 and V_2, both of size n, and models for degree and distance.

2. Sort the nodes in each graph by degree in descending order. Let V_1' and V_2' be the list of nodes in descending order by degree.

3. Set $\tau = 0$ and $m = 0$. Set the anchor set as $S_{\tau-1} = \emptyset$.

4. Set the size of the candidate set to be mapped as $n_\tau = 2^{\tau+1}$. If $n_\tau > n$, set $n_\tau = n$.

5. Denote the candidate sets as $V_1^\tau = V_1'(1 : n_\tau)$ and $V_2^\tau = V_2'(1 : n_\tau)$, as the list of the first n_τ nodes in V_1' and V_2', respectively.

6. For each node $u_{1i} \in V_1^\tau$ and $u_{2j} \in V_2^\tau$, compute a fingerprint vector as

$$\begin{aligned} F_{u_{1i}} &= [d_{u_{1i}}, x(u_{1i}, w_{11}), \ldots, x(u_{1i}, w_{1m})] \\ F_{u_{2j}} &= \left[d_{u_{2j}}, x(u_{2j}, w_{21}), \ldots, x(u_{2i}, w_{2m})\right], \end{aligned}$$

   where $i, j = 1, \ldots, n_\tau$ and $(w_{1k}, w_{2k}) \in S_{\tau-1}$, $k = 1, \ldots, m$.

7. For each pair of nodes $(u_{1i}, u_{2j}) \in V_1^\tau \times V_2^\tau$, compute the matching likelihood $r(u_{1i}, u_{2j})$, i.e., $P[u_{1i} = u_{2j} | F_{u_{1i}}, F_{u_{2j}}]$, using (4.13). Construct the complete weighted bipartite graph $G(V_1^\tau, V_2^\tau, E)$, where E includes n_τ^2 edges between all node pairs $(u_{1i}, u_{2j}) \in V_1^\tau \times V_2^\tau$, with the edge weights equal to $r(u_{1i}, u_{2j})$.

8. Solve the maximum weighted bipartite matching using the Hungarian algorithm [30]. The output is the matching π of size n_τ with the maximum sum of the log-likelihoods.

9. Sort the matched pairs $(u, \pi(u))$ by the descending order of their matching likelihood. Let S_τ be the set of $n_\tau/2 = 2^\tau$ node pairs with highest likelihoods. Set $m = |S_\tau|$.

10. If $n_\tau = n$, stop. Else, increment τ. Go to step 4.

Note that the algorithm starts with the two highest degree nodes in each graph ($n_\tau = 2$), computes node fingerprints based only on their degrees, and matches them using the bipartite matching. Next, half of the matched nodes (i.e., 1 node), are considered as anchors, and the size of the candidate set is doubled to $n_\tau = 4$. Thus, at the second phase, four nodes are matched by using the distance to one anchor from the previous round, and their degrees. Again, half of the matched nodes (i.e., 2 nodes), are considered as anchors for the next round, and the size of the candidate set is increased to $n_\tau = 8$. This process continues, and at each phase, n_τ nodes are matched using $n_\tau/4$ anchors, until all nodes are mapped. Notice that the matched nodes at each phase, though being considered as anchors for the next phase, are themselves still in the candidate sets V_1^τ and V_2^τ. This allows for a node to change its mapping during the execution of the algorithm. This is useful because a node wrongly mapped in an early phase might be correctly mapped in a subsequent phase when more evidence (in terms of distance to anchors) is available.

4.3.1 Complexity of the Matching Algorithm

We comment on the complexity of different steps of the algorithm, and how we derive the overall complexity.

- Steps 1, 3, 4 and 5 are computed in constant time, and sorting nodes in step 2 can be done in $O(n \log n)$.
- We precompute all distance pairs of attributes $x(u_{1i}, w_{1l})$ and $x(u_{2j}, w_{2l})$, which necessitates the computation of all pair-wise distances in the two graphs. In real graphs the number of edges is usually given by $k = O(n \log n)$. A shortest path algorithm over all node pairs is $O(n(k + n \log n))$, which in this case would be $O(n^2 \log n)$. Using this precomputation, step 6 would take $O(n)$ time, due to the growing size of anchor set with n (in the last iteration).
- It is shown in Appendix 4.A that (4.16) can be computed in $O(m)$, where m is the size of the anchor set. At each phase, we need to compute the likelihoods for n_τ^2 node pairs. Considering $n_\tau = O(n)$ in the last phase of the algorithm, and since the number of anchors grows up to $n_\tau/4$, the construction of the bipartite graph in step 7 is of $O(n^3)$.
- The Hungarian algorithm for maximum weighted bipartite matching works in $O(n(k + n \log n))$, where k is the number of edges in the bipartite graph. Thus, for the complete bipartite graph with $k = n^2$, the time complexity of the algorithm in step 8 is $O(n^3)$.
- Based on the above, the overall complexity of each phase is dominated by steps 7 and 8. The total number of iterations is $\log_2 n$. Thus, the overall complexity of the matching algorithm is $O(n^3 \log n)$.

In order to further reduce the complexity, we should reduce the number of edges in the bipartite graph. We consider the following idea: Instead of computing all $O(n^2)$ edge weights for all node pairs in the bipartite graph, we do a precomputation and consider a fixed set of only $n \log n$ edges, and compute the weights only for these edges as possible mappings in the bipartite graph over all iterations. To do this, we run the first phase of the algorithm over all n^2 node pairs once, using only the degrees as evidence (thus yielding a complexity of n^2 to build the bipartite graph), and then take the $n \log n$ edges with highest weights as the edges that can be present in the bipartite graph for all phases. This would reduce the number of edges in the bipartite graph to $n \log n$ as opposed to n^2 in the last phase. Thus, the complexity in step 7 becomes $O(n^2 \log n)$ which is the cost of building the bipartite graph, the complexity of step 8 becomes also $O(n^2 \log n)$, and the complexity of the algorithm becomes $O(n^2 \log^2 n)$, due to its $\log_2 n$ phases.

The intuition behind this choice is as follows. Notice that node degrees are well-behaved under the sampling process, i.e., the sampled degree is a binomial with parameter s. Thus, our candidate set of edges contains the $n \log n$ node pairs (instead of all n^2 pairs) with most *similar* degrees, so that there is a higher probability that correctly mapped pairs exist within this set. Due to the skewed degree distribution of real networks, we conjecture that this method yields similar matching error, enabling us to apply the algorithm to graphs of larger size.

Finally, there have been some works in the literature on reducing the complexity of maximum weighted bipartite matching problem [23, 29, 34]. More specifically, [23] shows that the maximum weight matching can be done in $O(k\sqrt{n}\log W)$, where k is the number of edges in the bipartite graph, and W is the maximum weight, if all weights are normalized to integers. Applying them to our method and using the technique above would reduce the complexity further to $O(n\sqrt{n}\log^2 n\log W)$.

4.3.2 Normalization of Likelihoods

In practice, we consider a slightly modified version of matching likelihoods derived above, where the likelihoods are normalized. At each phase, after computing a likelihood $r(u_{1i}, u_{2j})$, we normalize it by the square root of the sum of the likelihoods for all pairs such that $u_{1i} \in V_1$ and $u_{2j} \in V_2$. In other words,

$$r'(u_{1i}, u_{2j}) = \frac{r(u_{1i}, u_{2j})}{\sqrt{\sum_{\forall u_{2k}\in V_2} r(u_{1i}, u_{2k}) \; \sum_{\forall u_{1k}\in V_1} r(u_{1k}, u_{2j})}},$$

and $r'(u_{1i}, u_{2j})$ are the weights used in the bipartite matching and anchor selection (steps 8 and 9 of algorithm). The intuition behind this normalization is the following. Recall that we assume independence for pair-wise matching of node pairs, which allows us to use the weighted bipartite matching problem. However, this assumption leads to an overestimation of the joint node pairs likelihood. Intuitively, the joint likelihood of several pairs is likely to be smaller than the product of the likelihood of the pairs. The normalization procedure compensates this effect by reducing the value for the likelihoods. Furthermore, experimental results show that this normalization yields significantly better results in matching all node pairs.

As one example, consider the following scenario. At each iteration, for a certain node u_{1i}, it is possible that its correct mapping is not present in V_2^τ, although it might have a relative high likelihood of being mapped to some other node in V_2^τ, due to the likelihoods' boost as a result of independence assumption. Normalizing by the sum of the probabilities over *all* nodes in V_2 would give a lower likelihood to such a node and could avoid its being mapped to a node in V_2^τ that could be correctly mapped to another node in V_1^τ. As another example, assume two nodes u_{1i} and u_{1j} in V_1^τ, u_{1i} with high likelihoods to a large number of nodes in V_2^τ (due to the independence assumption), and u_{1j} with low likelihoods to most nodes and a high likelihood to only a few nodes in V_2^τ. Intuitively, u_{1j} is more distinguishable than u_{1i}, though without normalization, a wrong mapping would be more likely. Note that both arguments hold for nodes in V_2^τ as well, due to the symmetry of the problem. We compensate for these issues by using the normalization factor introduced above.

4.4 Experiments

We assess the performance of our algorithm using real data and two different scenarios. First, we consider a real graph G and use the edge sampling model with probabilities s_1 and s_2 to instantiate G_1 and G_2, which are then matched. Next, we consider two *similar* graphs G_1

and G_2 from real data, which are then matched. Note that sampling was not used in the latter and parameters will have to be estimated using G_1 and G_2 directly. To measure the performance we define the matching error as the number of incorrectly mapped nodes divided by the total number of nodes.

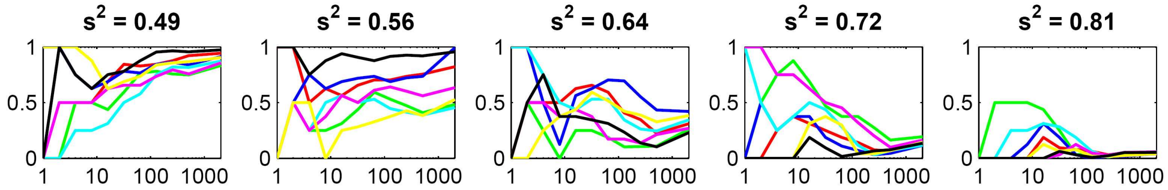


Figure 4.2: Matching error vs. number of mapped nodes for different edge overlaps (7 realizations) using EPFL E-mail network as underlying graph.

4.4.1 Matching Two Sampled Graphs from EPFL E-mail Network

We consider a dataset of e-mail messages collected at the mail server of EPFL. The dataset includes logs of e-mail exchanges among users on a weekly basis and we consider only exchanges *among EPFL users* (i.e., internal EPFL network). We construct the e-mail network (graph) by associating each user to a node and placing an edge between two nodes if there is at least one reciprocal e-mail exchange between the two users. We consider a period of five consecutive weeks and only nodes that appeared in every week, yielding a graph with 2024 nodes and 25603 edges.

Using the graph generated as the underlying graph G, we apply edge sampling to obtain G_1 and G_2. For simplicity, we assume $s_1 = s_2 = s$ and present our results for different values of s^2, which corresponds to the probability that an edge in G appears in both G_1 and G_2 (edge overlap). For each s, we repeat the sampling process to obtain different realizations of G_1 and G_2. To compute the likelihoods, we use the empirical degree and distance distributions of G for $p_i(y)$. Moreover, as we know s, the conditional distribution $q_i(x, y)$ is computed using the degree and distance models presented in Section 4.2.3. Figure 4.2 depicts the matching error as the algorithm progresses. Each plot corresponds to a fixed value of s^2 and each curve corresponds to one realization of G_1 and G_2. The results indicate that the performance of the algorithm at the end (after all nodes are matched) is consistently poor and good for small and large values of s^2 ($s^2 = 0.49$, $s^2 = 0.81$), respectively. The curves also indicate an interesting aspect of the approach: although the error might be high during the first iterations (due to incorrect anchors), the algorithm reconsiders such pairs as it progresses and achieves a smaller error at the end. To some extent, our approach does not depend on correctly identifying initial anchors and is therefore robust to initial errors. Figure 4.3 shows the average final matching error (over 20 realizations) versus the edge overlap s^2. As shown, the error is relatively high for small overlap $s^2 = 0.49$ since the structure of G is significantly destroyed. However, the error drops sharply when we increase s^2, reaching 6% for $s^2 = 0.81$. We can also notice the larger confidence intervals for mid-range s^2, indicating that in such regimes the matching error strongly depends on the realization.

Finally, we show how well our Bayesian method differentiates between the likelihood of mapped nodes and the likelihood of all pairs of nodes in the candidate set. Figure 4.4 depicts the CCDF of likelihoods $r(u_{1i}, u_{2j})$ for (a) all n^2 pairs of nodes in $V_1 \times V_2$ (bottom curve),

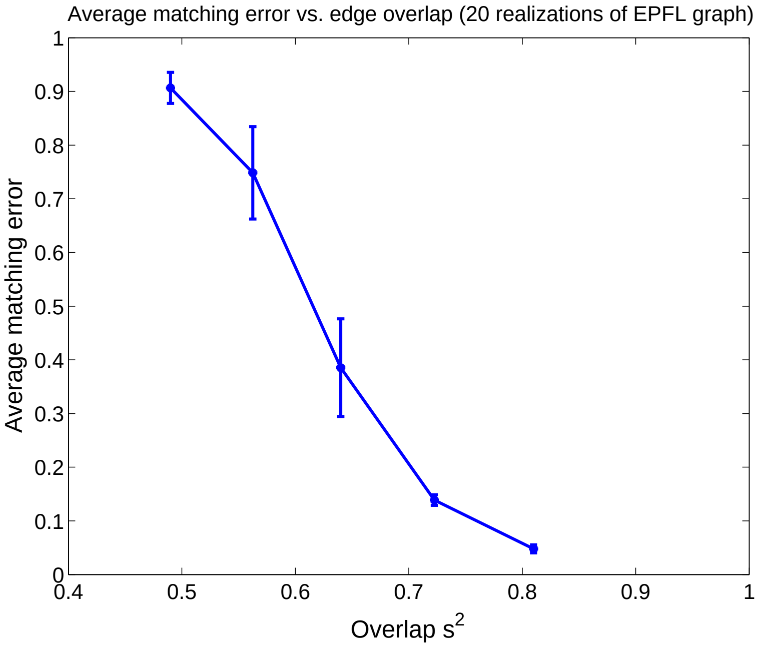


Figure 4.3: Average matching error as a function of edge overlap: curve corresponds to two sampled graphs from a real EPFL E-mail Network (error averaged over 20 realizations).

and (b) all n pairs of nodes mapped through bipartite matching (top curve). The likelihoods are those of the last phase of the algorithm, when it finishes. The clear distinction between these two plots demonstrates the superiority of our method: Most node pairs have very small likelihoods, whereas a small subset of nodes exhibit higher likelihoods, among which are the ones that are matched. For instance, 60% of the mapped node pairs have a likelihood greater than 0.8, whereas the probability of such likelihood over all nodes is slightly above 0.001! In other words, the likelihoods derived through our Bayesian method establish a plausible metric for the probability of two nodes being a correct match. This is the key reason why the maximum bipartite matching resolves the matching problem with minimal error.

4.4.2 Matching Two Snapshots from EPFL E-mail Network

We now attempt to match two real snapshots G_1 and G_2 without considering any fixed underlying graph G. We construct G_1 by considering the set of common nodes over a window of 10 weeks and all edges among them. The graph G_2 is constructed in the same manner by also considering a period of 10 weeks but with a time shift of t weeks relative to G_1. For example, $t = 3$ means that G_1 and G_2 overlap 7 weeks, thus giving $O = 7/(10 + 3) = 54\%$ time overlap. Note that time overlap translates to edge overlap. In particular, we estimate the parameters s_1 and s_2 based on the number of edges in G_1, G_2, and their intersection. We also compute the empirical degree and distance distributions for the (unknown) underlying graph by superposing the distributions in G_1 and G_2 (see Section 4.4.3 for more details). Finally, we apply our algorithm to match G_1 and G_2. Figure 4.5 depicts the matching error as a function of edge overlap between G_1 and G_2. Each point in the curve corresponds to a different time overlap O, as indicated. Note that for small edge overlap (equivalently, small

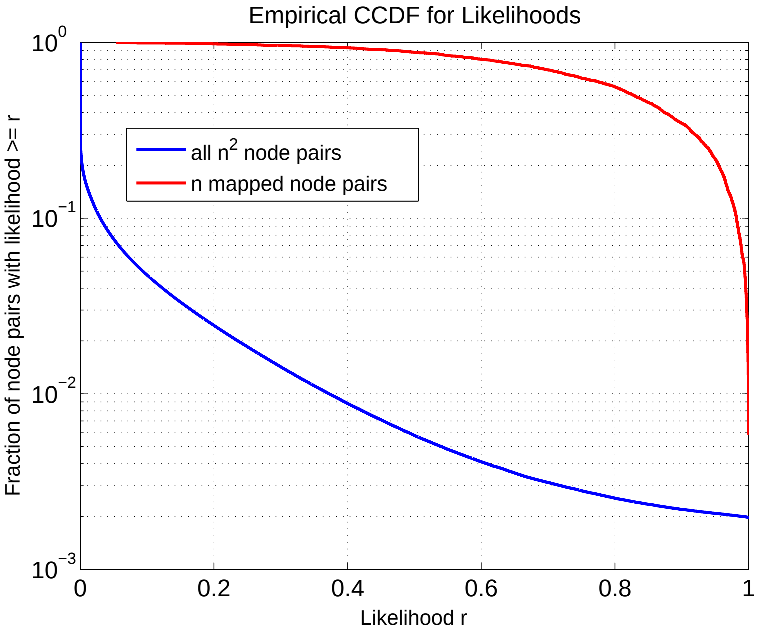


Figure 4.4: CCDF for likelihoods of matched vs. all node pairs.

O), the error is large, as there is little common structure between the graphs. However, the error rate drops sharply as the edge overlap increases: for 75% and 82% overlap ($O = 54\%$, $O = 66\%$), the error is 13% and 6%, respectively, and for 90% overlap ($O = 82\%$), remarkably almost all nodes are correctly matched.

We now construct G_1 by considering the set of common nodes over a window of 20 weeks and all edges among them. The graph G_2 is constructed in the same manner by also considering a period of 20 weeks but with a time shift of t weeks relative to G_1. We repeat this experiment for different time overlaps as in the previous experiments, each exhibiting a different edge overlap.

Figure 4.6 depicts the matching error as a function of edge overlap between G_1 and G_2. Again, we observe an interesting behavior: for small overlap the error is large, and it drops significantly after some threshold. The result is remarkable in the sense that for less than 70% edge overlap ($O = 48\%$), we can already match around 50% of the nodes, and for a 75% edge overlap ($O = 60\%$ only), the error rate drops to as low as 4%!

Further experiments suggest that this phase transition behavior is universal in our method, and indicates that even if the overlap is not too large, we can still achieve considerably low matching errors between two graphs given a minimum amount of overlap (as low as 75% in this experiment). The experiments above indicate the applicability of our method for real data even when the underlying graph is not known.

4.4.3 Estimation of Parameters and Priors

As explained in Section 4.4.2, applying the matching algorithm requires computing the likelihoods which in turn requires the estimation of (i) the prior degree and distance distributions of the underlying graph in order to compute $q_i(x, y)$ (See Section 4.2.3), and (ii) the sampling parameters s_1 and s_2. When directly using the edge sampling model (as in Section 4.4.1),

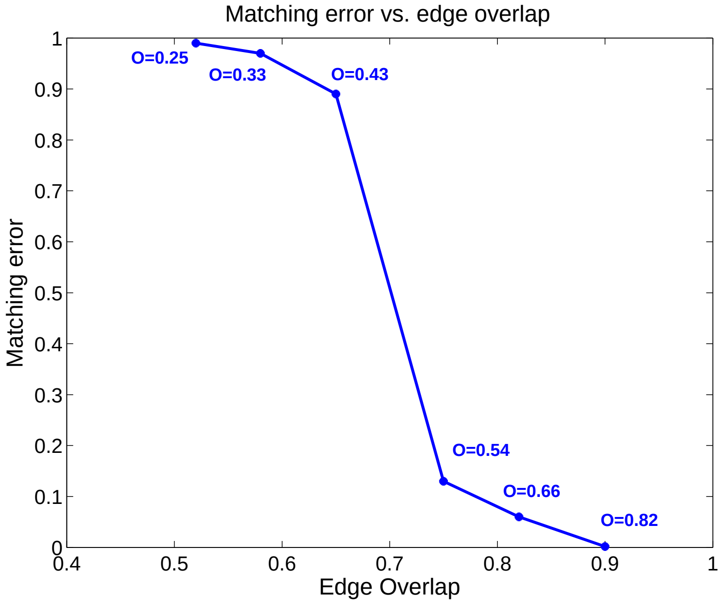


Figure 4.5: Matching error as a function of edge overlap: curve corresponds to two real EPFL E-mail Networks generated with overlapping time windows.

these parameters are known. However, for real graphs, e.g., two snapshots of a network as in Section 4.4.2, we need to estimate these parameters.

We estimate the underlying degree and distance distributions ($p_i(y)$), considering the distributions over the superposition of G_1 and G_2. In other words, we use the empirical degree (or distance) distribution of G_1 and G_2, and compute the underlying degree (or distance) distribution through aggregating the two distributions. This is indeed a crude approximation. However, our experiments show that the algorithm works quite well under this assumption.

For the sampling parameters, we can write $e_1 = es_1$ and $e_2 = es_2$, where e, e_1 and e_2 are the expected number of edges in G, G_1 and G_2 respectively. Also, $e_I = es_1 s_2$ corresponds to the expected number of edges in the intersection of edges of G_1 and G_2. Estimating e_1, e_2 and e_I as the actual number of edges in G_1, G_2, and their intersection (assuming the number of edges in the intersection is known), we can estimate s_1 and s_2 using the equations above. Note that for an accurate estimation, we require knowledge of the number of edges common to both graphs, e_I.

