UNIVERSITY OF CALIFORNIA,
IRVINE

Measurement of Online Social Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Networked Systems

by

Minas Gjoka

Dissertation Committee:
Athina Markopoulou, Chair
Carter T. Butts
Scott Jordan

2010

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

It is a pleasure to thank those who made this thesis possible.

First of all, I owe my deepest gratitude to my advisor, Athina Markopoulou. Her support and guidance in the last four years has been invaluable and she has contributed greatly to my intellectual maturity and the achievement of my academic goals. The directions I followed in this thesis would not be possible without her help.

I would like to thank Carter T. Butts and Scott Jordan for their advice as members of my thesis committee. I am especially grateful to Carter for all the input in this thesis the last two years and for suggesting the idea of multigraph sampling. His feedback, starting from the advancement exam, has shaped the research directions I followed in the area of online social networks.

I would like to thank my collaborators on all the research projects I have been involved during my studies: Pegah Sattari, Xiaoyuan Yang, Karim El Defrawy, Vinayak Ram, Parminder Chhabra, Christina Fragouli, KC Claffy. I am especially grateful to co-authors Maciej Kurant and Michael Sirivianos. I am grateful to Pablo Rodriguez for hiring me as an intern at Telefonica Labs and giving me the opportunity to work in a challenging project. I would like to thank Xiaowei Yang, Balachander Krishnamurthy and Oliver Spatscheck for their advice during my first year as a graduate student at UCIrvine.

This work would not have been possible without the support of my family and friends. I would like to thank my parents, my sister and Mariya for their practical, moral and emotional support. I would like to thank all the friends I made at UC Irvine for the fun distractions they provided throughout all these years, which allowed me to recharge my batteries and focus better on my academic goals. First and foremost, Michael Sirivianos, Karim El Defrawy, and Vinayak Ram, all co-authors and good friends, with whom I shared important moments and had extensive discussions on all matters imaginable. To Tim, Andrea, Younsun, Fragkiskos, Vicky, Fabio, Dhonam, Ersin, Hande, Rex, Alpha, Peyman, Blerim, thank you for the time we spent together. Many thanks to my lab mates Fabio, Pegah, Anh, Hulya, Abinesh, Giuseppe.

Finally, I would like to thank the Networked Systems program and the Center for Pervasive Communications and Computing at UC Irvine for assisting me with fellowships during my studies.

CURRICULUM VITAE

Minas Gjoka

EDUCATION

Doctor of Philosophy in Networked Systems **2010**
University of California, Irvine *Irvine, California*

Master of Science in Networked Systems **2008**
University of California, Irvine *Irvine, California*

Bachelor of Science in Computer Science **2005**
Athens University of Economics and Business *Athens, Greece*

ABSTRACT OF THE DISSERTATION

Measurement of Online Social Networks

By

Minas Gjoka

Doctor of Philosophy in Networked Systems

University of California, Irvine, 2010

Athina Markopoulou, Chair

In recent years, the popularity of online social networks (OSN) has risen to unprecedented levels, with the most popular ones having hundreds of millions of users.

This success has generated interest within the networking community and has given rise to a number of measurement and characterization studies, which provide a first step towards their understanding. The large size and access limitations of most online social networks make it difficult to obtain a full view of users and their relations. Sampling methods are thus essential for practical estimation of OSN properties.

Key to OSN sampling schemes is the fact that users are, by definition, connected to one another via some relation. Therefore, samples of OSN users can be obtained by exploring the OSN social graph or graphs induced by other relations between users. While sampling can, in principle, allow precise inference from a relatively small number of observations, this depends critically on the ability to collect a sample with known statistical properties. An early family of measurement studies followed Breadth-First-Search (BFS) type approaches, where all nodes of a graph reachable from an initial seed were explored exhaustively. In this thesis, we follow a more principled approach: we perform random walks on the social graph to collect uniform samples of OSN users, which are representative and appropriate for further statistical analysis.

First, we provide an instructive comparison of different graph exploration techniques and apply a number of known but perhaps underutilized methods to this problem. We show that previously used BFS-type methods can produce biased samples with poor statistical properties when the full graph is not covered, while randoms walks perform remarkably well. We also demonstrate how to measure online convergence for random walk-based approaches. Second, we propose multigraph sampling, a novel technique that performs a random walk on a combination of OSN user relations. Performed properly, multigraph sampling can improve mixing time and yield an asymptotic probability sample of a target population even where no single connected relation on the same population is available. Third, we apply the presented methods to collect some of the first known unbiased samples of large scale OSNs. An important part of this collection is the development of efficient crawlers that address the related technical challenges. Using the collected datasets we present characterization studies of `Facebook` and `Last.fm`. Finally we present the first study to characterize the statistical properties of OSN applications and propose a method to model the application installation process.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Overview of Online Social Networks

The term "social network" is defined as a structure of social entities connected to other social entities through various types of relations. We refer to these entities as "users", for our specific purposes. The properties of off-line [1] social networks have been extensively studied by sociologists in the past. A well-known example is the experiment conducted by Travers and Milgram [1], in which randomly selected individuals in Boston and Omaha were asked to direct letters to a target person in Boston, each forwarding his or her letter to a single acquaintance whom they judged to be closer than themselves to the target. Subsequent recipients did the same. The average length of the resulting acquaintance chains for the letters that eventually reached the target was approximately six, revealing not only that short paths exist between individuals (small world phenomenon) in a large social network

[1] The term offline here means non-computer mediated communication

but that ordinary people can find these short paths (navigability). Typically, studies of offline social networks were performed on a relatively small set of participants, due to the inherent difficulty of doing larger scientific studies in an offline setting with real people.

The commercialization of the Internet in the 90s laid the foundations for new disruptive and innovative technologies. Recently, web-based Online Social Networks (OSNS), such as `Facebook` and `Myspace`, have quickly emerged as a new Internet killer-application. We can view OSNs as natural extensions of Internet applications that establish relationships between users, such as email and instant messaging. However, unlike those applications, OSNs not only facilitate direct communication between users but also allow the users to post content that revolves around their profiles creating online personae that typically map to their real life personalities. In addition, OSNs explicitly expose a user's social contacts, enabling users to browse each other's social networks in search of common friends and interesting content.

For a brief history of online social networks, starting from their early years to becoming mainstream, we refer the interested reader to surveys such as [2, 3]. [4] contains a list of well-known social networking websites with information including the content and focus, date launched and number of registered users. [5] discusses the reasons behind the widespread usage of online social networks and attempts to present a taxonomy of OSNs based on their use, purpose, and location.