Although in practice we can have an estimation of the ratio $s_1/s_2 = e_1/e_2$, the number of common edges e_I might still be unknown; thus the need to estimate s_1 (or s_2) in a different way, and to compute the other value using the ratio above. One idea is to arbitrarily pick a large value (e.g., > 0.85) for s_1 (or s_2), independent of the graph structures. We did experiments on both sampled graphs and real graphs of two snapshots. Our findings suggest that for the choice of a large similarity parameter s, although it might not accurately match the real s, we still obtain a matching error comparable to the error if the real s was used. For instance, if the real s is 0.8, and we insert $s = 0.9$, the change in the matching error would be less than 15-20%. Also, when matching two snapshots of a network, the interesting observation is that the error might even decrease when we do not use the actual s! One explanation is that we make the assumption that the superposition of G_1 and G_2 corresponds to the underlying

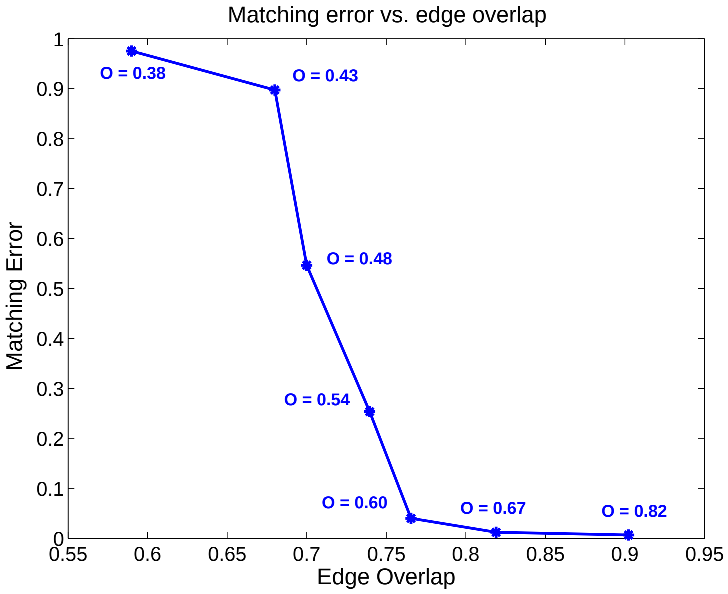


Figure 4.6: Matching error as a function of edge overlap: curve corresponds to two real EPFL E-mail Networks generated with overlapping time windows.

graph, whereas this might be a crude assumption. In short, we believe that for similarity parameters larger than some threshold, our algorithm behaves quite well independent of the choice of this parameter. Of course in case the two graphs are too dissimilar, the matching algorithm fails to find a mapping with small error, as also shown in Sections 4.4.1 and 4.4.2, and this is independent of the choice of the actual s (which is small), or a large s.

Using the estimations above, the matching algorithm can be applied to real networks for matching structurally similar graphs. The applications could vary: matching dictionaries of different languages, de-anonymizing a social network using a correlated publicly available network, matching users' data e.g., e-mail and phone data within a company, etc.

4.5 Summary

We proposed a Bayesian method for approximate graph matching where the two graphs are independent outcomes of some sampling process. Based on this approach we devise an iterative algorithm where evidence used to compute the likelihood of a correct match increases over the iterations by using previously mapped nodes. Using degree and distances as node attributes and assuming an edge sampling model, we apply our algorithm to real data. Our results indicate that accurate matching (less than 10% error) is feasible under reasonable edge overlap (greater than 75%).

Publication [85]

Pedram Pedarsani, Daniel R. Figueiredo and Matthias Grossglauser. **An Iterative Bayesian Method for Seedless Graph Matching**. Submitted to *2013 SIAM International Conference on Data Mining*, SDM '13, Austin, TX, USA.

4.A Appendix

Sampling Method for Computing Likelihood

We comment on the sampling method proposed in Section 4.2.2 to compute the matching likelihood. We hereby restate the matching likelihood:

$$\begin{aligned} &P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}] = \\ &\sum_{b=(0,1)^m} \prod_{i=1}^{m} (P[M_i = b_i]) \cdot P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}, M_1, \ldots, M_m]. \end{aligned} \tag{4.16}$$

The sum is over 2^m possible choices of the bit vectors of size m. Notice that $P(M = b) = \prod_{i=1}^{m} (P[M_i = b_i])$ corresponds to the probability of the vector of indicators $[M_1, ..., M_m]$, and the second term in the sum is a function of this vector. Thus, we can rewrite (4.16) as,

$$\begin{aligned} &P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}] = \\ &\sum_{b=(0,1)^m} P[M = m] \underbrace{P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}, M = b]}_{f(M)}. \end{aligned} \tag{4.17}$$

This corresponds to the expectation of the random variable $P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}]$. Using this observation, instead of computing the sum over over all 2^m terms, since the priors $P(M_i = b_i)$ are known, we propose taking constant number of samples (c) from the prior distribution on M. Denote $M^{(t)}$ as the t-th sampled vector. We compute $f(M^{(t)})$ for each sample, and estimate (4.16) as the sample average over all samples. In other words,

$$P[U_1 = U_2 | X_1, \ldots, X_{\ell+m}] \simeq \frac{1}{c} \sum_{t=1}^{c} f(M^{(t)}). \tag{4.18}$$

Note that as $c \to \infty$, due to the law of large numbers, (4.18) converges to the expected value at the right-hand side. Thus, more samples yields more accurate estimation for the likelihood. This reduces the complexity of computing the likelihood to $O(m)$, where m is the size of the anchor set, since we assume c to be constant (and equal to 50 in experimental evaluations).

Part II

Evolution of Social Networks

5 Network Densification

5.1 Introduction

Over the past decade, investigations in different fields have focused on studying and understanding networks that arise in their respective domains, ranging from biological to social to technological networks. Many of these networks exhibit common topological features, such as a heavy-tailed degree distribution and the small world effect[1].

Most of the work on such complex networks has focused on static scenarios, where a single snapshot of the network is considered for investigation. More recently, the focus has moved to dynamic scenarios, where the network evolves with time. Indeed, most real networks evolve over time, as edges and nodes can be added or deleted from the system. In this context, a recently observed feature is *densification*, which occurs when the number of edges grows faster than the number of nodes. More precisely, the average degree of the network grows with time. This surprising phenomenon is empirically validated by the recent work of Leskovec et al. [63], where densification is observed in six large datasets. Even more surprising is the fact that the observed densification exhibited a very specific and precise relationship, namely, a power law of the form $m(t) \sim n(t)^{\alpha}$, where $m(t)$ and $n(t)$ denote the number of edges and nodes of the network at time t and α is a constant greater than 1. We refer to power-law densification with exponent α as α-densification.

Leskovec et al. [63] propose a mathematical model for network growth, called forest fire model, that results in α-densification. A key element of their model is the fact that newly created nodes establish more edges than nodes that were created earlier. They show that this model can lead to a power-law densification over time.

Densification is a surprising feature. It implies that the average node degree grows without bound, which is counter intuitive in many real settings. For example, does it make sense that an individual's number of social ties depends on the size of the total population of the planet? Even more counter-intuitive is the notion that densification can lead to a decrease of the network diameter (or of the average distance) as the network grows. Does it make sense that the average distance between Internet domains decreases as the Internet grows? Despite these intuitive doubts, the evidence from several disparate datasets is convincing and solid. We believe therefore that reconciling the observed phenomenon with domain intuition is a promising area of research.

Our contribution is a novel explanation for the observed densification in real networks. Our explanation posits that densification can arise as a feature of a common procedure to observe - or measure - dynamic networks, rather than as a feature of the network itself. To sharpen this explanation, we show in this work that densification can actually arise even when observing a *fixed* network that is gradually discovered through a sampling process. This sampling process captures the usual way of observing dynamic networks, as described

---

[1] A network is considered a small world if it exhibits a high clustering coefficient and short pair-wise distances.

in Chapter 1. We argue that the network growth is a direct consequence of the sampling process, i.e., the gradual discovery of this underlying network, rather than a property of the network itself.

To support this claim, we use the *edge sampling model* proposed in Chapter 1. We consider two variants of this model, which capture two common procedures for the observation of real networks. The first variant is the *accumulation model*, where we assume that the observed network is the result of all the edges discovered since the beginning of the observation. As time evolves, the observed network grows and "converges" to the hidden full network. The second variant is the *modulation model*, where we assume that independent snapshots are obtained at different times, which we can view as samples of the hidden full network. We study these two variants to reflect different measurement methodologies used in the studies of network evolution, and we show that both variants can lead to densification in the observed network. How the sampled networks densify depends both on the structural properties of the underlying network and on the sampling process itself.

Using a simplified instantiation of the edge sampling model, we establish analytical results indicating that the number of nodes and edges discovered indeed densify over time. In particular, we prove that densification is present for all time instances greater than a threshold, and we derive a range (as a function of number of discovered nodes) over which densification arises. We also prove that densification converges to a power law as time increases, with an exponent that is inversely proportional to the probability mass of degree one nodes. More importantly, we prove that a network densifies if and only if its degree distribution is a power-law. Finally, we apply our model to real network data and show that it can fairly capture the densification observed in practice. Our results indicate that edge sampling is a plausible alternative explanation for the network densification.

As stated above, the key basis of our model lies in the way many real networks are observed: edges of an underlying graph are observed directly, with nodes being observed indirectly. We do not claim that all networks densify because of edge sampling; nor we do that all real networks densify. However, we do believe that edge sampling can be a plausible explanation for the observed densification of some networks, where the observation of edges leads to the discovery of nodes from the underlying graph. In other words, we explain densification as a feature of the statistical estimation instead of a feature of the real network. In [63] Leskovec et al. propose network growth as a plausible model to explain densification.

We close this section by commenting on the difference between degree power laws, a widely researched feature of many real networks, and power-law densification. Degree power laws have often been linked to the "rich get richer" phenomenon. This refers to the fact that rich (e.g., large degree) nodes tend to become richer (i.e., even larger degrees) as the network grows in size, giving rise to power law degree distributions. This is the main idea, for example, behind the preferential attachment models [47, 10], used for explaining how a single snapshot of the network is formed. This degree distribution is usually fixed, despite models that allow the graph to grow over time. However, power law densification refers to the relationship between number of nodes and edges as the network evolves (i.e., in different snapshots of the network). In other words, power-law degree distribution is the property of a single snapshot of the network, whereas power-law densification refers to the relationship between number of edges and nodes for a sequence of networks (e.g., because the network is growing). Notice that in classic graph models the average degree is assumed to be constant over time (i.e., the number of edges grows linearly in the number of nodes), thus no densification is present (though featuring power law degree distribution). So it is important to note that the two are orthogonal, in the sense that each can exist independently. For example, a sequence of d_n-regular graphs $G(n, d_n)$ with $d_n = n^{\alpha-1}$ ($\alpha > 1$) does not exhibit a degree power law, as every snapshot has constant degree over the ensemble of nodes; however, it does exhibit α-densification. Nevertheless, a sequence of random graphs whose degree distribution is drawn from a distribution D with power-law tail (and with $E[D] < \infty$) does exhibit degree power law, but not densification.

The remainder of this chapter is organized as follows. Section 5.2 shows a concrete example of network densification and the problem statement. Section 5.3 presents the Edge Sampling model, its theoretical properties and numerical evaluations. Sections 5.6 presents an evaluation of the proposed model when applied to real network data. Section 5.7 briefly discusses the related work. Finally, Section 5.8 concludes.

5.2 Motivation

In order to explain and illustrate the concept of densification, we present an example of a real network that densifies over time. In particular, we consider an *e-mail network*. In this directed graph, nodes represent e-mail addresses and edges represent message exchanges. The network evolves in time through the observation of new messages that are exchanged, as described below.

Let $m_{ij}(t)$ denote a message sent from e-mail address i to e-mail address j at time t. Moreover, let $M[t_1, t_2]$ denote the set of all messages sent in the interval $[t_1, t_2]$.
Let $G[t_1, t_2] = (V[t_1, t_2], E[t_1, t_2])$ be a directed graph defined as follows. Let $E[t_1, t_2] = \bigcup_{m \in M[t_1,t_2]} e(m)$, where $e(m_{ij}(t)) = \{(i, j)\}$. Thus, $E[t_1, t_2]$ denotes the set of directed edges that appear in the graph in the interval $[t_1, t_2]$. Similarly, let $V[t_1, t_2] = \bigcup_{m \in M[t_1,t_2]} v(m)$, where $v(m_{ij}(t)) = \{i, j\}$. Thus, $V[t_1, t_2]$ denotes the set of nodes that appear in the graph in the interval $[t_1, t_2]$.

We consider a dataset of e-mail messages collected at the mail server of EPFL for a period of 89 weeks. The dataset is aggregated by week, such that timestamps are in the timescale of weeks (i.e., all messages sent in a particular week have the same timestamp). To filter out bogus messages (spam, mailing lists, wrong addresses, etc), only e-mails to or from registered EPFL personnel were considered, as well as only e-mail addresses that both sent and received a message at least once in the observed interval of time. The filtered dataset contains a total of $23,679,417$ e-mail messages.

From this point, we use $m(t)$ to denote the number of edges and $n(t)$ to denote the number of nodes, as a function of time. Following the procedure describe above, we construct the e-mail network by growing it one week at a time, computing $G[1, t]$, where $t = 1, \ldots, 89$. Thus, for each week t, we have a value for the number of edges $m(t) = |E[1, t]|$ and the number of nodes $n(t) = |V[1, t]|$ in the graph. Figure 5.1 plots the number of nodes versus the number of edges for all values of t in log-log scale. The result clearly shows a densification of the network, as the number of edges increases much faster than the number of nodes. In particular, the number of nodes goes from $n(1) = 45,782$ to $n(197) = 431,200$, while the number of edges grows from $m(1) = 115,898$ to $m(197) = 3,945,937$. Moreover, we can observe a fairly precise linear relationship between $\log m(t)$ and $\log n(t)$, which indicates that the two variables are related through a power law of the type $m(t) \sim n(t)^{\alpha}$. Let $\hat{\alpha}$ be the slope of the line obtained by performing a linear regression of the points in the plot. Thus, for the data shown in Figure 5.1 we have that $\hat{\alpha} = 1.57$.

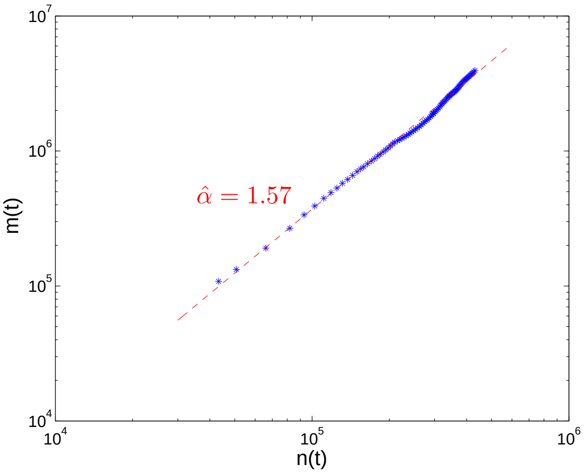


Figure 5.1: Evolution of the number of nodes $n(t)$ versus the number of edges $m(t)$ over time for EPFL e-mail network.

As stated earlier, Leskovec et al. [63] have also observed power-law relationships between the number of edges and nodes for various real networks, including an e-mail network. The fact that power laws seem to be ubiquitous relationship between the number edges and nodes of an evolving network motivates us to search for alternative explanations for densification.

It is important to notice how the e-mail network was grown in the example above. Indeed, the network is grown one message at a time, revealing edges at each step, and nodes are discovered through these edges. We believe that many real networks are also grown in this same way, that is, through the observation of edges. For example, in the World-Wide Web network, web documents (nodes) are discovered through the observation of hyperlinks in the documents (edges). In the Internet AS-level graph, ASes (nodes) are inferred through the observation BGP announcements (edges). In the IMDB actors to movies networks, actors and movies (nodes) are discovered when an actor plays in a movie (edges). Thus, the process of growing the network through the observation of edges (or *edge sampling*) seems fairly universal and is the key motivation for the model we introduce next.

5.3 Model

In this section, we use the edge sampling model, and elaborate on how it can well explain the densification behavior. We use this model to derive different properties of densifying graphs, and validate our results both theoretically and empirically. We elaborate the model by considering specific properties of the underlying graph and specific properties for the sampling of edges.

In particular, let $G = (V, E)$ be the underlying graph with $n = |V|$ denoting the total number of vertices and $m = |E|$ denoting the total number of edges in the graph. Let $\bar{f}(k)$ denote the empirical degree distribution of G. We obtain a single sample $G_s = (V_s, E_s)$ of the graph by sampling its edges independently with probability s. We define the set $E_s \subset E$ to denote the set of edges that have *fired* in the realization when the sampling parameter is s, hence discovered. Similarly, define the set $V_s = \bigcup_{e \in E_s} \gamma(e) \subset V$ to denote the set of vertices that have been discovered, where $\gamma(e)$ is a function that returns the set of vertices adjacent to the edge e, that is, $\gamma\left((u,v)\right) = \{u,v\}$. Thus, $|E_s|$ and $|V_s|$ are the number of edges and vertices, respectively, that are discovered when the sampling parameter is s. In the following, we determine both $E[|E_s|]$ and $E[|V_s|]$, that is, the expectation of these random variables. For the ease of expression, from this point we use the following notations:

$$n(s) = E[|V_s|] \tag{5.1}$$
$$m(s) = E[|E_s|] \tag{5.2}$$

Since all edges are discovered with same probability s, we can write:

$$m(s) = ms \tag{5.3}$$

Let $\bar{d}$ denote the expected degree of the underlying graph. It can be shown that $m = n\bar{d}/2$. Thus, we can rewrite equation (5.3) as follows:

$$m(s) = n\bar{d}s/2 \tag{5.4}$$

Let q denote the probability that a node is discovered by the process when the sampling parameter is s. This means that at least one edge adjacent to the node fired. This probability is the complement of the node not being discovered, which means that none of the edges adjacent to the node fired when edges fire with probability s. Thus, conditioning on the node degree and sampling of the adjacent edges, we have:

$$q = \sum_{k=0}^{\infty} \bar{f}(k)\left[1-(1-s)^k\right] \tag{5.5}$$

Similarly to $m(s)$, we can then write $n(s)$ as follows:

$$n(s) = nq \tag{5.6}$$

We can establish limiting results for both $m(s)$ and $n(s)$ as s goes to 1. In particular, we can show that:

$$m(s \to 1) = \lim_{s\to 1} m(s) = n\bar{d}/2 \tag{5.7}$$
$$n(s \to 1) = \lim_{s\to 1} n(s) = n(1-\bar{f}(0)) \tag{5.8}$$

Thus, $s = 1$ reveals all the edges of G, but only its non-isolated nodes (i.e., nodes with positive degree).

As stated earlier, our ultimate goal is to establish a relationship between $m(s)$ and $n(s)$. In particular, can the edge sampling model lead to densification on the number of edges and nodes discovered? We start by defining two terms: densification and α-densification. The former means that the number of edges and nodes discovered are related super-linearly, but not necessarily through a power law relationship. The latter means that densification follows a power law relationship of the form $m(s) \sim n(s)^{\alpha}$ for an approximately constant $\alpha > 1$. It is usually understood that this relationship should span several orders of magnitude over $n(s)$ to be unambiguous; we do not make this requirement explicit in the definition.

As we soon show, the edge sampling model can lead to densification, and under some conditions to α-densification. This indicates that densification can arise based on *how* we observe the fixed underlying graph, without requiring any dynamic growth process that modifies its structure.

We investigate the relationship between $m(s)$ and $n(s)$ by defining $\alpha(s)$, as follows:

$$\begin{aligned}\alpha(s) &= \frac{\partial \log(m(s))}{\partial \log(n(s))} \\ &= \left(\frac{\partial \log(m(s))}{\partial s}\right)\left(\frac{\partial \log(n(s))}{\partial s}\right)^{-1} \qquad (5.9)\end{aligned}$$

Thus, $\alpha(s)$ denotes the instantaneous slope of the $n(s)$ versus $m(s)$ plot in log-log scale. Densification then means that $\alpha(s) > 1$ for several order of magnitude on $n(s)$, while α-densification means that $\alpha(s)$ is larger than 1 and is approximately constant for several orders of magnitude on $n(s)$.

Using equations (5.4) and (5.6), we can derive $\alpha(s)$ analytically by applying its definition, which is given in equation (5.9). In particular, we have:

$$\alpha(s) = \frac{(1-s)\left(1 - \sum_k \bar{f}(k)(1-s)^k\right)}{s \sum_k \bar{f}(k)k(1-s)^k} \qquad (5.10)$$

We can obtain both a lower bound and an upper bound for $\alpha(s)$. In particular we show that for $s > 1/2$, the following holds:

$$1 \le \alpha(s) \le \frac{1}{\bar{f}(1)}. \qquad (5.11)$$

The proofs are found in Appendix 5.A.

We can also establish two limiting results for $\alpha(s)$. Let $\alpha(s \to 1)$ be the limit of $\alpha(s)$ when s goes to 1. Similarly, let $\alpha(s \to 0)$ be the limit of $\alpha(s)$ when s goes to zero. We show that:

$$\begin{aligned}\alpha(s \to 1) &= \lim_{s \to 1} \alpha(s) = \frac{1 - \bar{f}(0)}{\bar{f}(1)} \qquad (5.12) \\ \alpha(s \to 0) &= \lim_{s \to 0} \alpha(s) = 1 \qquad (5.13)\end{aligned}$$

Once again, the proofs are found in Appendix 5.A.

The above result can be intuitively understood as follows. As s grows to 1, where the probability that an edge fires is quite large, the probability that a node of degree two or higher has been revealed is very close to one. The asymptotic slope of the $n(s)/m(s)$ curve when almost all nodes have been discovered is then dominated by the discovery of the remaining edges and nodes of degree one, which occur at the same rate. As s shrinks to 0, where very few edges fire and therefore very few nodes are discovered, it is very likely that when an edge fires it will reveal two undiscovered nodes. Thus, the number of edges and nodes will grow linearly.

To support this last finding, we can establish yet another result for the case s is small, near zero. In particular, when s is small we have that $1 - (1-s)^d \simeq ds$. Therefore, $q \simeq s\bar{d}$. And finally,

$$n(s) \simeq n\bar{d}s \qquad (5.14)$$

Using equations (5.4) and (5.14) and applying it to equation (5.9), we obtain $\alpha(s) = 1$. Therefore, when s is sufficiently small, the discovery process through sampling does not yield densification.

It is interesting to note that the instantaneous slope of edge-node curve, $\alpha(s)$, in the asymptotic regimes analyzed above (i.e., when s is near 0 or near 1), does not depend on the degree distribution, which might seem counter-intuitive. However, these results say nothing about the behavior of the $n(s)/m(s)$ curve for non-extreme values of the sampling parameter. In what follows, we estimate $\alpha(s)$ for a range of values s when the degree of the underlying graph follows a specific distribution.

5.3.1 Densification Regimes and the Value of Cutoff

In this section, we first elaborate the transition of different phases by changing the sampling parameter s as we sweep the edge-node curve. Thus we hereby clarify the different regimes for the densification behavior, focusing on the regime where densification holds. More specifically, we introduce three regimes over the evolution of edge-node curve: transient regime, densification regime and saturation regime. This is well depicted in Figure 5.2.

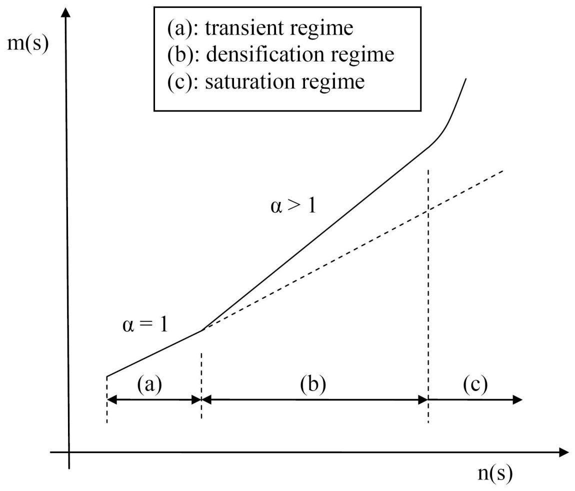


Figure 5.2: The number of nodes discovered ($n(s)$) versus the number of edges discovered ($m(s)$) in log-log scale, resulting in different regimes as the sampling parameter s is changed.

As discussed before and shown in Equations (5.12) and (5.13), the limiting behavior of $\alpha(s)$ is only a function of lags zero and one of the degree distribution of the underlying graph, and not the tail behavior. Thus, as we start growing s from zero, the edge-node curve in log-scale follows a linear relationship with slope 1 (transient regime). As s goes beyond a certain threshold, which we name it as *cutoff*, densification starts (densification regime). The slope of edge-node curve depends on the degree distribution of the underlying graph. As we will soon show, α-densification (i.e., $\alpha(s)$ being constant) is observed over several orders of magnitude of $n(s)$ in case of power-law degree distribution. Finally, as s gets close to 1, $\alpha(s)$ grows to $\alpha(s \to 1)$ (saturation regime), resulting in the saturation of edges and nodes with a high exponent.

Thus the regime of interest is the middle, where densification emerges. Here we derive the exact value of cutoff (starting point for densification) as a function of the parameters of the degree distribution, and validate our results numerically in Section 5.5.

To do so, we introduce an approximation for the densification regime that plays a key role in our theoretical results that follow. Given the sampling parameter s, the probability that a node with degree d is discovered is given by $1-(1-s)^d$. Note that this probability is close to 1 for large enough d. Moreover, when the node degree follows a power-law (such as for Zipf distribution), nodes with large enough degree can exist with non-negligible probability. This explains the following approximation for the probability that a node is discovered. If a node has degree greater than $1/s$, then with probability one it is discovered, otherwise with probability zero it is discovered. Thus, we let $1/s$ define a large enough degree. As before, let q denote the probability that a node is discovered. Using the approximation above, we have:

$$q \simeq P(D > 1/s) \tag{5.15}$$

where D is a random variable with distribution $f(k)$ denoting the node degree. Note that in our notation, $f(k)$ corresponds to the degree distribution for random variable D as the degree, whereas $\bar{f}(k)$ denotes the empirical degree distribution of the underlying graph. Similarly, $E[D]$ and $\bar{d}$ correspond to the average degree in each case, respectively.

Now notice that out of the d edges incident to a given node with degree d, the expected number of edges that will fire is given by ds. The above approximation is equivalent to saying that a node will be discovered if this expectation is greater than 1, but it will remain unknown to the discovery process otherwise.

Note that the above approximation is appropriate only for the *densification regime*, i.e. as densification starts when nodes are being discovered through edges, whereas in the transient regime edges and nodes are discovered linearly (as shown in Section 5.3). We use this idea for finding the cutoff. In other words, we examine *for which values* of s the approximation holds, which is the key in finding the cutoff value.

Let us assume $d_{\max}$ to be the maximum degree in the underlying graph. Observe that the approximation holds only for $s > 1/d_{\max}$, where $d_{\max}$ is the maximum degree of the graph. Indeed, for $s < 1/d_{\max}$, the right-hand side of (5.15) will be zero, thus no such approximation will be valid. This observation suggests that the cutoff occurs at the threshold value of $s = 1/d_{\max}$, which we call s^*, above which densification is present. So the transition from no densification to densification occurs at the cutoff value of:

$$s^* \simeq \frac{1}{d_{\max}}. \tag{5.16}$$

Note that the result holds for *any* distribution of the underlying graph. Furthermore, we can derive the approximate values of $m(s)$ and $n(s)$ for this value of s^*. This can be done as follows:

$$\begin{aligned} m(s)^* &= ms^* = nE[D]s^*/2 = \frac{1}{2}\left(\frac{n}{d_{\max}}\right)E[D] && (5.17)\\ n(s)^* &= n\sum_{k=1}^{d_{\max}} f(k)\left[1-(1-s^*)^k\right] \\ &\simeq n\sum_{k=1}^{d_{\max}} f(k)\left[kp^*\right] = \left(\frac{n}{d_{\max}}\right)E[D], && (5.18)\end{aligned}$$

where (5.18) follows since s^* is small and approximately $1/d_{\max}$.

The above result states that densification (or α-densification for certain cases, as shown below) emerges when the average number of discovered nodes exceeds the average degree of the graph (when the maximum degree in close to the total number of nodes). This is indeed confirmed by the numerical results in Section 5.5. In the next section, we focus on the densification regime and derive the condition for the underlying graph for which α-densification is present.

5.4 Conditions for Power-Law Densification

Now that we defined the different regimes in the sampling model, and derived the cutoff value, we should comment on the densification regime, and the dependence of the densification behavior on the underlying graph. As shown in Section 5.3, the instantaneous slope of edge-node curve in log-log scale is a function of the sampling parameter s. However, we hereby investigate whether α-densification is present for certain cases. Notice that α-densification corresponds to a constant densification exponent α over the densification regime, i.e. $\alpha(s)$ not depending on s.

5.4.1 Necessary Condition for Power-Law Densification

So far, we found the value of cutoff above which densification starts. In this section, we answer the following important question: Under what conditions do we see *α-densification*? We show in the following that to observe α-densification in the densification regime, the degree distribution of the underlying graph should be of a certain form which can be well approximated by a power-law. In other words, we will show that a power-law degree distribution is a *necessary* condition to see α-densification. Our argument uses approximation (5.15) used in Section 5.3.1.

The approximation is valid only for $s > 1/d_{\max}$, since the maximum degree cannot exceed $d_{\max}$. Now remember that the average number of edges and nodes at every sampling instance is expressed by (5.3) and (5.6) as follows:

$$\begin{aligned} m(s) &= ms \\ n(s) &= nq \end{aligned} \tag{5.19}$$

Since m and n are constants, in order to have a power-law relationship between the number of edges and nodes (i.e. to see α-densification), we should have $s \sim q^{\alpha}$, where α is the constant densification exponent larger than 1. We set $\gamma - 1 = \alpha^{-1}$, and write the probability of node discovery as:

$$q = cs^{\gamma-1}, \qquad 1 < \gamma < 2 \tag{5.20}$$

for some constant c. Now, as before, let $f(k)$ denote the degree distribution of the underlying graph. Combining (5.15) and (5.20) yields:

$$q \simeq P(D > 1/s) = \sum_{k=1/s}^{d_{\max}} f(k) = cs^{\gamma-1}. \tag{5.21}$$

To observe α-densification, this equality should hold for the values of s above the cutoff $1/d_{\max}$ - as obtained in Section 5.3.1 -, so that the edge-node curve is swept over several orders of magnitude. Knowing that k can take integer values for the discrete distribution, we start by evaluating (5.21) for different values of s, starting from $s = 1/d_{\max}$. We can write:

$$\begin{aligned}
s = \frac{1}{d_{\max}} \quad &\Longrightarrow \quad \sum_{k=d_{\max}}^{d_{\max}} f(k) = f(d_{\max}) = c\left(\frac{1}{d_{\max}^{\gamma-1}}\right)\\
s = \frac{1}{(d_{\max}-1)} \quad &\Longrightarrow \quad \sum_{k=d_{\max}-1}^{d_{\max}} f(k) = f(d_{\max}) + f(d_{\max}-1) = c\left(\frac{1}{(d_{\max}-1)^{\gamma-1}}\right)\\
&\Longrightarrow f(d_{\max}-1) = c\left(\frac{1}{(d_{\max}-1)^{\gamma-1}} - \frac{1}{d_{\max}^{\gamma-1}}\right)\\
s = \frac{1}{(d_{\max}-2)} \quad &\Longrightarrow \quad \sum_{k=d_{\max}-2}^{d_{\max}} f(k) = f(d_{\max}) + f(d_{\max}-1) + f(d_{\max}-2) = c\left(\frac{1}{(d_{\max}-2)^{\gamma-1}}\right)\\
&\Longrightarrow f(d_{\max}-2) = c\left(\frac{1}{(d_{\max}-2)^{\gamma-1}} - \frac{1}{(d_{\max}-1)^{\gamma-1}}\right)\\
&\vdots
\end{aligned}$$

And in general $f(k)$ can be expressed as:

$$f(k) = \begin{cases} c\left(\frac{1}{k^{\gamma-1}} - \frac{1}{(k+1)^{\gamma-1}}\right), & 1 \leq k < d_{\max} \\ c\left(\frac{1}{k^{\gamma-1}}\right), & k = d_{\max} \end{cases} \tag{5.22}$$

Thus the family of distributions giving rise to α-densification have the form of (5.22). We now further simplify this result for $1 \leq k < d_{\max}$ as follows:

$$f(k) = c\left(\frac{1}{k^{\gamma-1}} - \frac{1}{(k+1)^{\gamma-1}}\right) = c\frac{(k+1)^{\gamma-1} - k^{\gamma-1}}{k^{\gamma-1}(k+1)^{\gamma-1}} \overset{(a)}{\simeq} \frac{c(\gamma-1)k^{\gamma-2}}{k^{\gamma-1}(k+1)^{\gamma-1}} \simeq \frac{c(\gamma-1)}{k^{\gamma}}, \tag{5.23}$$

where (a) follows for k sufficiently large. Indeed, the above result shows that the family of networks exhibiting a power-law densification have a degree distribution close to a power-law (of the form $ck^{-\gamma}$). Note that the resulting PDF corresponds to the family of power-law distributions with the power exponent γ lying between 1 and 2. Indeed, the above argument suggests that in order to observe α-densification, the underlying graph should have a degree distribution in the form of a power law with degree exponent between 1 and 2. This is well in line with the fact that many real network topologies exhibit a heavy-tailed node degree distribution [14, 6].

In the following, assuming a power-law degree for the underlying graph, we show that α-densification emerges, and establish the relationship between the degree exponent and densification exponent.

5.4.2 Sufficient Condition for Power-Law Densification

Having proved the power-law degree as a necessary condition for α-densification, we assume such a degree distribution for the underlying graph, show that it leads to α-densification (and thus is a sufficient condition for α-densification), and establish a relationship between the degree exponent and the densification exponent.

To elaborate on this, we will consider a Zipf distribution to model the degree of the fixed underlying graph G. Thus,

$$f(k) = c\frac{1}{k^\gamma}, \quad k = 1, \ldots, d_{\max} \tag{5.24}$$

where γ is the Zipf parameter, and c is a normalization factor with $d_{\max}$ corresponding to the maximum degree in the underlying graph G. Note that as $d_{\max}$ grows large, the Zipf distribution approximates a power-law. Moreover, it can be shown that for very large $d_{\max}$ the distribution has infinite mean and variance for $\gamma < 2$, finite mean and infinite variance for $2 \leq \gamma < 3$, and finite mean and variance for $\gamma \geq 3$.

We have

$$1 = \sum_{k=1}^{d_{\max}} ck^{-\gamma} \sim \int_{k=1}^{d_{\max}} ck^{-\gamma}\, dk \quad \Longrightarrow \quad c = \frac{\gamma - 1}{1 - d_{\max}^{1-\gamma}} \simeq \gamma - 1,$$

when $\gamma > 1$ and $d_{\max}$ is large.

We again use the approximation (5.15), and try to verify the relationship $m(s) \sim n(s)^\alpha$, and derive the exponent α as a function of the parameters of the distribution. For the number of discovered nodes, using the approximation (5.15), we can write

$$\begin{aligned} q \simeq P(D > 1/s) &\simeq \int_{k=1/s}^{d_{\max}} ck^{-\gamma}\, dk \\ &= \frac{c}{\gamma - 1}(s^{\gamma-1} - {d_{\max}}^{1-\gamma}) = \frac{s^{\gamma-1} - {d_{\max}}^{1-\gamma}}{1 - {d_{\max}}^{1-\gamma}}. \end{aligned} \tag{5.25}$$

Observe that if $s^{\gamma-1} \gg {d_{\max}}^{1-\gamma}$, i.e. $s \gg 1/d_{\max}$, the right-hand side of (5.25) can be approximated as $s^{\gamma-1}$, where as this does not hold if $s < 1/d_{\max}$.[2]

Thus, for n large, we obtain

$$\begin{aligned} m(s) &= ms \\ n(s) &= ns^{\gamma-1}. \end{aligned}$$

Since m and n are fixed, we have $m(s) \sim n(s)^\alpha$ for all $s > 1/d_{\max}$, with

$$\alpha = \frac{1}{\gamma - 1}, \quad 1 < \gamma < 2 \tag{5.26}$$

This result shows that a power-law degree distribution is a sufficient condition for power-law densification. Furthermore, it establishes a direct relationship between the exponent of the degree distribution (γ) of the underlying graph and the α-densification exponent. Notice that the obtained exponent is *constant*, not depending on s, and only dependent on the parameter of the distribution. As we will soon show through numerical evaluations, this relationship is indeed a good approximation for the densification exponent α.

Notice that approximation (5.15) is too crude for $\gamma > 2$, as in this case s becomes larger than q (i.e., the the probability of edge discovery gets larger than the probability of node discovery, see (5.25)). For $\gamma > 2$, we conjecture that α is close to 1, in particular for increasingly

---

[2] Again, we derive the cutoff value of $s^* = 1/d_{\max}$ for the specific case of power-law degree distribution. We derived this result for any distribution in Section 5.3.1.

larger values of γ. We will investigate this issue numerically in the next section. Thus, when $\gamma > 2$ the edge sampling model does not yield densification. This was also previously shown in Section 5.4 as a necessary condition for α-densification.

For Zipf distribution, the average degree can be easily approximated as

$$E[D] \simeq \frac{\gamma - 1}{2 - \gamma}(d_{\max}^{2-\gamma} - 1). \tag{5.27}$$

Using this, the cutoff values $n(s^*)$ and $m(s^*)$ can well be approximated using (5.17) and (5.18). Thus, considering (5.26) and (5.27), we can observe that as the degree exponent increases, densification exponents decreases, while the range over which we see α-densification becomes larger (since $E(D)$ and thus the value of cutoff becomes smaller). This is also validated by numerical results in Section 5.5.

To summarize, we introduced the different regimes over the change of sampling parameter, calculated the value of cutoff for densification, and illustrated that for the case of power-law degree distribution we observe α-densification with the exponent α derived as a function of the degree exponent. More specifically, we showed that a power-law degree distribution for the underlying graph with exponent between 1 and 2 is a necessary and sufficient condition for exhibiting α-densification. Furthermore, we obtained the relationship between the degree exponent and the densification exponents ($\alpha = \frac{1}{\gamma-1}$).

5.5 Numerical Evaluations

In this section we conduct numerical evaluations of the edge sampling model presented above. We assume that the node degree of the underlying graph follows a Zipf distribution, given by equation ((5.24)), with γ denoting the exponent. Moreover, we assume that the underlying graph has $n = 10^7$ nodes. Note that in this case, $d_{\max} \simeq n$.

Figure 5.3 shows the evolution of $m(s)$ and $n(s)$ over different values for the sampling parameter s in log-scale, computed using equations (5.4) and (5.6), for a Zipf distribution with $\gamma = 1.5$. The curves exhibit an interesting behavior. As shown in Figure 5.3, at first, while s is small, the number of nodes discovered grows faster than the number of edges discovered. This occurs because in this range, almost every edge discovery results in the discovery of two nodes, which leads to a constant slope in edge-node curve. This is in line with our claim before, i.e. no densification when s is close to zero. Also notice the value of s where the number of edges exceeds the number of nodes and densification starts: the cutoff value for sampling parameter closely matches our theoretical result of $s^* = 1/d_{\max} = 10^{-7}$. As s increases beyond the cutoff, the number of nodes discovered grows much slower than the number of edges. This occurs because in this range almost all nodes have been discovered while edges continue to be discovered. Finally, as s approaches 1, the discovery process starts saturating.

Figure 5.4 depicts the expected number of nodes discovered versus the expected number of edges discovered using the same sampling parameter for Zipf distributions with different degree exponents. Notice that the plot is in log-log scale, thus, densification is present when the derivative (i.e., slope) of the curve is greater than 1 for a wide range of scales on $n(s)$. Moreover, if this derivative is approximately constant for a wide range of scales on $n(s)$, then the relationship between $m(s)$ and $n(s)$ is a power law of the type $m(s) \sim n(s)^{\alpha}$, where α is the slope of this straight line. Indeed, the curves for the Zipf distribution with $\gamma < 2$ in Figure 5.4 show α-densification, as indicated by the theoretical analysis.

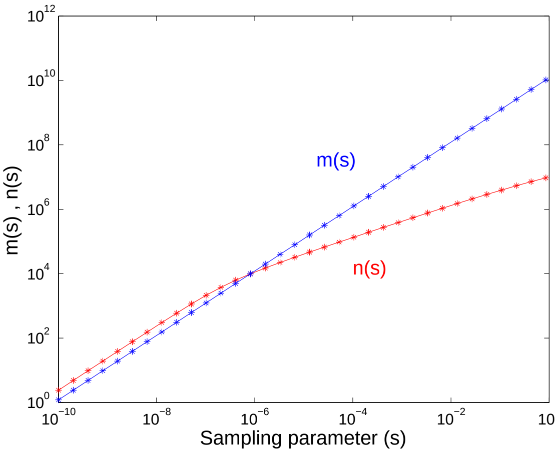


Figure 5.3: Evolution of the number of nodes ($n(s)$) and the number of edges ($m(s)$) discovered over change of sampling parameter for an underlying graph with a Zipf degree distribution ($n = 10^7$, $\gamma = 1.5$).

Figure 5.4 also illustrates the different regimes for α. In particular, when s is small enough, we have α close to 1. As s increases beyond the cutoff value of s^*, i.e. when the number of discovered nodes exceeds $n(s)^*$, we have α greater than 1 and approximately constant. Moreover, as we increase γ, the range over which densification is present (i.e., α greater than 1) decreases (for a fixed n), however, a higher α-densification exponent is achieved (as proved in Section 5.3.1). For $\gamma = 1.5$, α-densification spans three decades, while for $\gamma = 1.8$ it spans five decades. Finally, for $\gamma = 2.5$, we observe a slope close to 1 for the entire range of values for s, i.e. no densification is present, as expected.

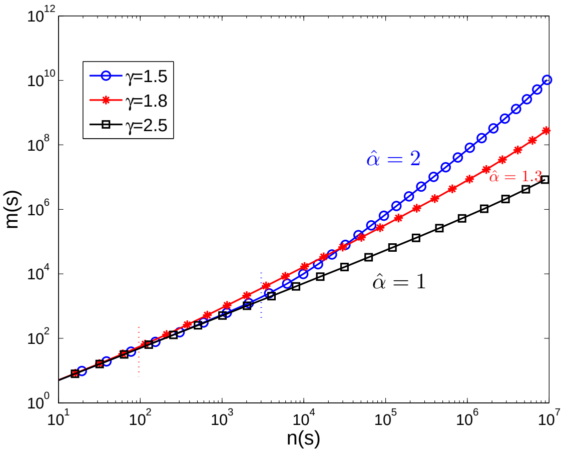


Figure 5.4: The number of nodes discovered ($n(s)$) versus the number of edges discovered ($m(s)$) by the same sampling parameter for Zipf degree distributions ($n = 10^7$, $\gamma = 1.5$, $\gamma = 1.8$, $\gamma = 2.5$).

For comparison, we estimate the densification exponent after the threshold by using linear regression of the data points in this interesting regime, denoted by $\hat{\alpha}$ (as shown in Figure 5.4). We observe that the estimated slope found through numerical evaluations is close to our previous result, i.e. $\frac{1}{\gamma-1}$. For example, for $\gamma = 1.5$, we have that $\hat{\alpha} = 2$, which is equal to our theoretical approximation of $1/(\gamma - 1)$, and for $\gamma = 1.8$, $\hat{\alpha} = 1.3$, which is again close to our prediction (1.25). Moreover, the value of cutoff, i.e., $n(s)^*$ can be computed theoretically using (5.17), which will be approximated by $E[D]$ in this case since the maximum degree is assumed to be equal to n. Using (5.27), we obtain two values of 3.16×10^3 and 96 for $\gamma = 1.5$ and 1.8 respectively. As seen in the plot (as dashed lines), the cutoff values closely match the numerical results.

Also, the final slope in the plot (not shown here) is close to the limiting theoretical slope, $\alpha(s \to 1)$, for different values of γ. It is interesting to observe that $\alpha(s \to 1)$ –which is the asymptotic slope– is not far from $\hat{\alpha}$. Indeed, this indicates that graphs where nodes follow a heavy-tailed distribution tend to saturate slowly, because they have many low-degree nodes. This results in an α-densification exponent that is not far from $1/f(1)$.

We should again emphasize that the interesting regime is when the edges and nodes are being discovered, while we are away from both early transient and late saturation phases. In this range, we observe a constant densification exponent over a wide range of $n(s)$. For a small sampling parameter, no densification is observed, whereas for large s we reach the asymptotic slope of $\alpha(s \to 1)$.