In recent years, the popularity of online social networks (OSNs) has skyrocketed. The adoption of online social networking websites by Internet users is off the charts with respect to almost every single metric. In November 2010, `Facebook`, the most popular OSN, counted more than 500 million members and the total combined membership in the top five OSNs (`Facebook`, `QQ`, `Myspace`, `Orkut`, `Twitter`) exceeded 1 billion users. Putting this number into context, the population of OSN users is approaching 20% of the world population and it is more than 50% of the world's Internet users. According to Nielsen

[6] the world now spends over 110 billion minutes on social media sites per month which accounts for 22% of all time online, surpassing even email messaging as the most preferred online activity. According to Alexa [7], a well-known traffic analytics website, `Facebook` is the second most visited website in the Internet (only behind Google) with each user spending 30 minutes on average per day on site (more than Google). Four of the top five OSNs are also contained in Alexa's top 15 websites in regard to traffic rankings. Clearly, OSNs in general and `Facebook` in particular have become an important phenomenon on the Internet, which is worth studying.

1.1.2 Why study Online Social Networks?

The aforementioned statistics show that OSNs already shape the Internet traffic and more importantly, the share of traffic generated from within OSNs, is bound to increase. Therefore an obvious reason to study OSNs is to optimize content delivery to users. From the point of view of OSN operators that could mean optimizing data storage in the cloud to reduce response times. For example, [8] design a social partitioning and replication middleware that transparently leverages the social graph structure to achieve data locality while minimizing replication. Network operators could exploit OSNs to optimize content delivery by predicting user demand and prefetching and caching content as proposed in [9]. [10] performed a measurement study of the network-level effects of popular third party applications on `Facebook`.

The importance of OSNs transcends the OSN boundaries and has the potential to drastically impact almost every website. The premise here is that third party Internet services can leverage the social graph to alter user behavior. Mislove *et al.* examined the ability to use social links to improve the Internet search results [11]. They exploit the information contained in hyperlinks and information from implicit and explicit user feedback. They

also rank search results with respect to the interests of a social network to the current Web search.

Another use of OSNs is to create algorithms that can exploit trusted or influential users. High-degree nodes in the core are critical for the connectivity and the flow of information in these networks. In [12], Mislove *et al.* designed and evaluated Ostra, a system that explores the use of social trust, to thwart unwanted communication while not impeding legitimate communication. Sirivianos *et al.* utilize social trust in [13] to build a collaborative spam filtering system that is Sybil-resilient and in [14] to build a system that enables online personas to cost-effectively obtain credentials that verify the credibility of their identity statements.

Many social networks have the useful property of being searchable within a few hops. Users can direct messages through their network of acquaintances to reach a specific but distant target person within a few hops. [15] offers an explanation of social networks searchability based on social characteristics. Their model defines a class of searchable networks and a method for searching them, that is applicable to many network problems, such as location of data files in P2P networks, pages in WWW, and user reputation.

Most importantly, the advent of Online Social Networks has given researchers multiple sources of large scale social graph data in an unprecedented scale and richness. OSN data would be of interest to multiple disciplines, *i.e.,* to design viral marketing strategies, to model the spread of influence through social networks, conduct low cost surveys in large scale, detect hidden community structures etc.

1.1.3 Why sample Online Social Networks?

The immense interest generated by OSNs has given rise to a number of measurement and characterization studies, which provide a first step towards their understanding. Only a very small number of these studies are based on complete datasets provided by the OSN operators [16, 17]. A few other studies have collected a complete view of specific parts within an OSN, *i.e.,* [18] collected a complete sample of the Harvard university network. However, in the majority of cases, the complete dataset is typically not available to researchers, as most OSNs are unwilling to share their company's data even in an anonymized form, primarily due to privacy concerns.

Furthermore the large size and access limitations of most OSN services (*e.g.,* login requirements, limited view, API query limits) make it difficult or nearly impossible to fully crawl the social graph of an OSN. As a practical example, let us estimate the effort needed to crawl `Facebook`'s social graph. In November 2010, `Facebook` advertised more than 500 million active users, each encoded by 64 bits (4 bytes) long userID, and 130 friends per user on average. Therefore, the raw topological data alone, without any node attributes, amounts to at least $500M \times 130 \times 8bytes \simeq 520GBytes$. In many cases, HTML scraping is necessary, which increases the overhead multifold. For the same example, if we conservatively assume that each user occupies 500 bytes in the HTML page that contains a user's friend list, one would have to download about $260TBytes$ of HTML data.

In practice, a relatively small but representative sample may be a sufficient input for studies of social graph properties and user behavior or for algorithms that leverage the social graph to improve systems design. Sampling methods are thus essential for estimation of OSN properties in practice. While sampling can, in principle, allow precise inference from a relatively small number of observations, this depends critically on the ability to draw a sample with known statistical properties. The lack of a sampling frame (*i.e.,* a complete list

of users, from which individuals can be directly sampled) for most OSNs makes principled sampling especially difficult. Our work has thus focused on methods of sampling that evade this limitation.

Key to OSN sampling schemes is the fact that users are, by definition, connected to one another via some relation. Therefore, samples of OSN users can be obtained by exploring the OSN social graph.

1.2 Contributions

The goal in this thesis is to measure properties and understand processes that develop in an OSN, without having access to the full data. Towards this purpose, we perform network sampling with statistically principled methods to infer properties of the underlying graph. The contributions can be categorized in the following four parts.

1.2.1 Social graph sampling

In chapter 4, we consider and implement several OSN crawling techniques and apply them in `Facebook`, the largest social graph at the time. Our contributions lie both in the measurement methodology and in the data collection effort. With regards to the methodology, our recommended measurement techniques apply and combine known results from random walk sampling specifically in the OSN context. Additionally, we introduce the use of online formal convergence diagnostics to assess sample quality during the data collection process.

With respect to the collected sample, this is the first representative sample of OSN users, collected by state-of-the-art methodologies and made publicly available to the research

community [19]. Finally, we perform a characterization of several key properties of Facebook's social graph, and find that some of them are substantially different from what was previously believed.

1.2.2 Multigraph sampling

In chapter 5, we introduce *multigraph sampling*, a novel methodology that utilizes multiple relations to explore the social graph for the purpose of collecting a representative sample of OSN users. Multigraph sampling can obtain a representative sample even when the individual relations fail, *i.e.,* are disconnected or highly clustered. To the best of our knowledge, our work is the first to explore sampling OSNs on a combination of multiple relations.