Figure 5.5 depicts the evolution of $\alpha(s)$ for Zipf distribution with different values of γ, computed using equations (5.10). We first observe that densification occurs for all sampling parameters ($\alpha(s) > 1$ for all $0 < s < 1$). More surprisingly, we observe that $\alpha(s)$ quickly converges to a nearly constant value as s increases, and finally reaches $\alpha(s \to 1)$. If this convergence occurs orders of magnitude before saturation of the discovery process, then the model yields an α-densification. Finally, although $\alpha(s \to 1)$ depends only on $f(1)$ (see equation (5.12)), the underlying degree distribution plays a role on determining if the edge sampling model will yield α-densification.

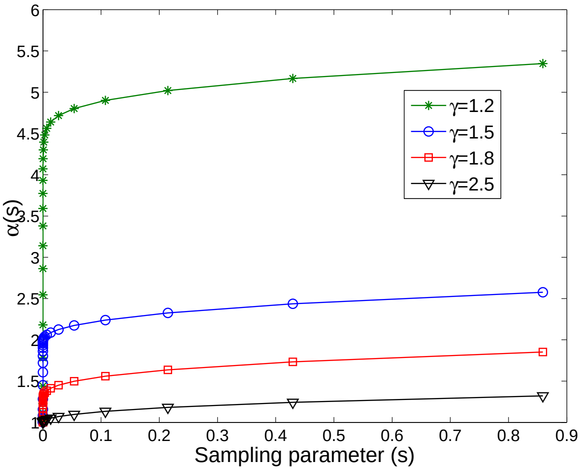


Figure 5.5: The densification exponent $\alpha(s)$ as a function of time for Zipf degree distributions ($n = 10^7$, $\gamma = 1.2$, $\gamma = 1.5$, $\gamma = 1.8$, $\gamma = 2.5$).

In order to support our claim, Figure 5.6 depicts the relationship between the estimated densification exponent $\hat{\alpha}$ and the degree exponent γ. As before, $\hat{\alpha}$ was computed through

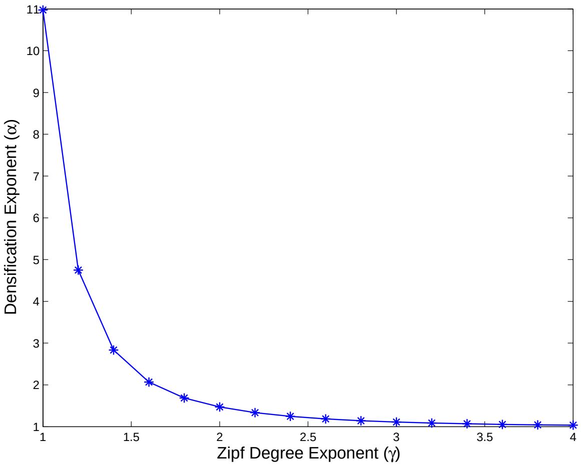


Figure 5.6: Relationship between the densification exponent (α) and the degree exponent (γ) for an underlying graph with a Zipf degree distribution ($n = 10^7$)

Table 5.1: Comparison of different parameters for different degree distributions

γ	s^*	$E[D](\simeq n(s)^*)$	$\alpha(s \to 1)$	$1/(\gamma - 1)$	$\hat{\alpha}$
1.5	1E-7	3161	2.6	2	2
1.8	1E-7	96	1.9	1.25	1.3
2.5	1E-7	3	1.3	–	1

a linear regression of the points in the interesting regime (i.e., densification regime). As illustrated in the figure, as γ increases large, specifically when γ is greater than 2, $\hat{\alpha}$ approaches 1. This supports our conjecture that α-densification occurs when $1 < \gamma < 2$.

A summary of the approximated values of cutoff, average degree, and the asymptotic/expected/estimated slope is shown in table 5.5. A basic comparison validates our theoretical results, and the accuracy of the obtained values for the densification behavior and cutoff. Basically, the estimated densification exponent $\hat{\alpha}$ (through linear regression) is quite close to the predicted exponent from theory ($\alpha = 1/(\gamma - 1)$). Moreoever, comparing the theoretical values of cutoff (s^* and $n(s)^*$) with the numerical results in Figures 5.3 and 5.4 well assesses the goodness of our theoretical approximations in Section 5.3.

Finally, we examine the robustness of edge sampling model through truncating the degree distribution from left and/or right. We show that the exponent for α-densification is only a function of the *degree exponent* of the underlying graph, and truncating the distribution does not affect the observed exponent α. In other words, we claim that densification behavior in the regime of interest does not depend on the maximum or minimum degree (or in general, individual lags) of the underlying graph, and the same densification exponent will be present as far as the degree γ is preserved. In order to do this, we truncate the distribution from left and right (resulting in lower maximum degree and higher minimum degree) without changing the behavior in the middle. Two scenarios are considered: In the first case, after truncation we rescale the distribution to distribute the mass over all points. In the second case, all the mass (either from the truncation point up to n for maximum degree, or from 1 to truncation point for minimum degree) is put at the truncation point.

Figure 5.7 depicts the edge-node curve of the same Zipf distributions (before saturation point), but truncated at left and right at the values of 5 and 8×10^6 respectively, i.e., $d_{\text{min}} = 5$ and $d_{\text{max}} = 8 \times 10^6$ (instead of the default values of 1 and $n = 10^7$).

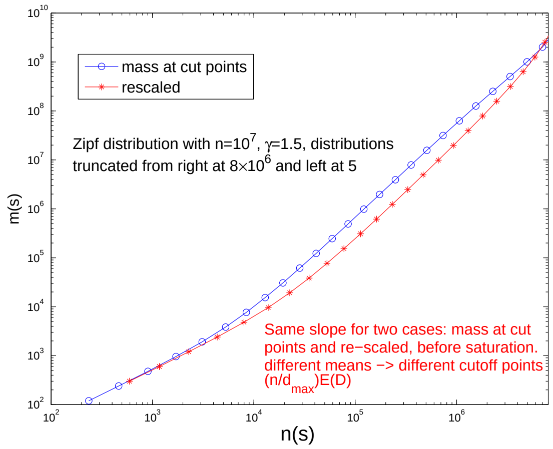


Figure 5.7: The number of nodes discovered ($n(s)$) versus the number of edges discovered ($m(s)$) for truncated Zipf degree distribution with $\gamma = 1.5$ ($n = 10^7$)

As shown in the figure, we observe two parallel lines for two cases of rescaling the distribution after truncation or putting the mass at truncation points. Furthermore, this slope is equal to the original densification exponent $\alpha = \frac{1}{\gamma - 1}$. Thus, the edge sampling model is robust in the sense that the densification behavior remains the same when truncating the distribution, while keeping the same degree exponent. However, as can be observed in Figure 5.7, the cutoff point (and thus the densification regime) will be different, because the average degree and the maximum degree are different in the two truncation processes. Also note that the left lag has an effect on the saturation regime, where moving the lag farther above zero results in an infinite densification exponent at saturation (i.e., $\alpha(s \to 1) = \infty$, since $f(1) = 0$). This was previously confirmed by theoretical results as well (but not shown here in the plot). The same result holds for other values of γ.

In summary, the edge sampling model has a single parameter s that varies between zero and one and is used to sweep the edge-node curve. As s approaches zero, the slope of the curve approaches 1, thus no densification is present. However, for sufficiently large s, it is possible that the curve follows a power-law, thus exhibiting α-densification. Moreover, this cutoff point for a sufficiently large s depends on the the average degree and maximum degree of the underlying graph. In particular, for a power-law degree distribution, a larger skew (i.e., smaller γ) requires a larger cutoff (i.e., a larger s). However, a larger skew also means that the densification exponent will be larger (since $\alpha = 1/(\gamma - 1)$). If the degree distribution is not skewed enough (i.e., if $\gamma > 2$), then no cutoff exists, meaning that the slope is always 1, and thus, no densification is present. Finally, the densification exponent is a function of the tail exponent of the distribution, and is robust to truncation through rescaling and putting mass at cut points.

5.5.1 Observation Models

In the previous section we evaluate the edge sampling model numerically, which has a single parameter, namely, the edge sampling probability, s. Before applying the model to real data, we comment on the relationship between how datasets of real networks are gathered, and also on the choice of the parameter s. In particular, considering how these datasets are obtained, we believe that the parameter s varies for either of the following reasons:

Accumulation In this interpretation of the sampling model, knowledge about the edges and nodes of the underlying graph is accumulated over time. This captures studies where new edges (and consequently nodes) are added to an evolving graph; this is the case, for example, in arXiv citation graph, or IMDB actors-to-movies graph (analyzed in [63]). Thus, the key parameter of this model is time, which will determine the probability that edges and nodes are discovered, i.e. $s = f(t)$. As time grows from zero to infinity, the sampling parameter varies from zero to one, resulting in the gradual discovery of edges and nodes.

We assume that any edge e is associated with an independent renewal process of rate λ, started in steady state. We define then $F_F(t, \lambda)$ to be the inter-sampling time cumulative distribution for an edge with a sampling rate of λ. Moreover, the sampling rate of an edge is also a random variable with cumulative distribution $F_R(\lambda)$, identical for all edges of G. We will also assume that sampling rates are chosen independently and that edges are also sampled independently of each other.

Modulation In this interpretation of the model, a *fixed time window* is considered. In other words, the sampling parameter s, is a function of a global sampling rate, which determines the rate with which the edges fire within a given fixed length time window. Different time windows can have different intensities with respect to the edge sampling rates. The motivation behind this interpretation is that in the analysis of real data we often consider fixed windows of time over the dataset, as opposed to an increasing time interval. However, different windows of time may have different sampling rates. For example, consider the study of e-mail networks and a time window of one month. It is natural that different months will have different intensities of e-mail exchange. For example, a vacation month will surely have a lower sampling rate than an end-of-semester month, in the e-mail network of a large university.

As before, we introduce an inter-sampling time distribution. However, here we define a modulating intensity parameter Λ; it influences the edge sampling rates. So we have $s = f(\lambda)$. In particular, the edge sampling rate will be given by

$$\lambda = \Lambda \lambda_o \tag{5.28}$$

where λ_o is a random variable that follows the original edge sampling rate cumulative distribution, $F_R(\lambda)$. Note that the actual edge sampling rate distribution is simply the original edge sampling rate distribution scaled by a constant Λ.

Notice that we are only giving different interpretations to s. In particular, we let s be controlled either by t or by Λ, which will be the case when we apply the edge sampling model to real datasets.

5.6 Real Data Experiments and Variations of the Model

In this section, we apply the edge sampling model to two datasets of real network data. The goal is to illustrate that the sampling model can capture and thus partially explain, the observed densification when considering data from these networks. In order to do this, we consider the two variations of the model, i.e. accumulation and modulation. Thus, the sampling parameter s will be a function of time or intensity, respectively, and is determined through the choice of this function, as described below.

Unfortunately, we do not have the real underlying graph to drive the edge sampling process. We also have no knowledge of the edge sampling probability s, which can be characterized by the edge sampling rates and the inter-sampling time distribution. Therefore, in order to apply the model we propose, we will use the actual dataset to derive estimates for these unknown parameters.

The underlying graph $G = (V, E)$ we consider is given by the accumulated graph over the entire dataset, i.e.,

$$G[0, t_f] = (V[0, t_f], E[0, t_f]),$$

where t_f is the length of the available dataset. Thus, we will assume that the degree distribution of the underlying graph is given by the empirical degree distribution of $G[0, t_f]$, namely $\bar{f}(k)$.

Concerning the edge sampling process, we will assume that the edge sampling rate for each edge is given by its average over the entire dataset (or its average over a time window, for the case of modulation). Thus, for each edge $e \in E$, we define $\lambda_e = |M_{ij}[0, t_f]|/t_f$, where edge $e = (i, j)$ and $M_{ij}[0, t_f]$ is the set of samples of the edge (i, j) available in the dataset in the period $[0, t_f]$ (same argument valid for a fixed time window). Finally, we assume that the inter-sampling times of an edge $e \in E$ is exponentially distributed with an associated rate λ, i.e. the sampling parameter s will be a function of time and rate. Thus,

$$F_F(t, \lambda) = s(t, \lambda) = 1 - e^{-\lambda t}. \tag{5.29}$$

To apply the sampling model to real data, we will consider various versions of the model, from the most parsimonious model to the most detailed use of information from real data. In all cases, the goal is to find the edge-node curve, to investigate whether a power-law relationship exists between the number of edges and nodes, and to compare the estimated exponent with that of the real data to verify the edge sampling model. We suggest the following experiments:

Simple Edge Sampling Model - Accumulation This is the simplest case where the only parameters we use from the data are the empirical degree distribution of the underlying graph, $\bar{f}(k)$, with n nodes and m edges, and the total number of exchanged messages in $[0, t_f]$. For the rate, we assume all edges have the *same rate* equal to the overall average message rate per edge, i.e. $\lambda = |M[0, t_f]|/t_f/m$. Thus by fixing λ and changing t from 0 to t_f, we derive the edge sampling probability for each t using (5.29). We can then derive different points in edge-node curve for each value of s using the following equations:

$$\begin{aligned} m(s) &= n\bar{d}s/2 \\ n(s) &= n \sum_{k=0}^{\infty} \bar{f}(k) \left[1 - (1-s)^k\right]. \end{aligned} \tag{5.30}$$

Note that here s is a function of t, and λ is fixed ($s(t) = 1 - e^{-\lambda t}$).

Simple Edge Sampling Model - Modulation In this case, we consider a fixed time window (thus t is fixed to 1). The parameters we use from data include the empirical degree distribution of the underlying graph $\bar{f}(k)$, and the number of exchanged messages at each window of time, i.e., $|M[t,t]|$ (which would yield the total number of messages $|M[0,t_f]|$ as well). Then, for *each* time window, we estimate an average message rate, same for all edges, using the edge sampling intensity of that window, i.e. $\Lambda_t = |M[t,t]|/\Lambda$, where $\Lambda = |M[0,t_f]|/t_f$ is the overall average message rate (per time window). Thus the average message rate per edge for each time window will be $\lambda = \Lambda_t/m$, fixed for all edges in time-stamp t, and we can again use the empirical degree distribution $\bar{f}(k)$ and equations (5.29) and (5.30) to derive the points in edge-node curve. Note that here s is a function of λ and t is fixed to 1 ($s(\lambda) = 1 - e^{-\lambda t}$).

Edge Sampling Using Rate Distribution Here we use both empirical degree distribution $\bar{f}(k)$ and the rate distribution $f_R(\lambda)$ of the underlying accumulated graph. So, we are using more detailed information from the data: the rate distribution. More specifically, rather than considering the *average* number of exchanged messages (overall or per time window), we assume that we know the empirical edge-sampling rate distribution, i.e., the distribution of the number of messages M_{ij} exchanged over the whole period of $[0,t_f]$. Then, for each time instance t, we find $m(s)$ and $n(s)$ by using the degree distribution and equations (5.30), knowing that here s can be derived by unconditioning on λ as follows:

$$s = \int_0^\infty (1 - e^{-\lambda t}) f_R(\lambda)\, d\lambda \tag{5.31}$$

Now for the case of accumulation, by changing the parameter t (and thus s), we can derive the edge-node curve using equations (5.30) and (5.31). Note that for the modulation case, we must have for each time window a different rate distribution, and we set t to 1. The rest of the argument is the same as above, only that different points on the edge-node curve are obtained for different rate distributions for each time window.

Edge Sampling Using Exact Structure In this case, we use the *exact structure* of the underlying accumulated graph, i.e. again more information of the underlying graph. In other words, instead of using the degree/rate distribution of the edges, we consider *every* edge and calculate its empirical rate λ_e (proportional to the sum of all the messages the edge has exchanged over the period of the dataset). We iterate over all edges and nodes of the underlying graph, and sum up their probability of being sampled using their exact rates to obtain the total number of edges and nodes. This way we derive the edge-node curve by varying either time instant t (and fixing λ_e for *each* edge), i.e., accumulation, or finding λ_e of each edge for each time window (and fixing t to 1), i.e., modulation. This can be summarized as follows:

$$\begin{aligned} m(t) &= \sum_{e \in E} (1 - e^{\lambda_e t}) \\ n(t) &= \sum_{v \in V} (1 - e^{\lambda_v t}), \end{aligned} \tag{5.32}$$

where λ_v corresponds to the rate of node v, equal to the sum of the rates of its incident edges, i.e.

$$\lambda_v = \sum_{e \in \mathcal{N}(v)} \lambda_e, \tag{5.33}$$

with $\mathcal{N}(v)$ denoting the neighborhood of node v.

Again note that for accumulation, the parameter t will be changed, whereas λ_e is fixed for *each* edge (but different for all edges), and for modulation, t is fixed and λ_e's change for every time window.

In this dissertation, we show the results for the first two experiments, i.e., *simple edge sampling model*, by using accumulation and modulation, and we show how well the edge sampling model works for modeling the densification behavior. We leave further experiments on the model to future work. We use an EPFL e-mail dataset for accumulation and modulation models, and Autonomous Systems data only for modulation.

5.6.1 Simple Accumulation Model

We will consider an EPFL e-mail dataset described in Section 5.2. Recall that this dataset leads to α-densification, as illustrated in Figure 5.1. Using the same dataset, we obtain the necessary parameters to construct the edge sampling model, as described above. The underlying graph is formed through the accumulation of edges and nodes over a 89-week period, and the same overall average message rate (per edge) is applied for all edges. Finally, we numerically compute $n(t)$ and $m(t)$ for this specific model. As with the original dataset, t is measured in weeks and varies from 1 to 89.

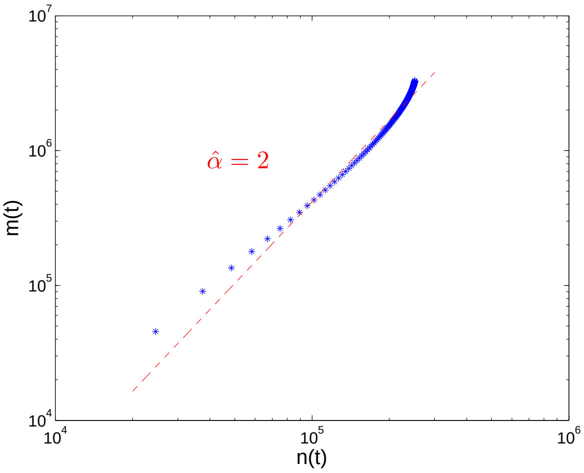


Figure 5.8: The edge sampling model applied to the EPFL e-mail network dataset.

The results produced by the model are shown in Figure 5.8. The plot clearly indicates that the edge sampling process leads to the densification of the number of edges versus nodes that are discovered over time. Moreover, the densification produced by the model seems to follow a power law, with $\hat{\alpha} = 1.8$ (recall that $\hat{\alpha}$ is the slope of the line obtained through a linear regression over the points in the plot). Surprisingly, this exponent closely matches the actual data ($\hat{\alpha} = 1.57$, see Figure 5.1), which indicates that the edge sampling process is a very plausible explanation for the observed densification.

To assess the goodness of the fitted slope, we consider the confidence interval for the slope of the line obtained through linear regression in Figure 5.1. Using simple statistical tools, we obtain the 95% confidence interval for the slope of the fitted line. For EPFL e-mail data set, we find a confidence interval of $(1.55, 1.57)$ with the slope being $\hat{\alpha} = 1.57$. Thus, the estimated slope is well above 1 with high probability, which shows a clear densification as the network grows. The same is true when the edge sampling model is applied to real data, thus resulting in a slope of sufficiently larger than 1 with small error.

5.6.2 Simple Modulation Model

In this section we will apply the edge sampling model over a fixed window of time (proposed in Section 5.5.1 as *modulation*) to real data. Recall that the basis for this model is that real data is often provided in snapshots over a fixed window of time. Moreover, even when this is not the case, data analysis often considers data only over a fixed window of time, varying the position of the window over the dataset.

EPFL E-mail Dataset

We start by presenting the results for the actual EPFL e-mail dataset. Recall that the EPFL e-mail dataset is provided in snapshots of weeks (i.e., all messages that were sent/received during a given week). Thus, for each week $t = 1, \ldots, 89$, we can define the graph $G[t,t] = (V[t,t], E[t,t])$, which is defined using only messages exchanged during that week. For each graph, we let $n(t) = |V[t,t]|$ denote the number of nodes and $m(t) = |E[t,t]|$ denote the number of edges. Differently from before, notice that $n(t)$ and $m(t)$ correspond to the number of nodes and edges respectively, seen on week t only.

Figure 5.9 shows the plot of $n(t)$ versus $m(t)$ for $t = 1, \ldots, 89$ for the actual EPFL e-mail dataset. Interestingly, we again observe densification despite the fact that information about the graph is not accumulated over time. Thus, densification arises here for reasons other than accumulation over time, as each point corresponds to exactly one week. Moreover, there is no trend between time and the points in the plot. Although the points do not form a straight line, there is a clear increasing trend among them. When fitted to a straight line, we obtain a slope of $\hat{\alpha} = 1.89$, as illustrated in the figure.

As before, we find the confidence interval for the estimated slope, which in this case is $(1.65, 2.12)$. Although the intervals do not express tight bounds, we can still claim that with high probability we observe a clear densification.

In order to apply the simple edge sampling model through modulation, we again need to determine its parameters. Besides the underlying graph and the sampling process, we also need to determine the edge sampling intensity. We will estimate as before the actual underlying graph and sampling process, by considering the graph accumulated over the entire dataset and the average rate of messages per edge. We still need to determine, however, the edge sampling intensity for each fixed time window.

Let Λ_t denote the edge sampling intensity of week t. We define $\Lambda_t = |M[t,t]|/\Lambda$, where $M[t,t]$ is the set of messages in the dataset during week t and Λ is the overall average message rate (per week). In particular, we define $\Lambda = |M[1,89]|/89$. Thus, Λ_t measures the average intensity of the edge sampling rate of week t in relationship to the overall average edge

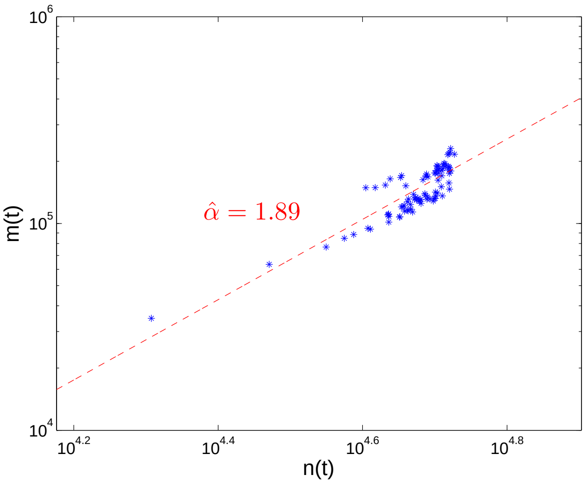


Figure 5.9: Densification of EPFL e-mail dataset using a fixed time window (1 week).

sampling rate. In other words, the sampling parameter s will be a function of the modulating intensity parameter in this case.