Although the idea of utilizing multiple relations to explore the social graph is conceptually simple and intuitively beneficial, it may be computationally difficult in practice if naively implemented. An additional contribution is the design of an efficient algorithm that walks on different graphs as the chain evolves, mixing across possible relations so as to avoid their individual limitations.

Finally, we demonstrate the benefits of our approach by performing measurements of Last.fm. We provide characterization of Last.fm user properties and present an application of our sampling, inferring Last.fm weekly charts.

1.2.3 Crawlers

An integral part of social graph and multigraph sampling was the development of efficient crawlers, that operate in a distributed manner and are customized for Facebook and Last.fm. The design and implementation of these crawlers is a step towards a general-purpose OSN crawler and contribution of this thesis. In chapter 3 we describe the chal-

lenges we faced and key mechanisms that we used to address them.

1.2.4 Facebook applications

In chapter 6, we present the first study to characterize the popularity and user reach of `Facebook` applications. We crawl publicly accessible `Facebook` user profiles and obtain per-user application installation statistics. We make this dataset publicly available to the research community [19]. Finally, we propose a simple and intuitive method to simulate the process with which users install applications. Using this method one can determine the user coverage from the popularity of applications, without detailed knowledge of how applications are distributed among users.

1.3 Thesis Outline

The structure of the rest of this thesis is as follows. Chapter 2 discusses related work. Chapter 3 describes challenges we faced in collecting data from OSNs. Chapter 4 presents a case study of unbiased sampling of OSN users, applied in `Facebook`. Chapter 5 presents multigraph sampling and an evaluation of its benefits on synthetic graphs and Last.fm. Chapter 6 characterizes the popularity and user reach of `Facebook` applications and presents a model of OSN application installations. Chapter 7 concludes the thesis.

Chapter 2

Related Work

Broadly speaking, there are two types of work most closely related to this thesis: (i) sampling *techniques*, focusing on the quality of the sampling technique itself and (ii) *characterization studies*, focusing on the properties of online social networks based on a collected sample. These two categories are not necessarily disjoint.

2.1 Graph sampling techniques

Sampling techniques can be roughly classified into two categories (a) graph traversal techniques and (b) random walks. In *graph traversal techniques*, each node in the connected component is visited exactly once, if we let the process run until completion. These methods vary in the order in which they visit the nodes; examples include Breadth-Search-First (BFS), Depth-First Search (DFS), Forest Fire (FF) and Snowball Sampling (SBS) [20]. BFS, in particular, is a basic technique that has been used extensively for sampling OSNs in past research [16, 21, 22, 23, 24, 25]. One reason for this popularity is that an (even incomplete) BFS sample collects a full view (all nodes and edges) of some particular region

in the graph. However, BFS has been shown to lead to a bias [26, 27] towards high degree nodes in various artificial and real world topologies. Our work also confirms the bias of BFS when sampling Online Social Networks. Furthermore, this bias has not been analyzed so far for general graphs.[1]. In general cases, in order to reduce the bias, effort is usually put on completing the BFS, *i.e.,* on collecting all or most of the nodes in the graph. It is also worth noting that BFS and its variants lead to samples that not only are biased but also do not have provable statistical properties.

Ye *et al.* [30] perform crawling simulations on previously collected samples of online social graphs and examine the efficiency, sensitivity and bias of four graph traversal methodologies. Their findings agree with our results in regard to the bias of BFS. In comparison to [30], we perform online crawling of large scale online social networks and we thoroughly address the challenge of crawling bias by using principled methods.

Random walks are well-studied - see [31] for an excellent survey. They have been used for sampling the World Wide Web (WWW) [32, 33], peer-to-peer networks [34, 35, 36], and other large graphs [37]. In [32], a random walk with jumps is used to achieve near-uniform sampling of URLs in the WWW. Their setting is different since the URL graph is directed and random jumps are needed to avoid entrapment in a region of the web. Gkantsidis *et al.* [36] simulate various peer-to-peer topologies and show that random walks outperform flooding (BFS) with regards to searching for two cases of practical interest. They also argue that random walks simulate uniform sampling well with a comparable number of samples. Leskovec *et al.* [37] explore several sampling methods and compare them in terms of various graph metrics; their evaluations in static and dynamic graphs show that random walks perform the best.

Random walks are also biased but their bias can be analyzed using classic results from

[1]Some efforts to correct bias for random graphs with a given degree distribution include the works by Achlioptas *et al.* [28] and by Kurant *et al.* [29].

Markov Chains and corrected either at the end or during the data collection. In the context of peer-to-peer sampling, Rasti *et al.* [35] have applied re-weighting at the end to yield unbiased estimators of peer properties. A re-weighted random walk is considered as a special case of Respondent-Driven Sampling (RDS) [38], if sampling is with replacement and exactly one neighbor is selected in every step [39]. Alternatively, the random walk can be modified during the collection using the Metropolis rules so as to achieve, by design, *any* desired stationary distribution [40, 41]. In our case, we would like this target distribution to be the uniform. This algorithm, known as Metropolis-Hasting Random Walk (MHRW) has been applied to peer-to-peer networks by Stutzbach *et al.* in [34]: they use a Metropolized Random Walk with Backtracking (MRWB), modified to deal with peer churn, to select a representative sample of peers in a peer-to-peer network and demonstrate its effectiveness, through simulations over artificially generated graphs as well as with measurements of the Gnutella network. [35, 42] contains a comparison of Re-Weighted Random Walk (or RDS in their terminology) with Metropolis-Hasting Random Walk.

Compared to the aforementioned sampling techniques, our work is mostly related to the random walk techniques. In Chapter 4 we obtain unbiased estimators of user properties of `Facebook` using MHRW and RWRW; BFS and RW (without re-weighting) are used mainly as baselines for comparison. We accompany the basic crawling techniques with formal, *online convergence diagnostic tests* using several node properties, which, to the best of our knowledge, has not been done before in measurements of such systems. We also implement *multiple parallel chains*, which have also been recently used in [35] but started at the same node (while we start from different nodes, thus better utilizing the multiple chains). We demonstrated that random walks, whose bias can be analyzed and corrected, are able to estimate properties of users in online social networks remarkably well in practice. We also observed that correcting for the bias at the end, rather than during the walk, appears to be more efficient due to faster mixing in the Markov chain - a finding that agrees with [35].

In terms of application, we apply the measurement techniques to *online social networks*, instead of peer-to-peer and other complex networks, and we study characteristics specific to that context. We are the first to perform unbiased sampling of large scale OSNs. Krishnamurthy *et al.* [43] ran a single Metropolis Random Walk, inspired by [34], on Twitter as a way to verify the lack of bias in their main crawl used throughout their paper. However, the Metropolis algorithm was not the main focus of their paper and Twitter is a directed graph which requires different treatment. In parallel to our work, Rasti *et al.* [42] also applied similar random walk techniques to collect unbiased samples of Friendster.

Design of random walk techniques to improve mixing include [44, 45, 46, 47]. Boyd *et al.* [44] pose the problem of finding the fastest mixing Markov Chain on a known graph as an optimization problem. However, in our case such an exact optimization is not possible since we are exploring an unknown graph. Ribeiro *et al.* [45] introduce Frontier sampling and explore multiple dependent random walks to improve sampling in disconnected or loosely connected subgraphs. Multigraph sampling aims towards the same goals but instead achieves it by exploring the social graph using multiple relations. We believe Frontier sampling can be combined with multigraph sampling for additional benefits. Multigraph sampling is also remotely related to techniques in the MCMC literature (*e.g.,* Metropolis-coupled MCMC or simulated tempering [46]) that seek to improve Markov chain convergence by mixing states across multiple chains with distinct stationary distributions. In [47, 48] Thompson *et al.* introduce a family of adaptive cluster sampling (ACS) schemes, which are designed to explore nodes that satisfy some condition of interest. Whereas in our work, we are interested in characterizing the properties of the whole population.

Previous work on the evolution and dynamic growth of social networks includes [49, 50, 51]. Kumar *et al.* [49] studied the structure and evolution of Flickr and Yahoo! 360, from datasets provided by their corresponding operators. Backstrom *et al.* [50] presented different ways in which communities in social networks grow over time and [51] proposed

a method for modeling relationships that change over time in a social network. In our work, we consider that the network of interest remains static during the duration of the crawl. This turns out to be a good approximation, as we justify for each data collection case. Therefore, we do not consider dynamics, which are essential in other sampling contexts [52, 42, 53].

A unique asset of our study is the collection of true uniform samples of OSN users through rejection sampling of userIDs (UNI), which served as *ground truth*. It is important to note that UNI yields a uniform sample of users regardless of the allocation policy of userIDs by the OSN. UNI is star random node sampling scheme and yields a different type of sample from induced subgraph random node sampling schemes presented in [54, 37].

2.2 Characterization studies of OSNs

In terms of studies that measure and characterize online social networks there have been several papers. In [55], Krishnamurthy presents a summary of the challenges that researchers face in collecting data from OSNs. Ahn et. al. in [16] analyze three online social networks; one complete social graph of `Cyworld` obtained from the `Cyworld` provider, and two small samples from `Orkut` and `Myspace` crawled with BFS. Interestingly, in our MHRW sample we observe a multi-scaling behavior in the degree distribution, similarly with the complete `Cyworld` dataset. In contrast, the crawled datasets from `Orkut` and `Myspace` in the same paper were reported to have simple scaling behavior. We believe that the discrepancy is due to the bias of the BFS-sampling they used. In [21, 22], Mislove *et al.* studied the properties of the social graph in four popular OSNs: `Flickr`, `LiveJournal`, `Orkut`, and `YouTube`. Their approach was to collect the large Weakly Connected Component, also using BFS; their study concludes that OSNs are structurally different from other complex networks.

[24, 18, 56] are closely related to our study as they also study Facebook. Wilson *et al.* [24] collect and analyze social graphs and user interaction graphs in Facebook between March and May 2008. In terms of methodology, their approach differs from previous work in that they use what we call here a Region-Constrained BFS (see section 4.2.1). They exhaustively collect all open user profiles and their list of friends in the 22 largest regional networks (out of the 507 available). First, such Region-Constrained BFS might be appropriate to study particular regions, but it does not provide any general Facebook-wide information, which is the goal of our study. Second, it seems that the percentage of users in the social graph retrieved in [24] is 30%-60% less than the maximum possible in each network. More specifically, it is most likely that for the collection of the social graph, their BFS crawler does not follow users that have their "view profile" privacy setting closed and "view friends" privacy setting open. We infer that by comparing the discrepancy in the percentage of users for those settings as reported in a Facebook privacy study conducted during the same time in [57] *i.e.,* in networks New York, London, Australia, Turkey. In terms of results, the main conclusion in [24] is that the interaction graph should be preferred over social graphs in the analysis of online social networks, since it exhibits more pronounced small-world clustering. In our work, we collect a representative sample of the social graph. This sample can also allow us to fetch a representative sample of user profiles Facebook-wide in the future. In terms of findings, some noteworthy differences from [24] are that we find larger values of the degree-dependent clustering coefficient as well as a significantly higher assortativity coefficient. [18] and [56] have also made publicly available and analyzed datasets corresponding to two university networks from Facebook with many annotated properties for each student. In contrast, we collect a sample of the global Facebook social graph.

Other works that have measured properties of Facebook include [57, 58, 25]. In [57] the authors examine the usage of privacy settings in Myspace and Facebook, and the potential privacy leakage in OSNs. Compared to that work, we have only one common

privacy attribute, "View friends", for which we observe similar results using our unbiased sample. But we also have additional privacy settings and a one node view of the social graph, which allows us to analyze user properties conditioned on their privacy awareness. Bonneau *et al.* [58] demonstrate that many interesting user properties can be accurately approximated just by crawling "public search listings".

In chapter 6, we present the first study to characterize the statistical properties of OSN applications and we model the application installation process. Previous work focused on the characteristics of the social graph itself [59, 49, 50, 16, 21] or the popularity of user-generated content [60]. Nazir *et al.* [61, 10] developed and launched three Facebook applications. Using the rich dataset gathered through these applications, they analyze user activities of Facebook applications and provide a characterization of network-level effects of such applications.

In Chapter 5 we perform measurements of `Last.fm`- an Internet website for music with social networking features. The rich features available in `Last.fm` (users, tags, artists, tracks etc.) have motivated previous data collection efforts. Some examples include [62], which develops a track recommendation system using social tags and friendship between users, [63], which examines user similarity to predict social links, and [64], which explores the meaning of friendship in `Last.fm` through survey sampling. Interestingly, [64] reports that 33% of friends in `Last.fm` have met face-to-face and that 51% are also friends on another OSN site. We emphasize that the importance of a representative sample is crucial to the usefulness of such datasets.