We can now apply the edge sampling model with a fixed time window. Notice that each week t corresponds to a different time window with edge sampling intensity Λ_t. Figure 5.10 shows the expected number of nodes $n(\Lambda_t)$ versus the expected number of edges $m(\Lambda_t)$ discovered for each week t. It is clear that the model yields densification when considering the different weeks. Moreover, there is strong linear dependence between $n(\Lambda_t)$ and $m(\Lambda_t)$ indicating a power law relationship. Finally, a straight line fitted to the data yields the slope $\hat{\alpha} = 1.5$, which is fairly close to the actual dataset, with a confidence interval far above 1. This supports the claim that the proposed edge sampling model captures the discovery process of edges and nodes.

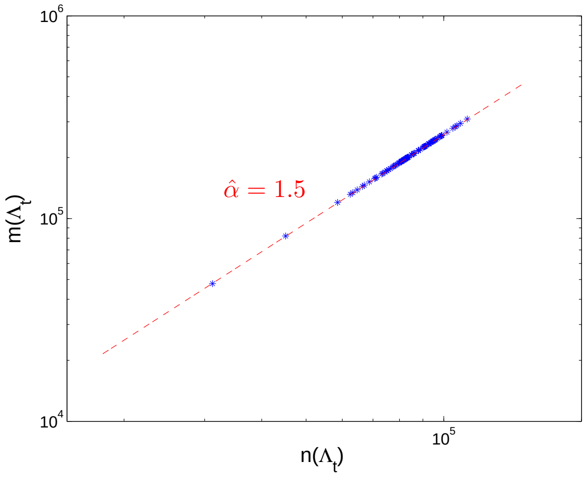


Figure 5.10: The edge sampling model with fixed time window applied to EPFL e-mail dataset (1 week).

AS Graph Dataset

In this section we consider another dataset of network data that is publicly available, the AS graph. The Internet today is composed of several thousands Autonomous Systems (ASs) that interconnect with each other providing global connectivity. Most of ASs are owned and operated by Internet Service Providers (ISPs) such as AT&T or MCI, and other ASs belong to smaller businesses or universities. The interconnection of ASs forms a graph, which is known as the AS graph.

Information about the AS graph is provided daily through snapshots generated from an aggregate of information about the network. However, for a myriad of reasons, these daily snapshots provide only an estimate of the real AS graph. Moreover, the real AS graph constantly changes, as both ASs and their interconnections are added and deleted over time.

We consider a dataset of the AS graph formed by 735 daily snapshots. As for the EPFL e-mail dataset, each snapshot defines a graph $G[t,t] = (V[t,t], E[t,t])$ formed by all nodes and edges that appear in day t. For each day t, we thus have a number of nodes $n(t) = |V[t,t]|$ and a number of edges $e(t) = |E[t,t]|$.

Figure 5.11 shows the result of plotting $n(t)$ versus $e(t)$ in log-log scale for all days. Although the number of nodes and edges vary little from one day to another, the plot still strongly indicates a power law densification with $\hat{\alpha} = 1.09$.

To assess the goodness of this fit, we calculate the confidence interval for the slope, which yields the interval $(1.08, 1.1)$. Thus although $\hat{\alpha}$ is close to 1, with high probability it will be a value above 1, resulting in densification.

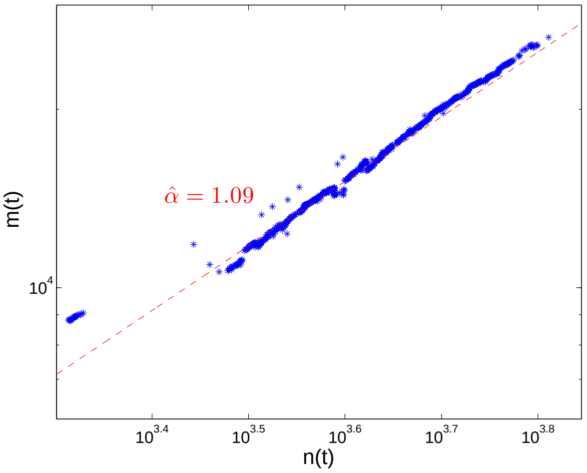


Figure 5.11: Densification of the AS graph dataset using a fixed time window (1 day).

We will apply the edge sampling model with fixed time window to the AS graph dataset. Notice that since each snapshot provides full view of the AS graph, the edge sampling model where information is accumulated over time is not suitable for this dataset. However, in order to apply the fixed time window model we need to estimate its parameters from this dataset. We execute the same procedure used for EPFL e-mail dataset, estimating the fixed underlying graph and the edge sampling rates using the accumulated graph[3]. We also determine in the

---

[3] In the AS graph dataset, for each snapshot, each edge has either rate 1 (i.e., edge is present in the AS graph) or 0 (i.e., edge is not present).

same manner, for each daily snapshot, the edge sampling intensity.

Figure 5.12 shows the result obtained with the model for the various sampling intensities Λ_t. Once again, we observe densification on the number of nodes and edges discovered when considering each fixed time window. Moreover, there is a roughly linear trend in this relationship, indicating a power law densification, with $\hat{\alpha} = 2$, as illustrated in the plot.

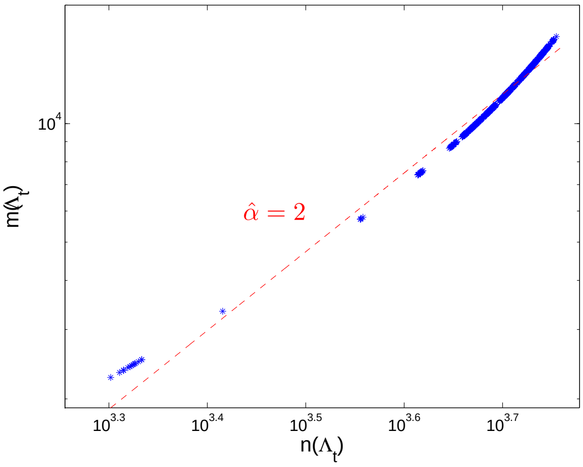


Figure 5.12: The edge sampling model with fixed time window applied to the AS graph dataset 1 day).

Although the slope of the fitted line ($\hat{\alpha}$) yielded by the model is not very close to that of the actual data, we still believe the model is representative of how nodes and edges are revealed in this dataset, and we observe a clear densifying behavior.

There can be different reasons the model does not match the data accurately. First, the real underlying graph is unknown, and it also changes in time. In order to apply the model to real data and do the simulations, we made the assumption of having the degree distribution of the final accumulative graph for the underlying graph. As seen before, the degree distribution of the underlying graph is a key factor in the densification behavior and the observed exponent. Thus, this assumption surely affects the result when edge sampling is applied to real data. Furthermore, in applying the model to fixed time windows, we estimated an average intensity through finding the ratio of total number of messages in a fixed window and the total number of messages in the final accumulative graph. How accurate this estimation is still needs to be investigated on more datasets. In general, the mismatch with data is mostly due to the fact that the *simple* edge sampling model uses minimal information from data, and the use of more detailed information such as rate distribution or exact structure would increase the accuracy.

We also applied the proposed models to the exact structure of the underlying graph (where the underlying graph is given by the final cumulative graph), as opposed to considering only the empirical degree distribution (i.e., the last experiment suggested in the beginning of Section 5.6). Using the edge sampling model with exponential inter-sampling time distribution, we observe that results from the model match real data very accurately for both cases of accumulation and modulation. This further strengthens our claim that densification arises regardless of the process used for sampling the edges.

We have considered a couple other datasets analyzed by Leskovec et al. [63], namely the

IMDB actors-to-movies bipartite graph and arXiv citation graph. However, these datasets are not good examples to which the proposed edge sampling model can be applied. To be more precise, in the citation graph, nodes (i.e., papers) in the graph appear first, and edges (i.e., citations) appear afterwards, when a given paper cites another. Thus, in this dataset, new nodes are not revealed through sampling the edges of the graph. In the IMDB dataset, movies and actors correspond to the two different types of nodes of a bipartite graph. An edge between a given pair of nodes is present when a given actor plays a role in a given movie. Thus, each edge is sampled only once (i.e., when an actor plays in a new movie), as the edge will never be sampled again, giving rise to a constant edge sampling rate for all edges. For these reasons, in this work, we do not consider these datasets in the framework of the proposed edge sampling model, and leave experiments on more datasets to future work.

5.7 Related Work

Network (or graph) densification is a phenomenon that was recently observed by Leskovec et al. in various datasets of real-data over time [61, 63]. In particular, they empirically observe that the number of edges grows much faster than the number of nodes in these graphs. More surprising, for all datasets considered is that this relationship seemed to follow a power law of the type

$$m(t) \propto n(t)^{\alpha},$$

where $m(t)$ and $n(t)$ denote the number of edges and nodes of the graph at time t, and α is called the densification exponent, a constant greater than 1. Under this relationship, known as the *densification power law*, the average node degree grows with time, thus the graph densifies.

Most of the graphs considered in their work were generated by accumulating the data available in the datasets over time. However, they also considered the case where the graph is generated by considering only fixed windows of time. This is the case for the AS graph and the e-mail network. For the e-mail network, they observe that the densification exponent is larger for the fixed time window case, which is consistent with our observations.

Besides providing empirical evidence of network densification, Leskovec et al. also provide mathematical models that can capture and partially explain this phenomenon [61, 63]. The intuition behind their models is that densification occurs due to the growth of the network. Basically, nodes arriving at a later point tend to establish more edges than nodes arrived earlier. The proposed *Forest Fire Model* is based on this intuition and is a model for network growth, where nodes and edges are added to the graph over time. They also prove that this model leads to the densification of the network over time. Leskovec et al. also investigate other growth models for networks, showing how properties such as power law densification can arise, in particular when using the model of Kronecker Graphs [57, 54].

An observation similar to the findings of Leskovec et al. concerning network densification was also previously made by Dorogovtsev and Mendes [22]. They empirically observed that the graph formed by the World Wide Web was densifying over time, naming this phenomenon *accelerated growth*. They also propose a model to capture this phenomenon where the average node degree grows as a power law in time.

5.8 Summary

In this work we investigate the recently observed phenomenon known as *densification*, where the number of edges of a network grows much faster than the number of nodes. We provide, in particular, a novel explanation for this phenomenon when considering some real datasets. The key idea is that densification arises naturally from the process used to reveal the edges and nodes of the unknown underlying graph. For most datasets, this process is based on the discovery of edges that are then used to discover nodes. Based on this idea, we propose the *edge sampling model*, where edges from a fixed underlying graph are sampled by changing a sampling parameter s, leading to their discovery and to the discovery of adjacent nodes.

By assuming a heavy-tailed degree distribution for the underlying graph, we prove properties concerning the densification of the number of edges and nodes discovered over different values for the sampling parameter. In particular, we show that a power law relationship, which we call α-densification, can arise over a wide range of scales with exponent given by $\frac{1}{\gamma-1}$, where γ is the power-law exponent of the node degree distribution of the underlying graph. We also prove limiting results for densification exponent as s approaches both extremes (i.e., 0 and 1), and show that densification can be present and it is bounded for all $0 < s < 1$.

Furthermore, we show that a graph densifies if and only if its underlying degree distribution is of the form of a power-law. This is the case for a wide range of real-world graphs and suggests that densification can be observed in most real networks.

We also comment on the relationship of empirical network studies with the edge sampling model, and specifically, on what factors affect the sampling probability s. We introduce two observation models, namely *accumulation* and *modulation*. In the first variation, edges and nodes are discovered and accumulated over time; in the second variation, time is fixed but the edge sampling intensity for a given time window is allowed to vary.

Finally, we consider datasets of real graphs, namely the EPFL e-mail network and the Internet AS graph, both of which show densification over time. We apply the two variations of our model to these datasets and the results obtained indicate that the edge sampling model is indeed a plausible alternative explanation for the observed densification phenomenon.

Publication [84]

Pedram Pedarsani, Daniel R. Figueiredo, and Matthias Grossglauser. **Densification Arising from Sampling Fixed Graphs**. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '08, pages 205–216, 2008.

5.A Appendix

Here we find upper and lower bounds for equation (5.10), i.e.

$$\alpha(p) = \frac{(1-p)\left(1 - \sum_k \bar{f}(k)(1-p)^k\right)}{p \sum_k \bar{f}(k)k(1-p)^k}$$

We first consider the upper bound. For $p > 1/2$, we have $1 - p < p$. Thus we can write

$$\frac{(1-p)\left(1-\sum_k \bar{f}(k)(1-p)^k\right)}{p\sum_k \bar{f}(k)k(1-p)^k} \overset{(a)}{\leq} \frac{1-p}{\sum_k \bar{f}(k)k(1-p)^k} = \frac{1}{\bar{f}(1)+\bar{f}(2)(1-p)+\cdots} \leq \frac{1}{\bar{f}(1)}$$

where (a) follows as the second term in the numerator is $P(v)$, and thus less than 1.

Now we consider the lower bound. We can rewrite $\alpha(t)$ as

$$\alpha(t) = \underbrace{\left(\frac{1-p}{\sum_k \bar{f}(k)k(1-p)^k}\right)}_{Q}\underbrace{\left(\frac{1-\sum_k \bar{f}(k)(1-p)^k}{p}\right)}_{T}. \tag{5.34}$$

Assuming $\bar{f}(0) = 0$ (which is the case for Zipf distribution), and using $\sum_k \bar{f}(k) = 1$, we have,

$$Q = \frac{\bar{f}(1)(1-p)+\bar{f}(2)(1-p)+\bar{f}(3)(1-p)+\cdots}{\bar{f}(1)(1-p)+\bar{f}(2)\cdot 2e^{(}1-p)^2+\cdots} \tag{5.35}$$

Comparing the coefficients of $\bar{f}(k)$ in the numerator and denominator of (5.35) for all $k > 1$, we have,

$$(1-p) \geq k(1-p)^k, k = 2, 3, \ldots \quad \text{if } \ln\frac{1}{1-p} \geq \frac{\ln(k)}{k-1}. \tag{5.36}$$

Since $\frac{\ln(k)}{k-1}$ is a decreasing function of k, the condition $p \geq 1/2$ (derived by substituting $k = 2$) suffices to have each term in the numerator equal or greater than the corresponding term in the denominator, i.e.

$$\bar{f}(k)(1-p) \geq \bar{f}(k)k(1-p)^k, k = 1, 2, 3, \ldots \quad \text{if } p \geq 1/2, \tag{5.37}$$

which results in $Q \geq 1$. Note that for $k = 1$ the two corresponding terms are equal, thus satisfying (5.37).

Following the same approach for T, we have,

$$T = \frac{1-\overbrace{\left(\bar{f}(1)(1-p)+\bar{f}(2)(1-p)^2+\cdots\right)}^{Y}}{1-\underbrace{\left(\bar{f}(1)(1-p)+\bar{f}(2)(1-p)+\cdots\right)}_{Z}}. \tag{5.38}$$

Comparing the coefficients of $\bar{f}(k), k = 1, 2, \ldots$ in Y and Z, we observe,

$$\bar{f}(k)(1-p)^k \leq \bar{f}(k)(1-p), k = 1, 2\ldots, \tag{5.39}$$

which results in $Z \leq Y$, yielding $T \geq 1$. Thus,

$$\alpha(t) = Q \cdot T \geq 1, \text{ if } p \geq 1/2, \tag{5.40}$$

which proves the lower bound in equation (5.11).

Putting all together, we proved that for $p \geq 1/2$,

$$1 \leq \alpha(p) \leq \frac{1}{\bar{f}(1)}. \tag{5.41}$$

Next, we prove the two limiting results for $\alpha(p)$ as defined in equation (5.10). Define $\alpha(p \to 0)$ to be the limit of $\alpha(p)$ when p goes to zero. Thus,

$$\begin{aligned}
\alpha(p \to 0) &= \lim_{p \to 0} \alpha(p) \\
&= \lim_{p \to 0} \frac{(1-p)\left(1 - \sum_k \bar{f}(k)(1-p)^k\right)}{p \sum_k \bar{f}(k) k (1-p)^k} \\
&= \lim_{p \to 0} \frac{1 - \sum_k \bar{f}(k)(1 - pk + \cdots)}{p \sum_k \bar{f}(k) k (1 - pk \cdots)} = 1
\end{aligned}$$

where we used Taylor's expansion for $(1-p)^k$.

Now define $\alpha(p \to 1)$ to be the limit of $\alpha(p)$ when p goes to 1. Thus,

$$\begin{aligned}
\alpha(p \to 1) &= \lim_{p \to 1} \alpha(p) \\
&= \lim_{p \to 1} \frac{(1-p)\left(1 - \sum_k \bar{f}(k)(1-p)^k\right)}{p \sum_k \bar{f}(k) k (1-p)^k} \\
&= \lim_{p \to 1} \frac{1 - \sum_k \bar{f}(k)(1-p)^k}{p} \cdot \frac{(1-p)}{\sum_k \bar{f}(k) k (1-p)^k} \\
&= \lim_{p \to 1} \frac{1 - \sum_k \bar{f}(k)(1-p)^k}{p} \cdot \frac{1}{\sum_k \bar{f}(k) k (1-p)^{k-1}} \\
&\overset{(b)}{=} \left(1 - \bar{f}(0)\right)\left(\frac{1}{\bar{f}(1)}\right) \\
&= \frac{1 - \bar{f}(0)}{\bar{f}(1)} \qquad (5.42)
\end{aligned}$$

where (b) follows because in the first term of the product, the denominator goes to one and all terms in the sum go to zero except for $k = 0$; in the second term of the product, all terms in the sum go to zero except for $k = 1$.

Note that when $\bar{f}(0) = 0$, equation (5.42) simplifies to $\frac{1}{\bar{f}(1)}$. This simplification applies in the case of the Zipf distribution, where $\bar{f}(0) = 0$ independent of its parameter s and n. Moreover, for the Zipf distribution, we have that $\bar{f}(1) = 1/A$, where A is the normalization factor of the distribution, which is given by

$$A = \sum_{i=1}^{n} 1/i^s = 1 + \frac{1}{2^s} + \frac{1}{3^s} + \cdots + \frac{1}{n^s}. \qquad (5.43)$$

Thus, for the Zipf distribution, we have that $\alpha(p \to 1) = 1/\bar{f}(1) = A$.

Part III

Preserving Privacy in Recommender Systems

6 Preserving Privacy in Collaborative Filtering

In recommender systems, usually, a central server needs to have access to users' profiles in order to generate useful recommendations. Having this access, however, undermines the users' privacy. The more information is revealed to the server on the user-item relations, the lower the users' privacy is. Yet, hiding part of the profiles to increase the privacy comes at the cost of recommendation accuracy or difficulty of implementing the method. In this chapter, we propose a distributed mechanism for users to augment their profiles in a way that obfuscates the user-item connection to an untrusted server, with minimum loss on the accuracy of the recommender system. We rely on the central server to generate the recommendations. However, each user stores his profile offline, modifies it by partly merging it with the profile of similar users through direct contact with them, and only then periodically uploads his profile to the server. We propose a metric to measure privacy at the system level, using graph matching concepts. Applying our method to the Netflix prize dataset, we show the effectiveness of the algorithm in solving the tradeoff between privacy and accuracy in recommender systems in an applicable way.

6.1 Introduction

Recommendation systems are widely used to help users, overwhelmed by the huge number of options available to them, to find items that they might like. The items can be of any type: books, movies, web pages, restaurants, sightseeing places, online news, and even lifestyles. By collecting information about users' preferences for different items, a recommender system creates user *profiles*. The preferences of a user in the past can help the recommender system to predict other items that might also be of interest to the user in the future. Collaborative filtering (CF), as one of the main categories of recommender systems, relies on the similarities of users' tastes: if two users have similar preferences for certain items, each user probably enjoys the items of interest to the other. Thus, the more each user gives information about his interests, the more meaningful the recommendations will be. This is the basis of CF systems.

In order to run the process of recommending items to users, recommender servers need to have access to users' profiles. Therefore, the profiles are usually stored on repositories to which collaborative filtering algorithms can have access. In such systems, the users do not have technical measures to limit the amount of information on their profiles to the server, that might not be necessary for generating recommendations. In other words, users have to put their profiles online (i.e., on the server) and trust the server (and the service providers) to keep the users' profiles private. The information available to the server hurts the privacy of the users on two levels.

First, if a user's real identity is available to the server, the server can associate the user's profile, which contains his private information, to his real identity (e.g., Amazon knows the real identities and postal addresses of those who have purchased products and can link them with the profile of those users). This is an obvious privacy breach, considering that a user does not want the link between his real identity and his profile to be revealed, yet he wants to use the service. This threat becomes more obvious when users share their opinion about the locations they regularly visit, e.g., restaurants, libraries, coffee shops, or sport centers [3].

Second, even if the real identity of a user is not known to the server, it can try to de-anonymize the user's identity by correlating the information contained in the user's profile and some information obtained from other databases [77].

Obviously, hiding information from the server helps to thwart these threats. Nevertheless, users want to receive accurate recommendations. Hence, the tradeoff between privacy and accuracy appears. The more accurate information the server has about users' profiles, the more meaningful the server's recommendations are, but the lower the users' privacy will be.

Moving from centralized approaches to distributed collaborative filtering algorithms solves the privacy issues to a great extent [13, 69]. Yet, distributed CF algorithms are not as accurate as their centralized versions that have complete information about users' profiles. Moreover, they are not as practical as the centralized CF systems. The users' cooperation is needed not only to protect their privacy but also to make the system run properly. Making use of sophisticated cryptographic tools [18] might be another approach to solving the problem. However, these approaches are usually not practical and therefore remain unused. The security of these systems depends on that of their key establishment and key management. Considering that the perfect implementation of these schemes is hard to achieve in practice, they are not well received. Therefore, finding a more practical solution that does not trade much system accuracy in order to gain privacy for the users is an unsolved problem and yet crucial for users of CF systems.