Finally, there is a large body of work on the collection and analysis of datasets for platforms or services that are not pure online social networks but include social networking features. To mention a few examples, Liben-Nowell *et al.* [65] studied the `LiveJournal` online community and showed a strong relationship between friendship and geography in social networks. Cha *et al.* [60] presented a vast data-driven analysis of user generated content

video popularity distributions by using data collected from YouTube and Daum. Gill *et al.* [66] also studied a wide range of features of YouTube traffic, including usage patterns, file properties, popularity and referencing characteristics. A noteworthy insight from their results is that although YouTube can benefit from caching as with the traditional Web, additional meta-data provided by social networking features, (*e.g.,* user ratings, video categories, etc.) could be exploited to improve the caching strategy.

Chapter 3

Crawling Online Social Networks

3.1 What is crawling?

The term crawling has been extensively used in the past to refer to the exploration of the World Wide Web graph, where the objects of interest are web pages. Online social networks revolve around users, thus the users are the objects of interest. Here, we define an OSN crawler to be a computer program that explores (or visits or samples) users in the social graph using an orderly method.

Algorithm 1 Operation of an OSN crawler

```
Initialize v ← v_0.
while NOT FINISHED do
    Fetch data for user v and obtain N(v), the list of friends of v
    Add N(v) to list of observed users, according to used technique
    Select next user v from the list of observed users, according to used technique
end while
return all users v
```

Algorithm 1 describes the operation of a typical OSN crawler. It begins from a seed user v_0 and proceeds iteratively to explore the unknown social graph. The primitive required for

crawling is the ability to list the friends of a user at each iteration. A crawler selects the next user to explore from the list of previously observed users, depending on the particular method used (*e.g.,* BFS, Random Walk etc, as explained later).

3.2 Data Collection

The value of an online social network is largely defined by the richness and magnitude of the user data it stores. Typically, OSN operators are looking for ways to monetize such user data without alienating their existing user base. Understandably, OSN companies are unwilling to share their users' data since this could lead to loss of confidence and undermine their core business. Furthermore, there are legitimate privacy concerns with sharing such data since users' profiles map to their real life personalities. Thus, in most cases researchers resort to collecting their own data from OSNs. Next, we list the two most common ways that researchers actively collect such data.

One way to collect data is to perform data scraping straight from the OSN website. That usually involves: i) requesting specific URLs and fetching the corresponding web pages, similar to using a web browser ii) parsing the returned HTML page and extracting the relevant information. For simple cases, regular expressions might be the easiest way to parse HTML files. For other cases, tools that interpret the structure of the HTML file [67], might be more appropriate. Data scraping needs knowledge of the URL structure of the website and periodic maintenance of the HTML parsing code due to potential changes in the server-side source code. Nevertheless, sometimes it is the only way to collect the user properties of interest.

Another way to collect data is to use API (Application Programming Interface) calls, if they are supported by the OSN operator. APIs were recently introduced by OSN companies to

selectively give access to their data to third parties and encourage innovation. This clearly worked for `Facebook`, with more than 500K applications created within three years. API calls allow us to specify precisely what data to obtain and thus have the potential to decrease by hundreds of times the amount of traffic we are generating during crawling. On the downside, API calls might be subjected to controlled rate limiting and constrained access.

3.2.1 Legal Considerations

Data collection of privacy sensitive information is a controversial issue these days. Therefore, we carefully designed our research study in the following ways so as not to cross the boundaries. First, we obtain only publicly declared list of friends and publicly available user profiles. Within this scope, the UC Irvine Human Research Protections Program concluded that our project does not qualify as human subjects research and is not subject to IRB review and approval. Second, we did not attempt to circumvent in any way user-specified privacy settings. Third, our crawlers are built so as not to overload the OSNs we are fetching data from and cause performance issues. We minimize, to the extent possible, redundant data fetching and according to our rough estimates, the traffic generated by our requests is negligible compared to the total traffic demand in such OSNs. Finally, our data releases have anonymized identifiers of users and other entities.

3.2.2 Challenges

There are technical challenges in crawling online social networks. Here we go over a few of the challenges we faced while crawling the social graph of OSNs.

Part of the allure of online social networks is the interactive experience with rich content that they offer to users. Such interactive web sites are typically developed by using client-

side application logic (usually written in Javascript). This technology, also referred to as AJAX programming, enables asynchronous loading of web content. Therefore, traditional web crawlers would need modifications to be able to extract the full list of URLs when crawling OSNs. In our case, this means that we need to customize crawling for a specific OSN, if data scraping is going to be needed.

Another challenge we need to face are defense mechanisms against automated data mining. Most OSNs require a logged-in account to expose information and might monitor the download rate during crawling. Mislove *et al.* [21] reported rate limits per IP while crawling `Orkut` and in our crawls we experienced banned accounts, probably due to excessive traffic. As expected, data access is subject to the individual user privacy settings but we observed that for the same access level, API calls are usually more restrictive compared to data scraping *i.e.,* there is information for which access is allowed through the web interface but denied through API calls.

The large size of most OSNs, with tens of millions to hundreds of millions of users, make it difficult to obtain a full view of users and their relations. Practically, in a given time frame we can only crawl a subset of the users and their corresponding relations. Therefore the quality of the sample becomes crucial and the challenge is to deal with potential biases that could arise during the data collection. In most of this thesis we follow statistically principled approaches and collect representative samples of OSN users, appropriate for further statistical analysis.

Online social networks are growing daily by registering new members (nodes) and adding new relations (edges). Unfriending and account deletions are also possible but happen much less frequently. In the OSNs we crawled, we can confirm a densification of the social graph as described in [68]. Fortunately, the user growth per day is quite small as a percentage of the total population, even though it is considerable in absolute numbers. However this means that our crawlers need to fetch data from the crawled OSN as fast as

possible because the longer the data collection time, the larger the effect of user growth in the collected sample. To address this challenge we built high performance crawlers that perform distributed data fetching and take advantage of the specifics of each crawling methodology.

3.2.3 Implementation

Here we describe the system level implementation of our crawlers and show that our ability to parallelize the crawling process depends on the selected methodology. In Chapters 4 and 5 we show that the selection of a crawling methodology is critical for our ability to obtain a representative sample of users.