In this chapter, we propose a method that provides a compromise for the tradeoff between privacy on the one hand and accuracy and practicality on the other hand. In other words, our method increases the privacy in a practical way with a negligible effect on the recommendation accuracy. To this end, we still rely on the central server to generate recommendations. Additionally, we employ a distributed communication between users in order to improve their privacy. In our model, each user has two versions of his profile: the *online* (on the server) and the *offline* (located on the user's side), where the online profile is frequently synchronized with the offline version. Basically, the actual profile of a user is a subset of his offline profile. Users' offline profiles are aggregated in order to obfuscate the actual items rated by each user. As we will show, the hybrid nature of our approach helps users to gain privacy from distributed aggregation and to reach a level of accuracy comparable to the one achieved with a centralized CF.

The remainder of this chapter is organized as follows. In Section 6.2, we define our notations and basic elements of a CF and state the problem, i.e., *privacy preserving in the presence of an untrusted server.* In Section 6.3, we provide an overview of the solution and elaborate our approach. In Section 6.4, we define evaluation metrics for privacy and recommendation accuracy. In Section 6.5, we validate the efficiency of our method by applying it to the real data set from the Netflix prize competition. Finally, in Sections 6.6 and 6.7, we review the related work and conclude.

6.2 Problem Statement

6.2.1 Definitions and Notations

Formally, a collaborative filtering (CF) algorithm deals with a set of users and a set of items. In our setting, the non-empty set of users in the system is denoted by $\mathcal{U}$, where $|\mathcal{U}| = N$. We also represent the non-empty set of items by $\mathcal{I}$, where $|\mathcal{I}| = M$. Let $r_{u,i}(t)$ be the rating of user u to item i at time t, where $r_{u,i}(t) \in \{1, 2, \cdots r_{max}\}$ and r_{max} is the maximum valid rating. The set of items rated by user u up to time t is denoted by $\mathcal{I}_u(t) \subseteq \mathcal{I}$. The *profile* of user u at time t is defined as $\{(i, r_{u,i}(t)) \text{ s.t. } i \in \mathcal{I}_u(t)\}$ which is the set of items coupled with their ratings rated by the user until t. We denote by $f_i(t)$ the *rating frequency* of item i at time t which is the fraction of users that have rated item i up to time t (i.e., $f_i(t) = |\{u \in \mathcal{U} \text{ s.t. } i \in \mathcal{I}_u(t)\}|/N$). To simplify the presentation, we sometimes omit the time index if it is clear from the context.

Collaborative filtering algorithms attempt to make predictions on the ratings of a particular user by collecting taste information from other users. CF algorithms fall into two general classes: *model-based* and *memory-based* methods [17]. Model-based algorithms learn a probabilistic model from the underlying data using statistical techniques, then they use the model to make predictions. Memory-based methods find similar users or items in the dataset in order to predict the items that a user might like and recommend them to him. This approach is based on the assumption that similar users prefer similar items, or that the preferred items of a user are similar. Memory-based CF methods can be further divided into two groups: *user-based* and *item-based* [96]. User-based methods, which are the focus of this work, are heuristics to predict a rating of a user to an item by combining the ratings of users who are most similar to the target user (so called, the user's *neighbors*). The Pearson's correlation coefficient, as a popular measure [42], estimates the similarity $w_{u,v}$ between two users u and v, as follows.

$$w_{u,v} = \frac{\sum_{i \in (\mathcal{I}_u \cap \mathcal{I}_v)} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in (\mathcal{I}_u \cap \mathcal{I}_v)} (r_{u,i} - \overline{r}_u)^2 \cdot \sum_{i \in (\mathcal{I}_u \cap \mathcal{I}_v)} (r_{v,i} - \overline{r}_v)^2}} \tag{6.1}$$

where, $\overline{r}_u$ is the mean of ratings assigned by user u.

In order to predict the rating of user u to item i, first, the similarity of u to all other users is computed and then the nearest neighbors of u, denoted by set N_u, are determined. The set N_u contains $|N_u|$ users who are, based on (6.1), the most similar users to u. Next, the prediction of $r_{u,i}$, denoted by $\hat{r}_{u,i}$, is done by combining the ratings of neighbors of u to item i [42, 90], as follows.

$$\hat{r}_{u,i} = \overline{r}_u + \frac{\sum_{v \in N_u} w_{u,v}(r_{v,i} - \overline{r}_v)}{\sum_{v \in N_u} w_{u,v}} \tag{6.2}$$

Finally, the items that have a high potential of interest to the user (i.e., are predicted to have high and positive ratings) are recommended to him.

6.2.2 Problem Definition

We consider the scenario where a collaborative filtering algorithm (described in Section 6.2.1) is implemented on a server and users give information about their profiles to the server in order to receive recommendations. We define the problem as finding a mechanism by which the users can adapt their profiles (available to the server) to the privacy level they expect from the system. The solution must have minimum effect on the system's accuracy. We assume the server to be untrusted, and we evaluate the level of privacy in the system with respect to that. Yet, the information revealed among users themselves, if there is any need for communication, must be under the users' control (i.e., what information is given and to whom). Intuitively, the system privacy is high if the server is not able to construct the users' profiles based on the information available to it.

6.3 Proposed Method

In this section, we first give an overview of our solution in Section 6.3.1, and we define a few new concepts that we need to describe our scheme. Next, in Sections 6.3.2 and 6.3.3 we formally model the problem and our proposed solution.

6.3.1 Sketch of the Solution

We assume there is a central recommender system where users' profiles are stored and from which the users receive recommendations. We call a user's profile stored on the server his *online profile*. We assume that a user can have a local repository where he stores his own profile; we call these locally stored profiles the *offline profiles*. The server does not have access to the users' offline profiles and the recommendations are produced based on the online profiles available to the server. Each user independently synchronizes his online profile with his offline profile. Therefore, from time to time, the changes that have been made to the offline profile, since the last synchronization, will be applied to the online version, all at once. Obviously, the recently rated items are among these changes. In addition to these actual ratings, users add other items to their profiles, which are not originally rated by them; instead, they are received from other users with whom they communicate.

Users communicate with each other through different media such as face-to-face communication in a meeting, communicating over cellular networks, instant messaging through the Internet, communicating via a social network, or exchanging e-mails. We refer to any such communication as *contact* between two users. A user arbitrarily selects his peers and certain information about the users' offline profiles is exchanged between them at each contact. Based on this information, they add a subset of the other user's items to their own profiles. The exchanged information between users, the number of items that are added to their profile and the items (plus their associated ratings) that are selected for the aggregation depend on the *aggregation* function they use.

From the server's point of view, each user adds a batch of new items (plus the ratings) to his online profile at each synchronization. It is not known to the server which of these items have been actually rated by the user. In other words, the user's actual profile is hidden from the server; hence, there is privacy for the user. This, of course, comes at a cost: a drop in the recommendation accuracy. In the next sections, we answer the following questions, in order to find the appropriate aggregation functions: How many items should be aggregated in each contact? What items are better candidates for aggregation? What is the effect of contact rate on the privacy and accuracy?

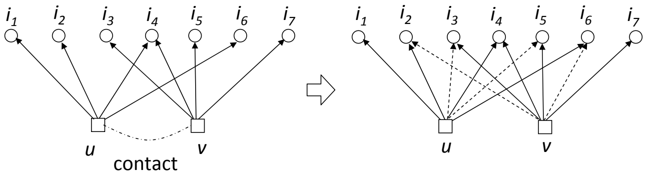


Figure 6.1: Aggregation of users' offline profiles after a contact. A contact occurs between users u and v at time t, with $\mathcal{I}_u^{\text{off}}(t) = \{i_1, i_2, i_4, i_6\}$ and $\mathcal{I}_v^{\text{off}}(t) = \{i_3, i_4, i_5, i_7\}$. User u gives $\{i_2, i_4, i_6\}$, which is a subset of his profile, to v. User v also gives $\{i_3, i_5\}$ to u. Additional edges (dashed lines) appear after the aggregation, and the offline graph is evolved through aggregating the new added edges. After the aggregation, $\mathcal{I}_u^{\text{off}}(t^+) = \{i_1, i_2, i_3, i_4, i_5, i_6\}$ and $\mathcal{I}_u^{\text{off}}(t^+) = \{i_2, i_3, i_4, i_5, i_6, i_7\}$.

6.3.2 The Model

We model users' profiles by a bipartite graph, where nodes denote the users and items, and weighted edges correspond to the ratings of the users to the items.

Let $G_t(U, V, E)$ be a bipartite weighted graph, where the vertex set $U = \mathcal{U}$ corresponds to the users, the vertex set $V = \mathcal{I}$ corresponds to the items, and the edge set $E(t)$ corresponds to the users' rating for the items until time t. Thus, an edge $(u, i) \in E(t)$ exists when user $u \in U$ has rated item $i \in V$ at some time before t. Each edge (u, i) has an associated weight $r_{u,i}(t)$ denoting the rating of user u assigned to item i. In such a setting, a user's profile or $\mathcal{I}_u(t)$ will be the set of nodes adjacent to node u. We call G_t the *actual* graph, because it represents the actual users' ratings of the items, i.e., in the case that there is no privacy preserving method in use.

Using our method, there will be two other graphs in the system. One should represent the users' online profiles and the other should model the users' offline profiles.

Let us denote the *offline* graph by $G_t^{\text{off}}(U, V, E^{\text{off}})$. This conceptual graph, which is stored in a distributed manner, is formed by the aggregation of edges through users' contacts and by rating new items over time until t. The edges that are added to the offline graph can appear either as a result of actual ratings by the users, or through aggregation by contacting other users. Hence, the actual graph G_t is embedded and hidden in G_t^{off}. We also denote $\mathcal{I}_u^{\text{off}}(t)$ to be the user's offline profile at t. By definition, for every user u at any time t we have $\mathcal{I}_u(t) \subseteq \mathcal{I}_u^{\text{off}}(t)$. Note that $\mathcal{I}_u^{\text{off}}(t) \setminus \mathcal{I}_u(t)$ is the set of items (plus their ratings) that are added through aggregation to a user's offline profile.

In our model, the *online* graph is denoted by $G_t^{\text{on}}(U, V, E^{\text{on}})$ and $\mathcal{I}_u^{\text{on}}(t)$ represents the online profile of user u. The online graph gets updated over time when users synchronize their online profiles with their offline versions. As a result, at any time, there might be some edges in graph G_t^{off} that have not yet been added to the graph G_t^{on}. Therefore, $\mathcal{I}_u^{\text{on}}(t) \subseteq \mathcal{I}_u^{\text{off}}(t)$, for every user u at any time t.

Figure 6.1 illustrates the effects of aggregation on the offline profiles of two users after their contact. Assume two users u and v contact each other at time t, with offline profiles $\mathcal{I}_u^{\text{off}}(t)$ and $\mathcal{I}_v^{\text{off}}(t)$ before the contact. We also denote $\mathcal{I}_u^{\text{off}}(t^+)$ and $\mathcal{I}_v^{\text{off}}(t^+)$ to be their offline profiles after contact. As it is shown, each user (e.g., u) gives a subset of his own profile to his peer (e.g., v) to be aggregated to the peer's previous profile. The way they select the mentioned subset is explained in the next section. Note that the added edges preserve their weights (ratings). In the case the receiver is given an item that is already in his offline profile, he

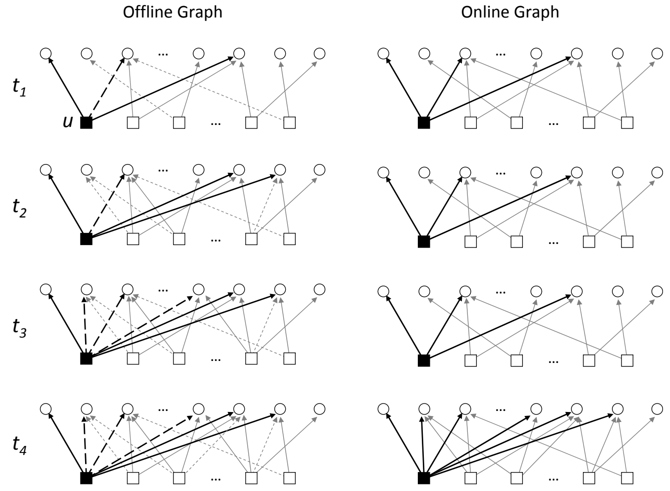


Figure 6.2: Evolution of the offline (left column) and online (right column) graphs over time ($t_1 < t_2 < t_3 < t_4$). Circles represent the items and the users are shown by the squares. We focus on the profiles of user u, distinguished by the black square. In the left column, the solid lines are the actual ratings of the users, and the dashed lines are the additional ratings aggregated to the users' offline profiles. Profiles $\mathcal{I}_u^{\text{off}}$ and $\mathcal{I}_u^{\text{on}}$ are synchronized at time t_1. Graph G_t^{on} stays unchanged, regarding user u, until the next synchronization time t_4. Graph G_t^{off} is accumulated through addition of both actual rating edges (solid lines) and also the aggregated edges (dashed lines). User u rates a new item at t_2 and aggregates some other items at t_3. The solid lines in the right column (i.e., the online graph) indicate that the server is unable to distinguish between the actual and dummy (i.e., those added through aggregation) ratings. The grey lines belong to other users' profile that are evolving independently of each other.

updates its rating, if it is not an actual rating.

We assume each user, on average, contacts n users per time unit and the contact peers are selected arbitrarily from $\mathcal{U}$. The process of contacting other users and updating the offline profiles continues for all users over time, in parallel with the process of actually rating new items. Thus, graph G_t^{off} will be accumulated. More formally, for $t_1 < t_2$, $\mathcal{I}_u^{\text{off}}(t_1) \subseteq \mathcal{I}_u^{\text{off}}(t_2)$, i.e., the resulting bipartite graph $G_{t_2}^{\text{off}}$ has the same vertex sets as $G_{t_1}^{\text{off}}$, and $E^{\text{off}}(t_1) \subseteq E^{\text{off}}(t_2)$.

Each user occasionally synchronizes his offline and online profiles at arbitrary time instants. Thus, the online graph on the server changes dynamically over time. Figure 6.2 shows this evolution.

We emphasize that each time a user synchronizes his online profile with the offline version, the server is incapable of distinguishing between the new edges that belong to the user's actual profile and the edges added through aggregation. Moreover, users who contact each other have access only to information derived from the offline profiles of their peers and cannot pinpoint the actual items of each others' offline profiles. Hence, a user's privacy is protected not only against the server but also against the other users.

In order to prevent a user being confused about his own rated items, the process of aggregation and also the aggregated items (i.e., his offline profile minus his actual profile) can be made transparent to the user. Hence, a user can add, remove, or modify his actual ratings without any confusion, although he is able to find out what the extra items are in his profile. Looking from the server's side, the server is not aware of the items that are actually rated by the user, and generates recommendations based on their online profiles. Therefore, the server passes the top items that might be of interest to a user without filtering out the items already existing in his profile. Instead, the process of filtering is done at the user's side, in a transparent way, i.e., the server recommends a set of highly recommended items to the user and it is the user application that avoids recommending the items that are previously rated by the user himself. This prevents a user from missing a recommendable item because it is in his profile but not actually rated by the user.

6.3.3 Profile Aggregation

As described in the previous section, the process of updating graph G_t^{off} is accomplished through an aggregation process where users contact each other and update their offline profiles by adding a subset of their peer's rated items to their own profile. Consider a contact between users u and v at time t. We denote $\mathcal{I}_v^{\text{agg}}(t)$ (named the *aggregated items* from v) as the subset of items (plus their ratings) in the offline profile of user v, which are added to the offline profile of user u. Considering $\mathcal{I}_u^{\text{off}}(t^+)$ as the offline profile of u after the aggregation, the following equation holds:

$$\mathcal{I}_u^{\text{off}}(t^+) = \mathcal{I}_u^{\text{off}}(t) \cup \mathcal{I}_v^{\text{agg}}(t). \tag{6.3}$$

The same argument holds for the inverse case (updated profile of v). Note that the added edges keep the same rating value after being added to a user's profile, i.e., assuming item i is added to the offline profile of user u after his contact with user v, we have:

$$r_{u,i}^{\text{off}}(t^+) = r_{v,i}^{\text{off}}(t), \quad i \in \mathcal{I}_v^{\text{agg}}(t), i \notin \mathcal{I}_u(t). \tag{6.4}$$

But, if the candidate item already exists in u's actual profile, i.e., $i \in \mathcal{I}_v^{\text{agg}}(t) \cap \mathcal{I}_u(t)$, the rate stays unchanged.

In order to choose the candidate items for aggregation, we should consider two issues: 1) The number of items to be aggregated ($|\mathcal{I}_v^{\text{agg}}(t)|$), and 2) which subset of $\mathcal{I}_v^{\text{off}}(t)$ (with cardinality $|\mathcal{I}_v^{\text{agg}}(t)|$) to choose. We provide answers to these questions by introducing different types of aggregation.

In this work, we introduce two kinds of possible aggregation processes, focusing on our key idea: aggregation based on the similarity of the users who contact each other. We believe that this approach yields the best performance of the system for preserving both privacy and accuracy.

Similarity-Based Aggregation

In this case, at each contact, the similarity of the two users is calculated using (6.1). Each user then gives a proportion - equal to the similarity value - of his items in his offline profile to the other user for aggregation, i.e., if we assume a contact between users u and v at time t, using (6.3) we have,

$$|\mathcal{I}_u^{\text{off}}(t^+)| \leq |\mathcal{I}_u^{\text{off}}(t)| + \lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor, \tag{6.5}$$

where $|\mathcal{I}_u^{\text{off}}(t^+)|$ denotes the size of u's offline profile after aggregation, and $|\mathcal{I}_u^{\text{off}}(t)|$ and $|\mathcal{I}_v^{\text{off}}(t)|$ denote the size of u and v's profiles before aggregation, respectively. Note that

$|\mathcal{I}_v^{\text{agg}}(t)| = \lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$, and the equality holds when $\mathcal{I}_u^{\text{off}}(t) \cap \mathcal{I}_v^{\text{off}}(t) = \phi$. We denote $sim_{u,v}$ as the similarity between users u and v (value between 0 and 1) is computed as follows:

$$sim_{u,v} = \begin{cases} w_{u,v} & \text{if } w_{u,v} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

In other words, the two users only accept the aggregated items when there is a positive similarity between them, otherwise no aggregation occurs.

Although our main focus is on the protection of the users' privacy against the untrusted server and not against each other, we still can employ some tools to protect users' privacy from each other. Users, who contact each other, need to reveal their profiles to each other in order to compute the similarity value. The level of privacy can be improved by using methods that compute similarity between two profiles without revealing their content. Lathia *et al.*[53] propose an interesting concordance measure to estimate the similarity between two users in a distributed system without revealing their profiles to each other. This method can be used when one user contacts another user in whom he has little trust.

To answer the question of "which subset of $\mathcal{I}_v^{\text{off}}(t)$ to choose", we notice that the items in $\mathcal{I}_v^{\text{agg}}(t)$ can be chosen through different methods. We hereby suggest two ways:

- **Similarity-based Minimum Rating Frequency (SMRF)** One way to choose the set $\mathcal{I}_v^{\text{agg}}(t)$ is to sort the elements of $\mathcal{I}_v^{\text{off}}(t)$ by their rating frequency, and select the first $\lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$ elements with minimum rating frequency. In other words, we choose the subset of the user's rated items that have been least rated by the users in the system (i.e., in the online graph). We assume that the server makes the current rating frequency of items, $f_i^{\text{on}}(t)$, available to the users.

- **Similarity-based Random Selection (SRS)** In this case, the $\lfloor sim_{u,v} \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$ candidate items are selected uniformly at random from the set of items in $\mathcal{I}_v^{\text{off}}(t)$.

We evaluate the effectiveness of these aggregation functions and compare them in Section 6.5.

Fixed Random-Selection Aggregation

Aside from choosing the aggregated items based on the similarity of users, we consider another simple type of aggregation: *fixed random selection*, where each user gives a *fixed* proportion of his rated items to the other user, i.e.,

$$|\mathcal{I}_u^{\text{off}}(t^+)| \leq |\mathcal{I}_u^{\text{off}}(t)| + \lfloor k \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor, \tag{6.7}$$

where, k is a constant value between 0 and 1. Note that here $|\mathcal{I}_v^{\text{agg}}(t)| = \lfloor k \cdot |\mathcal{I}_v^{\text{off}}(t)| \rfloor$, and the aggregated items are chosen uniformly at random.

- **Union** is an example of this aggregation, where each user gives all of his rated items to the other user, i.e.,

$$\mathcal{I}_u^{\text{off}}(t^+) = \mathcal{I}_u^{\text{off}}(t) \cup \mathcal{I}_v^{\text{off}}(t). \tag{6.8}$$

6.4 Evaluation Metrics

In this section, we describe our definition of privacy and recommendation accuracy. Following the definitions, we formalize our metrics for evaluating the proposed method in terms of the privacy gain and the accuracy loss.

6.4.1 Privacy Measurement

We define the *lack of privacy* as the amount of information the server has about the actual profile of the users. In other words, it reflects how accurately the server can guess the actual profile of the users (modeled as G_t) using the online graph G_t^{on}, and how valuable the estimated profiles are, in terms of identifying the users and distinguishing between them. To define the privacy, we focus more on the users-items connection rather than on the ratings the users assign to the items (for an adversary who tracks a user it is more interesting to know to which places the user has been, rather than knowing the user's opinion about those places).

To evaluate the privacy provided by our method, based on the above-mentioned privacy definition, we define a privacy metric considering the graphs G_t and G_t^{on}. We emphasize that privacy is preserved in our model through additional edges that exist in the online graph, but not in the actual graph. Thus, we compute to what extent the structures of graphs G_t and G_t^{on} are similar, as a higher structural difference accounts for higher privacy. This falls into the subject of approximate graph matching introduced in Chapter 1, where the structures of two graphs are compared with the goal of matching the correspondent nodes based on the structures. In our problem, as the node set of the two graphs are the same, the structural difference can be viewed as the difference between the correspondent edges.