Fig 3.1 contains an overview of our distributed crawling process. First, we assume that we have in our disposal a large number of machines with limited memory (100 Mbytes-1GBytes RAM) and disk space (up to 5Gbytes). We use these machines to parallelize our crawling and shorten the data collection time. We have up to three layers of parallelism in each machine. Each crawling machine runs one or more crawling processes. Each crawling process shares one user account between multiple crawling threads within it. Each crawling thread fetches data asynchronously where possible *i.e.,* in `Last.fm` the web pages for events and groups are fetched in parallel.

The logic of each crawling methodology is implemented inside each crawling thread. The data, obtained from the OSN either through API calls or data scraping, is extracted and dumped in a text format for offline processing. [1]

The determination of the number of processes per machine and threads per process mostly depends on the defenses that the OSN has implemented. If the OSN rate limits per IP ad-

[1]The alternative way to proceed is to store the data directly in a relational database.

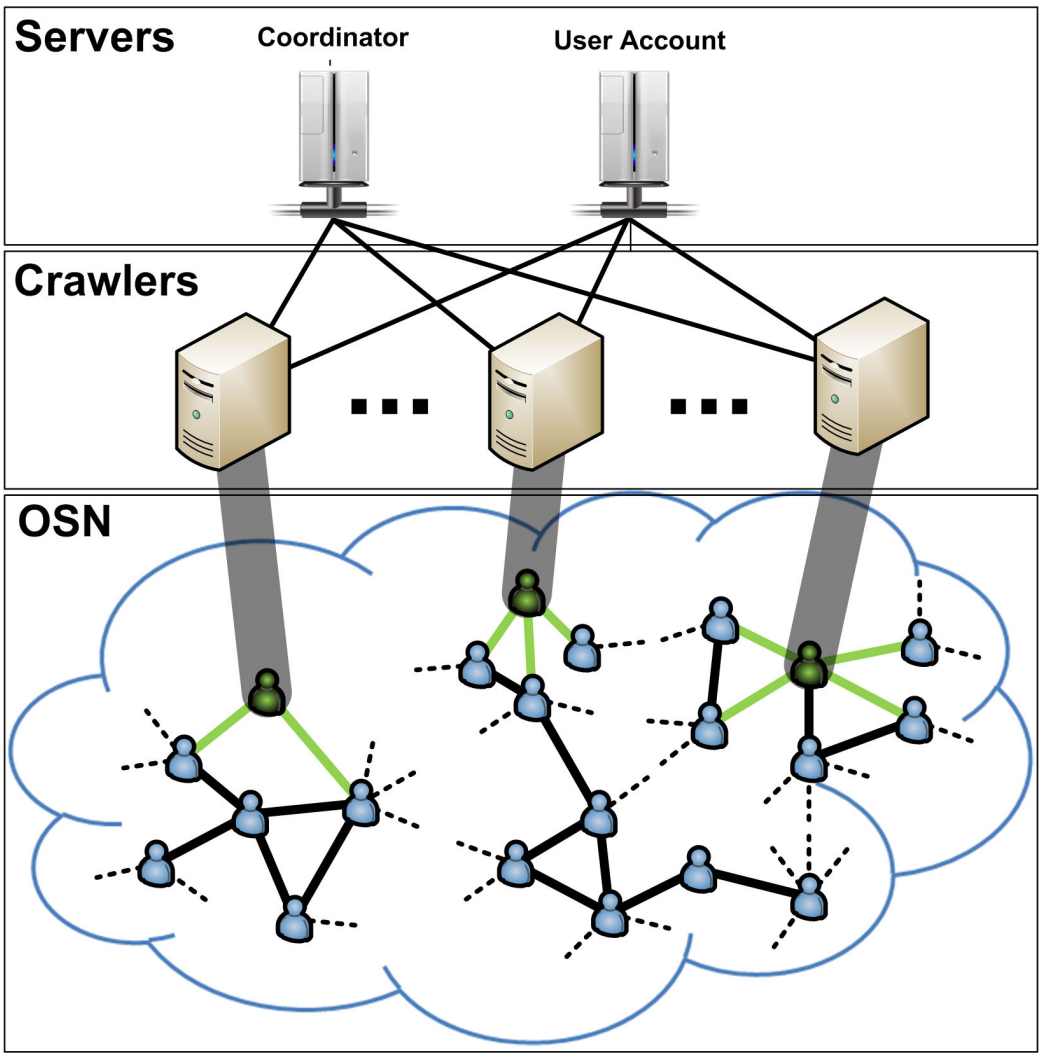


Figure 3.1: Distributed Crawling of an Online Social Network

dress then one process per machine with a large number of threads will suffice. On the other hand, if the OSN rate limits per user account then we could increase the crawling rate by running more processes per machine. The crawling technique used is another consideration when choosing the number of threads per process. For example, in random walks, each crawling thread corresponds to a walk. If we were to run a large number of crawling threads within a process, we would have to make sure that each walk has converged to the stationary distribution before we can stop crawling. Therefore, if there is a sample size budget, running a large number of threads per process might not be advisable.

Second, we use one of the machines in our disposal to better coordinate the distributed crawling process. The coordinator server supports the following operations. i) it contains a mechanism to rate-limit the number of connections or amount of bandwidth over the whole cluster. We can use it so as not to cause a high load to the crawled OSN. ii) it keeps track of already fetched users to avoid fetching duplicate data. As a result, the crawling process continues in a faster pace, since each request to the OSN returns new information. Additionally, we can increase on-demand the number of crawling machines without redundant data fetching. iii) it supports the maintenance of a data structure which stores the crawl frontier *i.e.,* a queue for BFS iv) it has the ability to cache data for re-use by crawling machines *i.e.,* for random walks which perform sampling with replacement.

Finally, we use a user account server to handle the authentication (login account or API account) per crawling process. When a crawling process is initiated, it requests an unused account from the user account server. The crawling process is activated only if a valid account is available. Furthermore, a crawling process has the ability to detect an invalidated account and request from the user account server to erase it and to return a new valid account. The user account server contains the accounts available for use by the crawling processes. The assumption is that the crawling administrator will populate the user account server with valid accounts at the beginning of the crawl and maintain a number of valid accounts comparable to the number of crawling processes during the crawl. The user account server keeps track of the used accounts such that no two crawling processes use the same account. It also monitors for graceful or ungraceful exits of crawling processes to free up unnecessarily used accounts.

Breadth-First-Search. To parallelize the BFS crawling technique, we use in the coordinator a queue, that maintains the crawl frontier, and a hash table, that makes sure we do not refetch the same data. The sequence of events at each crawling thread and the coordinator are as follows. A crawling thread retrieves the next unexplored user from the coordinator queue

and fetches the user's data and friend list. It then sends the friend list to the coordinator and requests the next user. At each request from a crawling thread, the coordinator returns the user at the front of the queue and marks the hash table. When the coordinator receives a list of friends, every user in the list that is not contained in the hash table is appended at the end of the coordinator queue.

Random Walks. To parallelize random walk techniques, we use in the coordinator a size constrained hash table, that caches data that might be later re-used. This is necessary because random walks perform sampling with replacement. Nodes can be revisited when the walk explores a highly clustered area. By caching the list of friends for a small window of previous samples, we can avoid re-fetching the list of friends. Usually a FIFO cache with size 1000 per crawling thread was sufficient for our crawling.

Other considerations. The coordinator, the only centralized component in our diagram, is a potential bottleneck. However, all operations it supports could be decentralized. For example, data caching could be replaced with a cluster of machines running memcached [69] or run individually in each crawling process.

More details about the data collection process in `Facebook` and `Last.fm` (*i.e.,* used API calls or data scraping process) are provided in Appendix B.

3.3 Summary

In this part, we described the key design mechanisms of the distributed crawlers that we later use to collect data from `Facebook` and `Last.fm`. Although, the crawlers we implemented were customized for each OSN, they shared large parts of common functionality. We believe our work is a step towards general-purpose crawlers for online social networks. The parameters for such crawlers could include the crawling methodology, the user prop-

erties of interest, formal diagnostics that assess convergence online, storage format specifications (*i.e.,* text dumps, sqlite, centralized SQL). Easy-to-use wizards could guide novice users in selecting user properties of interest from data scrapes or available API calls.

Chapter 4

Social graph sampling

4.1 Overview

Our primary goal in this chapter is to explore the utility of various graph-crawling algorithms for producing a representative sample of `Facebook` users, appropriate for further statistical analysis. We crawl `Facebook`'s web front-end, which can be challenging in practice. A second goal is to introduce the use of formal convergence diagnostics to assess sample quality in an online fashion. These methods allow us to determine, in the absence of a ground truth, when a sample is adequate for subsequent use, and hence when it is safe to stop sampling, which is a critical issue in implementation. In the process of applying these methods to `Facebook`, we hope to illuminate more general characteristics of crawling methods that can be used to achieve asymptotically unbiased sampling of Online Social Networks.

In terms of methodology, we consider several candidate crawling techniques. First, we consider Breadth-First-Search (BFS) - the heretofore most widely used technique for measurements of OSNs [21, 16] and `Facebook` [24]. BFS is known to introduce bias towards

high degree nodes; moreover, this bias is not formally characterized. Second, we consider Random Walk (RW) sampling, which also leads to bias towards high degree nodes, but at least its bias can be quantified by Markov Chain analysis and thus can be corrected via appropriate re-weighting (RWRW). Third, we consider the Metropolis-Hastings Random Walk (MHRW) that directly achieves the goal, *i.e.,* yields a uniform stationary distribution of nodes (users). This technique has been used in the past for P2P sampling [34], recently for a few OSNs [42, 43], but not for `Facebook`. Finally, we also collect a sample that represents the "ground truth" (UNI) *i.e.,* a truly uniform sample of `Facebook` userIDs, selected by a rejection sampling procedure from the system's 32-bit ID space. We note, however, that such ground truth is in general unavailable or inefficient to obtain, as discussed in Section 4.2.3; in contrast, crawling friendship relations is a fundamental primitive available in OSNs and, we believe, the right building block for designing sampling techniques for OSNs. Therefore, we believe that out proposed approach is applicable to any OSN. We compare all sampling methods in terms of their bias and convergence properties. We also provide recommendations for their use in practice: *e.g.,* we implement online formal convergence diagnostic tests and parallel walks for improved speed; we also discuss pros and cons of MHRW vs. RWRW in practice.

In terms of results, we show that MHRW and RWRW work remarkably well. We demonstrate their aggregate statistical properties, validating them against the known uniform sample, and show how our formal diagnostics can be used to identify convergence during the sampling process. In contrast, we find that the more traditional methods - BFS and RW - lead to a significant bias in the case of `Facebook`. Finally, using one of our validated samples (MHRW), we also characterize some key properties of `Facebook`; we find some of them to be substantively different from what was previously believed based on biased samples. The collected datasets are made publicly available for use by the research community at [19].

4.2 Sampling Methodology

Facebook can be modeled as an undirected graph $G = (V, E)$, where V is a set of nodes (Facebook users) and E is a set of edges (Facebook friendship relationships). Let k_v be the degree of node v. We assume the following in our problem statement: (i) we are interested only in the publicly declared lists of friends, which, under default privacy settings, are available to any logged-in user; (ii) we are not interested in isolated users, *i.e.,* users without any declared friends; (iii) we consider that the Facebook graph remains *static* during our crawling. We justify and discuss in detail assumption (iii) in Section 4.3.

4.2.1 Sampling Methods

The crawling of the social graph starts from an initial node and proceeds iteratively. In every operation, we visit a node and discover all its neighbors. There are many ways, depending on the particular sampling method, in which we can proceed. In this section, we describe the sampling methods we implemented and some close variations. Our ultimate goal is to obtain a uniform random sample of users in Facebook.

Breadth First Search (BFS)

BFS is a classic graph traversal algorithm which starts from a seed node and progressively explores neighboring nodes. At each new iteration the earliest explored but not-yet-visited node is selected next. As this method discovers all nodes within some distance from the starting point, an incomplete BFS is likely to densely cover only some specific region of the graph. BFS is known to be biased towards high degree nodes [26, 70] and no statistical properties can be proven for it. Nevertheless, BFS-based crawling and its variants, such as snowball, are widely used techniques for network measurements.

Region-Constrained BFS (RC-BFS)

In some cases it is possible and reasonable to try to collect all nodes of some particular type, and no other nodes. This can be achieved by constraining BFS to these nodes only, and by letting it run until the neighbor list is exhausted. This technique was often used in web crawling to discover all pages that belong to a particular domain, e.g. to `nd.edu` [71]. It was also applied to `Facebook` to discover all users in a particular regional network [24]. Such Region-Constrained BFS is good to study particular regions, but does not provide any general `Facebook`-wide information.