Following the above discussion, we introduce the same error measure for edge-inconsistency as used in Chapter 3. We consider only the structures of two given graphs $G_1(V, E_1)$ and $G_2(V, E_2)$, where V denotes the node set (the same for both graphs), and E_1 and E_2 denote different edge sets. The matching error Δ is defined as the number of edges that exist in one graph with their correspondent edges not existing in the other graph:

$$\Delta = \sum_{(u,i)\in E_1} \mathbf{1}_{\{(u,i)\notin E_2\}} + \sum_{(u,i)\in E_2} \mathbf{1}_{\{(u,i)\notin E_1\}}, \tag{6.9}$$

where $\mathbf{1}_{\{A\}}$ denotes the indicator function on A, and (u, i) denotes an edge between nodes u and i in either of the two graphs.

In our setting, graphs G_t and G_t^{on} correspond to G_1 and G_2 respectively, with their node set divided into two sets V and U to exhibit the bipartite property of users and items. The privacy metric is equivalent to the matching error Δ, as Δ counts the number of differences in the corresponding edges of two graphs, and higher structural difference results in higher privacy. In our case, the first summation in (6.9) is zero, because G_t^{on} is an accumulated version of G_t. Consider our own notations, where $\mathcal{I}_u^{\text{on}}$ and $\mathcal{I}_u$ stand for the items associated with user u in his online and actual graphs, respectively. Counting the edges by iterating over the user set U, the matching error can be written as:

$$\Delta = \sum_{u} \sum_{i\in\mathcal{I}_u^{\text{on}}} \mathbf{1}_{\{i\notin\mathcal{I}_u\}}. \tag{6.10}$$

However, Narayanan and Shmatikov [77] show that an item with a high rating frequency contains less information about those who have rated the item than an item with a low rating frequency (e.g., it is more valuable, in identifying a user, to know that a user has purchased "The Color of Pomegranates" than the fact that he purchased a Harry Potter DVD). Their result shows that the higher the rating frequency of an item is, the more important this item is for identifying users associated with that item, thus the more valuable it is in system privacy preservation. If an adversary (the one who wants to break users' privacy) falsely believes that someone has rated an item with a low rating frequency, then the user's privacy is much higher than the case where the item has a high rating frequency.

Hence, we use the rating frequency f_i in (6.10) to weight the items. We then normalize it per user by dividing it by its maximum value for each user. This also guarantees a value between 0 and 1 for the system privacy. Thus, privacy in the current work can be defined as the normalized weighted edge-difference function over the node pairs of the two graphs. Rewriting (6.10), this can be expressed as follows. For the sake of simplicity, we omit the time index.

$$privacy = \frac{1}{N} \cdot \sum_u \left(\frac{\sum_{i \in (\mathcal{I}_u^{\text{on}} \setminus \mathcal{I}_u)} \left(\frac{1}{f_i}\right)}{\sum_{i \in \mathcal{I}_u^{\text{on}}} (\frac{1}{f_i})} \right) \tag{6.11}$$

To put is simply, looking at the bipartite graph, for each user we compute the weighted difference of his rated items in graphs G_t and G_t^{on}, with the weights being the inverse of the item's rating frequency. The overall *system privacy* is then computed as the normalized sum of all such terms for all users. Such a metric captures both the server's chance for successfully guessing the users' actual profiles and the importance of the profiles in identifying the users (using the rating frequency).

Let us have a deeper look at how the privacy metric can be interpreted. The closer is the privacy degree of a system to 1, the harder is for the adversary to distinguish the actual profiles of the users in their online profiles. Moreover, it becomes harder for the adversary to distinguish between users and de-anonymize their profiles [77]. As it approaches 1, the privacy growth becomes slower by adding more items to the profile of users, whereas its growth is faster for small privacy values. The privacy becomes 1 if the actual graph is infinitely small compared to the online graph and it is 0 if they are the same. It is important to note that this metric is not designed to compare the privacy of two systems with different settings, rather to reflect the privacy gain inside a system.

Note that there are some items in a user's offline profile whose rating changes due to 1) being rated later by the user himself, 2) being received again from contacting users. The first subset is not distinguishable from the second and their ratio intuitively follows (6.11). Therefore, we do not re-evaluate the effect of this factor on the users' privacy.

6.4.2 Recommendation Accuracy

Modifying the users' actual profiles in order to increase their privacy level might impose some error on the accuracy of the recommendation system. We measure the cost of running our method as the difference between the recommendation error in our system and the system with no privacy. In both cases we compute the recommendation error based on RMSE. Let $\widehat{\mathcal{I}}_u$ denote the set of predicted items in the recommender system for user u. The system error for any graph G is computed as follows (the same for graph G^{on}).

$$RMSE(G) = \sqrt{\frac{\sum_u \sum_{i \in \widehat{\mathcal{I}}_u} (r_{u,i} - \hat{r}_{u,i})^2}{\sum_u |\widehat{\mathcal{I}}_u|}} \tag{6.12}$$

Finally, the cost of using our method is computed as the percentage of the accuracy loss, i.e., the additionally imposed error, formalized as follows.

$$accuracy\ loss = \frac{RMSE(G^{\text{on}}) - RMSE(G)}{RMSE(G)} \tag{6.13}$$

6.5 Experimental Results

In this section, we validate the effectiveness of our model and the mechanisms we proposed in Section 6.3. First, we describe the data set that we used for the experiments and explain our simulation model. Next, we evaluate the effectiveness of different aggregation functions used in our mechanism based on the metrics described in Section 6.4.

In each experiment, we used a subset of the Netflix data set, released for Netflix prize [2], containing 300 randomly chosen profiles with ratings between early 2001 and late 2006. The ratings are on a scale from 1 to 5 (integral) stars.

To evaluate the recommendation accuracy, we used 10% of the actual ratings of each user as the testing set and the remaining 90% as the training set. For each experiment, we evaluated the privacy achievement and also the accuracy of the system for different values of users' contact rates. The contacts between users were selected uniformly at random, both for the contacting peers and their time of contact. We evaluated the privacy and accuracy of an aggregation function with respect to the average users' contact rate. We made the evaluation at the end of each experiment (December 2006). The contact rate value determines the average number of users a given user has contact with per *year*. Besides, in all experiments the number of neighbors of each user, needed for generating recommendations, was selected to be 30 (i.e., $|N_u| = 30$ for any user u). Outcome of various experiments are averaged to obtain the final results. In this setting, we implemented the following aggregation functions.

- Similarity-based minimum rating frequency (SMRF)
- Similarity-based random selection (SRS)
- Union aggregation

The Union function provides the maximum privacy that can be achieved using our method. Therefore, it acts as a benchmark to evaluate the effectiveness of the similarity-based aggregation functions.

Figures 6.3(a) and 6.3(b) illustrate the privacy preservation level and accuracy loss, respectively, for different aggregation functions, calculated using Equations (6.11) and (6.13). Using these results, one can easily observe the privacy-accuracy trade-off for different mechanisms, and select the appropriate type of aggregation. The effect of contact rate on the privacy level and accuracy loss is also shown.

As depicted in Figure 6.3(a), using Union aggregation the system privacy level quickly approaches the maximum value of 1 as the average contact rate increases. Moreover, when this aggregation is used, the users' profiles (offline and therefore online) become more flat as time goes on (i.e., users' profiles become more similar). This leads to a considerable decrease in system accuracy, compared to other aggregation functions (Figure 6.3(b) shows a fast increase in accuracy loss for Union function). Contrarily, Figure 6.3(b) shows that the other two methods SRS and SMRF result in a slight decline in the system accuracy, in comparison with Union aggregation, for different values of contact rate.

The efficiency of our mechanism becomes more visible when we notice that a small number of contacts per user substantially increases the privacy level and imposes minor costs in terms of accuracy. For instance, notice that using the SMRF (SRS) aggregation function, number of 6 contacts per year per user results in a 0.64 (0.48) degree increment in system privacy with an accuracy loss of less than 2% (1%).

We observe that although the SRS method causes a smaller accuracy loss, the major increase in system privacy for SMRF distinguishes it as a good alternative mechanism. This is due to the fact that sorting the aggregated candidates by minimum rating frequency makes the users more indistinguishable for the server. However, SRS is the most *practical* aggregation function, as it is not dependent on any system parameter or any kind of information from the server. Another advantage of SRS over SMRF is the stability of its accuracy loss for different contact rate values.

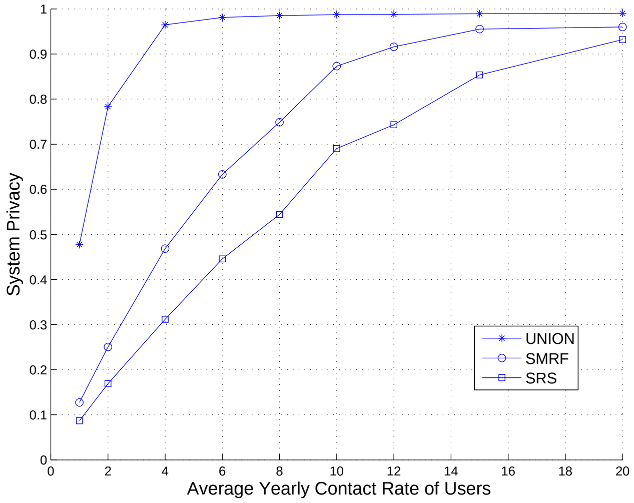
In general, experimental results validate our proposed mechanism in preserving privacy in a distributed way. The results show that *similarity-based aggregation* achieves decent results in terms of higher system privacy with a negligible effect on accuracy loss for small average contact rates. When users have information about the rating frequency of items, giving the items with minimum rating frequency increases the privacy. This is because it increases the number of users who have rated sensitive items of a user (i.e., the items that help identifying the user). Yet, if such information is not available for users, the SRS aggregation can be used, which provides a bit less privacy but imposes a lower accuracy loss. SRS is also more stable (in terms of the accuracy loss) for different contact rate values. The key factor of similarity-based aggregation functions is that they preserve the similarity between users almost unchanged, compared to the case where there is no privacy protection mechanism in use.

6.6 Related Work

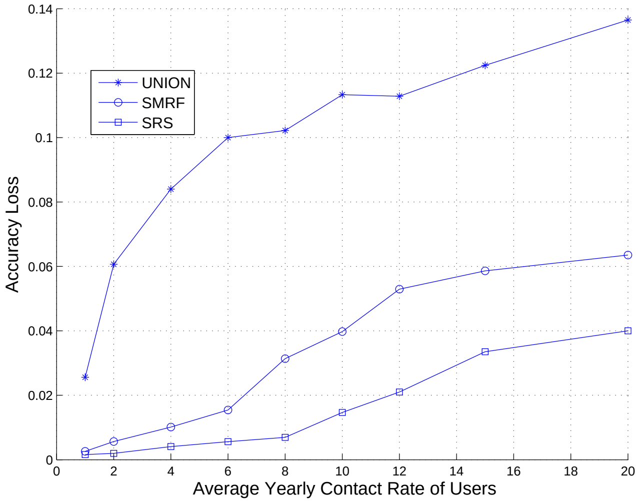
Several techniques have been proposed to preserve the privacy of users in recommender systems. Perturbing users' ratings, using cryptographic tools such as homomorphic cryptography, and storing users' profiles in a distributed manner are the main categories for privacy preservation in collaborative filtering systems.

Polat and Du [88] propose a randomized perturbation technique to preserve privacy in collaborative filtering. Users' ratings are modified by adding random noise to them in order to prevent the central server from deriving the users' actual ratings. The challenge is to find a perturbation algorithm that imposes the smallest error on the recommendation process. The users enjoy a high level of privacy if the server is not able to estimate the actual ratings they assigned to the items. However, the items rated by the users are revealed in the proposed technique, regardless of the perturbation level. This is despite the fact that keeping the connection between users and items is more crucial (in order to preserve users' privacy) than disguising the ratings assigned to those connections. Revealing the places visited by the users to the server enables it to track users over space and time, whether they liked those places or not.

Using homomorphic cryptography in the public server in order to hide the operations of the recommender system is proposed by Canny [18, 19]. In the proposed method, users create communities and each user searches for recommendations from the most appropriate community with the hope of receiving more valuable recommendations, rather than asking from those who have similar profiles. Each community of users can compute a public aggregate



(a) Privacy gain for different users' contact rate



(b) Accuracy loss for different users' contact rate

Figure 6.3: Performance of different aggregation functions with respect to the achieved privacy and drop in system accuracy.

of their profiles, which does not expose the individuals' profiles. Homomorphic cryptography allows the users to hide the aggregation operation from the server, although it is performed by the server itself. Participation of users in the distributed system to provide privacy for others was assumed to happen in this work, which might not be the case in reality. Moreover, the implementation of such a cryptographic scheme, especially its required key management, is difficult to achieve, considering the status of the current usage of cryptographic systems in the Internet. Similar ideas are proposed in [4] where homomorphic cryptography is used to hide similarity measurement from server's eyes.

Storing users' profiles on their own side and running the recommender system in a distributed manner, without relying on any server, is another option. Miller *et al.*[69] propose transmitting only the similarity measures over the network and keeping users' profiles secret on their side to preserve their privacy. Berkovsky *et al.*[13] propose a distributed P2P system to avoid storing users' profiles on a single server. Although these methods eliminate the main source of threat against users' privacy, they need high cooperation among users to generate meaningful recommendations. Every user pays the price of using this method, regardless of his interest in protecting his privacy.

Lathia *et al.*[53] propose a concordance measure to estimate the similarity between two users in a distributed system without revealing their actual profiles to each other. A randomly generated temporal profile is shared between two users, and both of them compute the number of concordant, discordant and tied pairs of ratings between their own profile and the temporal profile. Exchanging the results, they are able to estimate the similarity between their profiles. Hence, they keep the items they have rated as well as the rating values private. In this method, users need to reveal their profiles for generating recommendations. Therefore, this method provides privacy only for measuring similarity, not for a collaborative filtering system as a whole. Two users who do not trust each other, in our proposed method, can use Lathia's method to estimate their similarity.

Finally, the concept of approximate graph matching – from which the idea of privacy metric in this work was inspired – has been widely studied in literature. We refer to Section 2.2 for a more in-depth discussion of the related work in this field.

6.7 Summary

In this work, we proposed a novel method for privacy preservation in collaborative filtering recommendation systems. We addressed the problem of protecting the users' privacy in the presence of an untrusted central server, where the server has access to users' profiles. To avoid privacy violation, we proposed a mechanism where users store locally an offline profile on their own side, hidden from the server, and an online profile on the server from which the server generates the recommendations. The online profiles of different users are frequently synchronized with their offline versions in an independent and distributed way. Using a graph theoretic approach, we developed a model where each user arbitrarily contacts other users over time, and modifies his own offline profile through a process known as aggregation. To evaluate the privacy of the system, we applied our model to the Netflix prize data set to investigate the privacy-accuracy tradeoff for different aggregation types. Through experiments, we showed that such a mechanism can lead to a high level of privacy through a proper choice of aggregation functions, while having a marginal negative effect on the accuracy of the recommendation system. The results illustrate that similarity-based aggregation functions, where users receive items from other users proportional to the similarity between them, yield a considerable privacy level at a very low accuracy loss.

As future work, our mechanism can be implemented in a realistic setting in which users contact each other based on their friendship (e.g., in social networks) or their physical vicinity (e.g., using wireless peer-to-peer communication). In such a setting, various practical issues such as the effect of the privacy preserving mechanism on the overhead of users' profiles and their maintenance, and the acceptability of the system in a real world scenario can be investigated. Moreover, the robustness of the algorithm to sophisticated adversarial attacks and its relation to the proposed metric are worth studying.

Publication [97]

R. Shokri, P. Pedarsani, G. Theodorakopoulos and J.-P. Hubaux. **Preserving Privacy in Collaborative Filtering through Distributed Aggregation of Offline Profiles**. In *Proceedings of ACM International Conference on Recommender Systems*, RecSys '09, pages 157-164, 2009.

Conclusion

With the rapid growth of social networks and the increasing availability of network data, the study of such networks has gained particular attention in the recent years. In this work, we have studied two aspects of social and complex networks. First, we have looked into the privacy issue in social networks. Network owners and social network operators release extensive amounts of data to the public for research and marketing purposes. In most cases, they apply anonymization techniques in order to protect the privacy of users in the released data. We have investigated how the availability of side information threatens the identification of users in anonymized network data. Furthermore, we have introduced a privacy-preserving mechanism for social recommender systems. Second, we have investigated network evolution and the dynamics of social networks. It has been recently observed that social networks tend to become denser over time. We have proposed a novel explanation for the growth of social networks and their densification.

In **Part I**, we study network de-anonymization and how to infer user identities from anonymized network data. In **Chapter 2**, we define our approach to the de-anonymization problem. We assume the only information available to the attacker is an anonymized target network (an unlabeled graph), and an auxiliary network that is derived from some publicly available resource and is structurally similar to the anonymized graph. We introduce the de-anonymization problem as a graph matching problem, where the goal is to find the mapping between the nodes of two overlapping graphs by using only their structures. This is referred to as *approximate graph matching*. We review the literature on network de-anonymization and approximate graph matching at the end of this chapter.

In **Chapter 3**, we push further the theoretical aspect of the matching problem. We first introduce a new probabilistic model for the structural similarity of two graphs. We assume that each graph is a sample from an underlying generator graph and that a sampling probability (or similarity parameter) controls the amount of similarity between the graphs. We introduce a sampling process that samples every edge in the graph with the same probability s, independent of all other edges. Applying this process twice on the generator graph, we obtain two correlated (edge-)overlapping graphs that are structurally similar but not identical. Using this model and introducing a $G(n, p)$ random graph model for the underlying graph, we establish conditions on the network parameters and on the similarity under which an attacker with infinite computational power can perfectly match the nodes with high probability as the graph size grows large. More specifically, we derive a lower-bound for the similarity of two networks above which perfect matching is feasible, and we show that the bound is indeed a mild condition on the scaling of the average degree with the number of nodes, which is the case for many real networks. Our theoretical results imply that de-anonymization is *easy* in the presence of some minimal side information: the structural similarity to a known network.

In **Chapter 4**, we look into the algorithmic perspective of approximate graph matching:

how to match two structurally similar large graphs in an efficient way. Relying on the same probabilistic model for the similarity of two graphs, we propose a novel iterative algorithm for graph matching. At each phase of the algorithm, we consider a set of candidate nodes to be mapped; for every node in this set, we define a fingerprint including a set of node attributes, namely its degree and its distances to previously mapped nodes in the corresponding graph. Using these fingerprints and introducing models for node distances and degrees after sampling, we develop a clean Bayesian framework to compute the likelihood of two nodes being a correct match. Using all pair-wise likelihoods, we transform the matching problem to a maximum weighted bipartite matching problem at each phase. Unlike some recently proposed methods, our algorithm relies on no initial seed set and uses only the network structures to build the map incrementally. Furthermore, we show the efficiency of our algorithm to match a high percentage of nodes by applying it to real data.

Part II, including **Chapter 5**, contains our contribution for the second direction of this thesis: network dynamics. We study the evolution of social networks and a recently observed phenomenon in social networks known as *densification*: the super-linear growth of the number of edges versus nodes as the network evolves, and thus the increase of the average degree over time. Observing how network data is gathered and used in practice, we introduce an observation process, where the edges of a fixed underlying graph are sampled, and nodes are revealed only indirectly through the edges. We show that this model leads to densification, both theoretically and experimentally. We also establish conditions for densification, and show a direct relationship between a power-law degree distribution and power-law densification, which is verified well in practice. Our results imply that densification, rather than an inherent property of social and complex networks, is due to an observation bias and results from the way we observe real networks.

Finally, in **Part III** and **Chapter 6** we elaborate on our additional work on privacy in recommendation systems. We consider a collaborative filtering recommender system, and a scenario in which users intend to preserve their privacy by hiding their profiles from an untrusted recommander server while receiving accurate recommendations. We propose a model where each user modifies his/her profile before revealing it to the server. In a process called aggregation, each user contacts other random users at arbitrary points in time, and reveals an aggregated profile to the server. Our experiments on real data show that our method yields high-level privacy for users at a low accuracy loss.

Future Research Directions

The results of this thesis indicate that it is possible to break user privacy by using the simplest side information attained from an auxiliary source. Although most of our works in this direction are based on a specific similarity model, the results imply the need for designing smarter and more efficient methods of publishing user data in order to protect users' privacy. Our work can be extended in various directions.

- The theoretical approach for finding the conditions for graph matching used edge sampling to model the similarity of two networks. Currently, we are working on solving the same problem for *node sampling*, where every node is sampled with some probability, independently of all other nodes. Next, to generalize our approach, a breakthrough in this direction would be the combination of edge and node sampling, which would give the most general model for the similarity of two networks. Furthermore, the choice of a random graph as the generator can be extended to more realistic types of graphs, although the mathematical tractability could be an obstacle to generalizing the results in this regards.

- Our matching algorithm is distinguished from other methods in the literature because it uses no a priori information (no initial seeds) other than graph structures. This suggests that even though the algorithm complexity might hinder the matching of the graphs of tens of millions of nodes, it can still be used for matching a subset of nodes over a few iterations, fixing them as seeds, and then using the seeds with an existing propagation algorithm as in [78] to match all nodes.

- We applied our matching algorithm to a dataset of EPFL e-mail exchanges. The algorithm can be applied to other real datasets of interest and used not only for de-anonymization but also for other application purposes. An interesting example is matching documents from two different languages, e.g., cross-referencing dictionaries, or matching Wikipedia pages from different languages.

- Based on our results, keeping a network anonymous is difficult, knowing that adversaries can access various side information from other sources. Our method suggests that any kind of structural information can be used as node attributes and might leak some information about the node. A promising research direction could be how to perturb the structure of the graph in order to keep it private while preserving global statistics for graph properties; thus the released data could still be useful for research purposes. More fundamentally, we could study how to release network data and *guarantee* no leak in user identities. There have been several works in the database community on the notion of differential privacy for minimizing the chance of identifying records in the database [24, 25]; in graph literature, there are also some preliminary works on differential private release of network data while maintaining some structural information for public or research use [39, 70]. However, this is still an immature area with strong potential for research.

Bibliography

[1] AOL Search Data Scandal. `http://en.wikipedia.org/wiki/AOL_search_data_scandal`.

[2] Netflix prize, http://www.netflixprize.com.

[3] Rummble, http://www.rummble.com.

[4] W. Ahmad and A. Khokhar. An architecture for privacy preserving collaborative filtering on web portals. In *International Symposium on Information Assurance and Security (IAS)*, Aug. 2007.

[5] William Aiello, Fan Chung, and Linyuan Lu. A Random Graph Model for Massive Graphs. In *STOC '00*, pages 171–180, 2000.

[6] R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[7] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore Art Thou R3579X?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In *WWW '07*, pages 181–190, 2007.

[8] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 44–54, 2006.

[9] Lars Backstrom, Ravi Kumar, Cameron Marlow, Jasmine Novak, and Andrew Tomkins. Preferential behavior in online groups. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 117–128, 2008.

[10] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[11] Albert László Barabási. Scale-free networks: A decade and beyond. *Science*, 325:412, 2009.

[12] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, and C. Boeres. Inexact graph matching using learning and simulation of bayesian networks. an empirical comparison between different approaches with synthetic data. In *Workshop Notes of CaNew2000: Workshop on Bayesian and Causal Networks: From Inference to Data Mining*, 2000.

[13] Shlomo Berkovsky, Yaniv Eytani, Tsvi Kuflik, and Francesco Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *Proceedings of ACM RecSys*, 2007.

[14] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D-U. Hwang. Complex networks : Structure and dynamics. *Physics Reports*, 424(4-5):175–308, feb 2006.

[15] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D-U. Hwang. Complex Networks: Structure and Dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[16] B. Bollobas. *Random Graphs (2nd edition).* Cambridge University Press, 2001.

[17] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Microsoft Research, 1998.

[18] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, 2002.

[19] John Canny. Collaborative filtering with privacy via factor analysis. In *ACM SIGIR*, 2002.

[20] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. *IEEE Transactions on Mobile Computing*, 6:606–620, 2007.

[21] Li Chen and Ho Keung Tsoi. Privacy concern and trust in using social network sites: a comparison between french and chinese users. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part III*, INTERACT'11, pages 234–241, 2011.

[22] S. N. Dorogovtsev and J. F. F. Mendes. Accelerated growth of networks. In *Handbook of Graphs and Networks: From the Genome to the Internet, S. Bornholdt and H.G. Schuster*, Eds, Wiley-VCH, Berlin, Germany, 2002.

[23] Ran Duan and Hsin-Hao Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1413–1424, 2012.

[24] Cynthia Dwork. Differential privacy. In *in ICALP*, pages 1–12, 2006.

[25] Cynthia Dwork. Differential privacy: a survey of results. In *Proceedings of the 5th international conference on Theory and applications of models of computation*, TAMC'08, pages 1–19, 2008.

[26] Holger Ebel, Jörn Davidsen, and Stefan Bornholdt. Dynamics of social networks. *Complexity*, 2002.

[27] David Emms, Richard C. Wilson, and Edwin R. Hancock. Graph matching using the interference of discrete-time quantum walks. *Image Vision Comput.*, 27(7):934–949, 2009.

[28] Francisco Escolano, Edwin R. Hancock, and Miguel Angel Lozano. Graph matching through entropic manifold alignment. In *CVPR*, pages 2417–2424, 2011.

[29] Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989.

[30] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[31] M. Gjoka, M. Kurant, C.T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM, 2010 Proceedings IEEE*, march 2010.

[32] K.-I. Goh, Y.-H. Eom, H. Jeong, B. Kahng, and D. Kim. Structure and evolution of online social relationships: Heterogeneity in unrestricted discussions. *Physical Review*, 73, June 2006.

[33] Jennifer Golbeck. The dynamics of web-based social networks: Membership, relationships, and change. *First Monday*, 12, 2007.

[34] Andrew V. Goldberg and Robert Kennedy. An efficient cost scaling algorithm for the assignment problem. *MATH. PROGRAM*, 71:153–177, 1995.

[35] Marco Gori, Marco Maggini, and Lorenzo Sarti. Exact and Approximate Graph Matching Using Random Walks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1100–1111, 2005.

[36] Ralph Gross and Alessandro Acquisti. Information Revelation and Privacy in Online Social Networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[37] S. Guha, K. Tang, and P. Francis. NOYB: Privacy in Online Social Networks. In *WOSN'08: Workshop on Online Social Networks*, 2008.

[38] Muhammad Haseeb and Edwin R. Hancock. Feature point matching using a hermitian property matrix. In *SIMBAD*, pages 321–332, 2011.

[39] M. Hay, Chao Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 169 –178, 2009.

[40] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Chao Li. Resisting Structural Re-Identification in Anonymized Networks. *VLDB Journal*, 19(6), December 2010.

[41] Xiaoyun He, Jaideep Vaidya, Basit Shafiq, Nabil Adam, and Vijay Atluri. Preserving privacy in social networks: A structure-aware approach. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, pages 647–654, Washington, DC, USA, 2009. IEEE Computer Society.

[42] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *ACM SIGIR*, 1999.

[43] Shawndra Hill, Foster Provost, and Chris Volinsky. Network-based Marketing: Identifying Likely Adopters via Consumer Networks. *Statistical Science*, 21:256–276, 2006.

[44] S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs.* Wiley, 2000.

[45] E.M. Jin, M. Girvan, and M.E.J. Newman. Structure of growing social networks. *Phys. Rev. E*, 64(4), September 2001.

[46] Brian Karrer and M. E. J. Newman. Random Graph Models for Directed Acyclic Networks. *Physical review. E*, 2009.

[47] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: measurements, models and methods. In *Proc. of the 5th International Computing and combinatorics Conference (COCOON)*, 1999.

[48] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symposium on Theory of Computing (STOC)*, 2000.

[49] Aleksandra Korolova, Rajeev Motwani, Shubha U. Nabar, and Ying Xu. Link Privacy in Social Networks. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 289–298, 2008.

[50] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The Structure of Information Pathways in a Social Communication Network. In *KDD '08*, pages 435–443, 2008.

[51] B. Krishnamurthy and C. Willis. Characterizing Privacy in Online Social Networks. In *WOSN'08: Workshop on Online Social Networks*, 2008.

[52] Victor V. Kryssanov, Frank J. Rinaldo, Evgeny L. Kuleshov, and Hitoshi Ogawa. Modeling the dynamics of social networks. *CoRR*, 2006.

[53] Neal Lathia, Stephen Hailes, and Licia Capra. Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of ACM RecSys*, 2007.

[54] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *International Conference on Machine Learning (ICML)*, 2007.

[55] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-Tracking and the Dynamics of the News Cycle. In *KDD '09*, pages 497–506, 2009.

[56] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 462–470, 2008.

[57] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, and Christos Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2005.

[58] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker Graphs: An Approach to Modeling Networks. *J. Mach. Learn. Res.*, 11:985–1042, 2010.

[59] Jure Leskovec and Christos Faloutsos. Sampling from Large Graphs. In *KDD '06*, pages 631–636, 2006.

[60] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting Positive and Negative Links in Online Social Networks. In *WWW '10*, pages 641–650, 2010.

[61] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2005.

[62] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *KDD '05*, pages 177–187, 2005.

[63] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (ACM TKDD)*, 1(1), 2007.

[64] Jure Leskovec, Ajit Singh, and Jon Kleinberg. Patterns of influence in a recommendation network. In *Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'06, pages 380–389, 2006.

[65] L. Li, D. Alderson, J. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4), 2006.

[66] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. of the twelfth international conference on Information and knowledge management (CIKM)*, pages 556–559, 2004.

[67] Bin Luo and Edwin R. Hancock. Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10):1120–1136, 2001.

[68] David Martin and Andrew Schulman. Deanonymizing Users of the SafeWeb Anonymizing Service. In *Proceedings of the 11th USENIX Security Symposium*, pages 123–137, 2002.

[69] Bradley N. Miller, Joseph A. Konstan, and John Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3), 2004.

[70] Darakhshan J. Mir and Rebecca N. Wright. A differentially private graph estimator. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, ICDMW '09, 2009.

[71] Alan Mislove, Hema Swetha Koppula, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Growth of the Flickr Social Network. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 25–30, 2008.

[72] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, 2007.

[73] Prateek Mittal, Charalampos Papamanthou, and Dawn Song. Preserving link privacy in social network based systems. *CoRR*, 2012.

[74] Richard Myers, Richard C. Wilson, and Edwin R. Hancock. Bayesian Graph Edit Distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(6), 2000.

[75] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the feasibility of internet-scale author identification. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 300–314, 2012.

[76] Arvind Narayanan, Elaine Shi, and Benjamin I. P. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *IJCNN*, pages 1825–1834, 2011.

[77] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[78] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing Social Networks. *Security and Privacy, IEEE Symposium on*, 0:173–187, 2009.

[79] Arvind Narayanan and Vitaly Shmatikov. Myths and fallacies of "personally identifiable information". *Commun. ACM*, 53(6):24–26, June 2010.

[80] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 1):2566–2572, 2002.

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

[82] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, 2004.

[83] Gergely Palla, Albert lászló Barabási, Tamás Vicsek, and Budapest Hungary. Quantifying social group evolution. *Nature*, 446, 2007.

[84] Pedram Pedarsani, Daniel R. Figueiredo, and Matthias Grossglauser. Densification arising from sampling fixed graphs. In *SIGMETRICS '08*, pages 205–216, 2008.

[85] Pedram Pedarsani, Daniel R. Figueiredo, and Matthias Grossglauser. An iterative bayesian method for seedless graph matching. In *SIAM International Conference on Data Mining (SDM '13)*, forthcoming.

[86] Pedram Pedarsani and Matthias Grossglauser. On the privacy of anonymized networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1235–1243, 2011.

[87] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology, 2008.

[88] Huseyin Polat and Wenliang Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of IEEE ICDM*, 2003.

[89] Anand Rangarajan, James M. Coughlan, and Alan L. Yuille. A bayesian network framework for relational shape matching. In *ICCV*, pages 671–678, 2003.

[90] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 1994.

[91] Bruno Ribeiro and Don Towsley. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 390–403, 2010.

[92] Bruno F. Ribeiro, Pinghui Wang, Fabricio Murai, and Don Towsley. Sampling directed graphs with random walks. In *INFOCOM*, pages 1692–1700. IEEE, 2012.

[93] Yossi Richter, Elad Yom-Tov, and Noam Slonim. Predicting Customer Churn in Mobile Networks through Analysis of Social Groups. In *Proc. SIAM Int'l Conference on Data Mining (SDM) 2010*, 2010.

[94] John Riordan. *An Introduction to Combinatorial Analysis.* Wiley, 1958.

[95] A. Sanfeliu and K. Fu. A Distance Measure between Attributed Relational Graphs for Pattern Recognition. *IEEE Transactions On Systems, Man, and Cybernetics*, 13(3):353–362, 1983.

[96] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[97] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, and Jean-Pierre Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 157–164, 2009.

[98] Yuanyuan Tian and Jignesh M. Patel. TALE: A Tool for Approximate Large Graph Matching. *Data Engineering, International Conference on*, 0:963–972, 2008.

[99] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *J. ACM*, 23(1):31–42, 1976.

[100] Tianyi Wang, Yang Chen, Zengbin Zhang, Peng Sun, Beixing Deng, and Xing Li. Unbiased sampling in directed social graph. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, pages 401–402, 2010.

[101] Tianyi Wang, Yang Chen, Zengbin Zhang, Tianyin Xu, Long Jin, Pan Hui, Beixing Deng, and Xing Li. Understanding graph sampling algorithms for social network analysis. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops*, ICDCSW '11, pages 123–128, 2011.

[102] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[103] Mark L. Williams, Richard C. Wilson, and Edwin R. Hancock. Multiple graph matching with bayesian inference. *Pattern Recognition Letters*, 18(11-13):1275–128, 1997.

[104] Richard C. Wilson and Edwin R. Hancock. A bayesian compatibility model for graph matching. *Pattern Recognition Letters*, 17(3):263–276, 1996.

[105] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A Practical Attack to De-Anonymize Social Network Users. In *IEEE Symposium on Security & Privacy*, 2010.

[106] Bai Xiao, Edwin R. Hancock, and Richard C. Wilson. A generative model for graph matching and embedding. *Computer Vision and Image Understanding*, 113(7):777–789, 2009.

[107] Bin Zhou and Jian Pei. Preserving Privacy in Social Networks Against Neighborhood Attacks. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 506–515, 2008.

Index

anonymization, 2
approximate graph matching, 3, 12, 13, 30, 33, 91
Autonomous Systems dataset, 75
auxiliary information, *see* side information

Bayes' rule, 36
Bayesian inference, 34
binomial random variable, 20, 40
bipartite graph, 42, 87

Chernoff bound, 21
collaborative filtering, 83
  centralized, 84
  distributed, 84
  memory-based, 85
    item-based, 85
    user-based, 85
  model-based, 85
complex network, 4
complex networks, 1, 53

de-anonymization, 2, 12, 15
densification, 4, 53
  cutoff, 59
  regimes, 59
differential privacy, 101

edge overlap, 27, 45, 46
edge sampling, 5, 16, 35, 39, 54, 56
  independent edge sampling, 28
  sampling parameter, 18, 47, 56, 64, 66
EPFL e-mail network, 27, 45, 46, 55, 72

First Moment Method, 19
forest fire model, 4, 53

generator graph, *see* underlying graph
graph isomorphism, 3, 33
  automorphism, 16, 23
graph matching, 11, 33
graph sampling, 4, 35

Hungarian algorithm, 41, 43

law of large numbers, 50

matching algorithm, 34, 41
  anchor, 35, 38, 41
  Bayesian framework, 35
  complexity, 43
  confidence metric, 34
  cost function, 34, 36
  iterative, 35, 41
  matching likelihood, 36, 45
  node degree, 37, 41
  normalization of likelihoods, 44
  pair-wise distance, 37, 41
  prior distributions, 47
  statistical framework, 34
matching error, 45, 46
maximum weighted bipartite matching, 42, 43

Netflix prize dataset, 93
network evolution, 3, 5, 54
node attributes, 36
  absolute attributes, 36
  relative attributes, 36
node mismatch, 23, 38

observation models
  accumulation model, 54, 69, 70
  exact structure, 71
  modulation model, 54, 69, 71
  rate distribution, 71

Pearson's Chi-Square test, 28
Pearson's correlation coefficient, 85
permutation, 17
  identity permutation, 18, 20, 23
  non-identity permutation, 19
  order-k permutation, 18

personally identifiable information, 2, 11
power-law degree distribution, 54, 61
    degree exponent, 62, 64
power-law densification, 53, 61
    densification exponent, 61, 62, 65
preserving privacy in collaborative filtering
    offline graph, 87
    online graph, 87, 91
    privacy measure, 91
    profile aggregation, 86, 87, 89
        similarity-based minimum rating frequency, 90, 93
        similarity-based random selection, 90, 93
        union aggregation, 90, 93
    profile synchronization, 88
    users' contact, 86, 87
privacy-preserving, 2, 83

random graph, 16
    $G(n, p; s)$ model, 16
    Erdös-Rényi random graph, 16
re-identification, *see* de-anonymization
recommendation accuracy, 92, 93
recommendation systems, 83

seed set, 2, 33, 101
side information, 2, 15, 101
similarity parameter, 16, 24
social networks, 1
    dynamics, 3, 53
    privacy, 1
sparse graph, 30
sparse graphs, 34
structural similarity, 5, 16, 26, 33
    cost function, 17, 91
    edge-consistency, 17, 91

underlying graph, 5, 16, 53
    fixed, 5, 53
untrusted server, 86
user identities, 2, 11
user privacy, 1, 11, 83
    anonymity, 2
user profiles, 83
    offline, 84, 86
    online, 84, 86
user ratings, 85, 87
users' similarity, 85

Zipf distribution, 63, 64

Pedram Pedarsani

Work Address
EPFL IC ISC LCA4
BC 203 (Bâtiment BC)
Station 14
CH-1015 Lausanne
Switzerland

Phone
work +41 21 693 12 61
home +41 21 691 03 74
mobile +41 77 410 05 31
E-mail
pedram.pedarsani@epfl.ch
Website
http://people.epfl.ch/pedram.pedarsani

Nationality: Iranian
Birth Date: Dec. 04, 1982
Place of Birth: Shemiran, Iran
Marital Status: Married

Education

[Apr. 2007 – present] *PhD Student and Research Assistant,* Laboratory for computer Communications and Applications (LCA), School of Computer and Communication Sciences (IC), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. **PhD Advisor**: Prof. Matthias Grossglauser

[Oct. 2005 – Feb. 2007] *M.Sc. in Communication Systems*, with **"Specialization in Wireless Communications",** EPFL, School of Computer and Communication Sciences (IC), Lausanne, Switzerland. Overall GPA: **5.32/6**

[Sep. 2001 – Jul. 2005] *B.S. in Electrical Engineering - Telecommunications*, University Of Tehran, Department of Electrical and Computer Engineering, Faculty Of Engineering, Tehran, Iran. Overall GPA: **18.50/20.00**

Research and Professional Experience

Research Assistant (PhD Thesis)

Working as research assistant at Laboratory for computer Communications and Applications (LCA), Doctoral program in computer, communication and information sciences (EDIC), EPFL, 2007-2013.

Thesis title: ***"Privacy and Dynamics of Social Networks"***.

Research skills and tasks:

- Theory and modeling of complex and social networks
- Data mining and analysis of large graphs and datasets
- Proposing probabilistic models for network properties and dynamics
- Investigating the privacy issues and the threat of user identification in anonymous network data

PhD Advisor: Prof. Matthias Grossglauser

Visitor

NOKIA Research Center (NRC), Sep-Nov 2008 and Nov-Dec 2009, Helsinki, Finland.

Research tasks: Research on the same topics as of PhD plan, focusing on *Matching Large Graphs.*

Supervisor: Prof. Matthias Grossglauser

Master Thesis

Thesis title: **"*Social Networks: Modeling and Applications*"**, Fall 2006, EPFL (LCA).

Research tasks: Studying models of complex networks, focusing on social networks, developing probabilistic models for observed phenomena in social networks through investigating large datasets of e-mail exchanges

Advisor: Prof. Matthias Grossglauser
Grade: 5.5/6

Semester Project

Project title: **“*Distributed Detection in Wireless Ad Hoc Sensor Networks*”**, Spring 2006, EPFL (LICOS).

Research tasks: Studying a known scheme for distributed detection using ad hoc sensors, and optimizing its performance

Advisor: Prof. Suhas Diggavi and Prof. Matthias Grossglauser
Grade: 5.5/6

Bachelor Thesis

Thesis title: “*Reducing PAR (Peak to Average Ratio) in OFDM*”, Fall 2004 / Spring 2005, University of Tehran.

Research tasks: Investigating several methods for the problem and implementation using MATLAB

Advisor: Prof. Said Nader Esfehani
Grade: 20/20

Publications

An Iterative Bayesian Method for Seedless Graph Matching, Pedram Pedarsani, Daniel Figueiredo, Matthias Grossglauser, Submitted to 2013 SIAM International Conference on Data Mining (SDM 2013), Austin, TX, USA.

On the Privacy of Anonymized Networks, P. Pedarsani and M. Grossglauser, 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-2011), San Diego, CA, USA, 2011.

Preserving Privacy in Collaborative Filtering through Distributed Aggregation of Offline Profiles, Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, and Jean-Pierre Hubaux, Proceedings of the 2009 ACM Conference on Recommender Systems, New York City, USA, 2009.

Densification Arising from Sampling Fixed Graphs, Pedram Pedarsani, Daniel Figueiredo, Matthias Grossglauser, Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Annapolis MD, USA, 2008.

Teaching Experience

Teaching Assistantship

Spring 2010	“Informatique II”, EPFL, STI Department
Spring 2009	“Programmation”, EPFL, ENAC Department
Fall 2007	“Computer Networks”, EPFL, IC Department
Fall 2006	“Signal Processing for Communication”, EPFL, IC Department
Spring 2005	“Engineering Mathematics”, University of Tehran, ECE
Fall 2004	“Engineering Mathematics”, University of Tehran, ECE
Fall 2004	“Linear Control Systems”, University of Tehran, ECE
Spring 2004	“Engineering Mathematics”, University of Tehran, ECE

Honors and Awards

2005-2007, ***Departmental Fellowship*** at EPFL (in terms of financial aid covering all educational and living expenses).

2003-2005*,* ***Ranked 1st*** in Communications field, Electrical Engineering, University of Tehran.

2004, ***Ranked 7th*** among 12000 participants in the nationwide university entrance exam for graduate studies in electrical engineering, Tehran, Iran.

2002-2003 & ***2001-2002*****,** ***FOE*** (Faculty Of Engineering) Award for ***ranking 3rd and 2nd*** among all Electrical Engineering students at University of Tehran.

Computer Skills

Platforms: Windows, Linux/Unix, DOS

Programming Skills: Matlab, Java, Python, C, C++, SQL

Other: MS Office (Word, Excel, PowerPoint, FrontPage, Visio), LaTeX

Languages

English: Professional (Reading, Writing, Listening, Speaking)

- **TOEFL**: 637/677 in paperbased exam, August 2004
- **GRE**: *Verbal*: 440, *Quantitative*: 800, *Analytical Writing*: 4.5, Oct. 2004
- **FCE (First Certificate in English)**, from Cambridge University, with Grade **'A'**, Dec. 1999
- Working for 4 years in an English-speaking work environment

French: Intermediate/Advanced (Reading, Writing, Listening, Speaking)

- Level B2 Certificate from EPFL Language Center
- Living in French-speaking part of Switzerland (Lausanne) for 4 years

German: Basic, Level A1 (Reading, Writing, Speaking)

Persian: Mother Tongue

Relevant Graduate Courses (EPFL)

- Microeconomics
- Dynamical Networks
- Pattern Classification and Machine Learning
- Stochastic Models in Communications and Computer Science
- Advanced Analysis of Algorithms
- Models and Methods for Random Networks
- Advanced Digital Communications
- Information Theory
- Algebra for Digital Communications
- Mobile Networks
- Mobile Satellite Communications
- Advanced Digital Image Processing
- Wireless Communications and Mobility
- Advanced Signal Processing: Wavelets and Applications

Extracurricular Activities and Hobbies

- Playing Piano for 8 years, & Santoor (an Iranian traditional musical instrument)for 9 years, and participating in various competitions and concerts
- Playing Tennis, swimming, hiking, running
- Social activities and organizing events, e.g., being elected as the President of Iranian Students Association at EPFL (IRSA) in 2007-2008