Random Walk (RW)

Another classic sampling technique is the classic random walk [31]. In this case, the next-hop node w is chosen uniformly at random among the neighbors of the current node v. Therefore, the probability of moving from v to w is

$$P_{v,w}^{RW} = \begin{cases} \frac{1}{k_v} & \text{if } w \text{ is a neighbor of } v, \\ 0 & \text{otherwise.} \end{cases}$$

The random walk has been deeply studied; *e.g.,* see [31] for an excellent survey. It is simple and there are analytical results on its stationary distribution and convergence time. Unfortunately, it is also inherently biased. Indeed, in a connected and aperiodic graph, the probability of being at the particular node v converges with time to:

$$\pi_v^{RW} = \frac{k_v}{2 \cdot |E|}$$

i.e. the classic RW samples nodes w.p. $\pi_v^{RW} \sim k_v$. This is clearly biased towards high degree nodes; *E.g.,* a node with twice the degree will be visited by RW two times more

often. Moreover, we show later that many other node properties in OSNs are correlated with the node degree; these include, for example, the privacy settings, clustering coefficient, network membership, or even the 32 bit user ID. As a result of this correlation, all these metrics are estimated with bias by RW sampling.

Re-Weighted Random Walk (RWRW)

A natural next step is to crawl the network using RW, but to correct for the degree bias by an appropriate re-weighting of the measured values. This can be done using the Hansen-Hurwitz estimator [72] as first shown in [39, 73] for random walks [72, 74] and also later used in [35]. RWRW is essentially a special case of importance sampling [74] reweighted by the stationary distribution of the RW MCMC. In general, Re-Weighted Markov Chain Monte Carlo Walk involve two steps: an initial MCMC process involving a (possibly biased) walk on the graph is used to generate a sample with known asymptotic statistical properties; and a secondary weighting or resampling process is then employed to enable the use of this sample to approximate some distribution or estimand of interest. Our MHRW process, presented next, is also a special case of this procedure in which the MCMC algorithm is chosen to produce a uniform sample, and no secondary re-weighting procedure is needed.

Here is how to apply the Hansen-Hurwitz estimator. Consider a stationary random walk that has visited $V = v_1, ...v_n$ unique nodes. Each node can belong to one of m groups with respect to a property of interest A, which might be the degree, network size or any other discrete-valued node property. Let $(A_1, A_2, .., A_m)$ be all possible values of A and corresponding groups; $\cup_1^m A_i = V$. *E.g.,* if the property of interest is the node degree, A_i contains all nodes u that have degree $k_u = i$. To estimate the probability distribution of A,

we need to estimate the proportion of nodes with value $A_i,\ i = 1, ..m$:

$$\hat{p}(A_i) = \frac{\sum_{u \in A_i} 1/k_u}{\sum_{u \in V} 1/k_u}$$

Estimators for continuous properties can be obtained using related methods, e.g. kernel density estimators.

Metropolis-Hastings Random Walk (MHRW)

Instead of correcting the bias after the walk, one can appropriately modify the transition probabilities so that it converges to the desired uniform distribution. The Metropolis-Hastings algorithm [40] is a general Markov Chain Monte Carlo (MCMC) technique [41] for sampling from a probability distribution μ that is difficult to sample from directly. In our case, we would like to sample nodes from the uniform distribution $\mu_v = \frac{1}{|V|}$. This can be achieved by the following transition probability:

$$P^{MH}_{v,w} = \begin{cases} \frac{1}{k_v} \cdot \min(1, \frac{k_v}{k_w}) & \text{if } w \text{ is a neighbor of } v, \\ 1 - \sum_{y \neq v} P^{MH}_{v,y} & \text{if } w = v, \\ 0 & \text{otherwise.} \end{cases}$$

It can be shown that the resulting stationary distribution is $\pi^{MH}_v = \frac{1}{|V|}$, which is exactly the uniform distribution we are looking for. $P^{MH}_{v,w}$ implies the following algorithm, which we refer to simply as MHRW:

$v \leftarrow$ initial node.

while stopping criterion not met **do**

Select node w uniformly at random from neighbors of v.

Generate uniformly at random a number $0 \leq p \leq 1$.

if $p \leq \frac{k_v}{k_w}$ **then**

$v \leftarrow w$.

else

Stay at v

end if

end while

In every iteration of MHRW, at the current node v we randomly select a neighbor w and move there w.p. $\min(1, \frac{k_v}{k_w})$. We always accept the move towards a node of smaller degree, and reject some of the moves towards higher degree nodes. This eliminates the bias towards high degree nodes.

Maximum Degree

Another solution to achieve uniform stationary distribution during the walk is to modify RW to *accept only some hops*, as follows:

$$P^{MD}_{u,w} = \begin{cases} \frac{1}{k_{max}} & \text{if } w \text{ is a neighbor of } u, \\ 1 - \frac{k_u}{k_{max}} & \text{if } w = u, \\ 0 & \text{otherwise}, \end{cases}$$

where k_{max} is the maximal allowed node degree. This is applicable in the case of `Facebook` which has a known $k_{max} = 5000$. It can be shown that this walk is asymptotically unbiased. Indeed, $P^{MD}_{u,w}$ is equivalent to adding $k_{max} - k_u$ self-loops at each node u of graph G. In the resulting graph, all nodes have the same degree $k_u = k_{max}$ and thus the stationary distribution is $\pi^{MD}_u = \frac{1}{|V|}$ for all nodes u in V.

The problem with this approach is that we stay at node u for many (on average $\frac{k_{max}}{k_u}$) iterations before leaving it, which significantly increases the convergence time. In our measurements we will not be using Maximum Degree and we will instead prefer applying

the Metropolis-Hastings method to correct the RW bias during the walk.

4.2.2 Convergence

Multiple Parallel Walks

Multiple parallel walks are used in the MCMC literature [41] to improve convergence. Intuitively, if we only have one walk, we might run into a scenario where it is trapped in a certain region while exploring the graph and that may lead to erroneous diagnosis of convergence. Having multiple parallel walks reduces the probability of this happening and allows for more accurate convergence diagnostics.We note that the advantage of multiple random walks is achieved when there is no fixed budget in the number of samples that would lead to many short walks; this is true in our case. An additional advantage of multiple parallel walks, from an implementation point of view, is that it is amenable to parallel implementation from different machines or different threads in the same machine. Some coordination is then required to increase efficiency by not downloading information about nodes that have already been visited by independent walks.

We implemented each of the considered crawling algorithms with several parallel walks. Each walk starts from a different node in $V_0 \subset V$, $|V_0| \geq 1$ ($|V_0| = 28$ in our case) and proceeds independently of the others. The initial nodes V_0 are randomly chosen in different networks. For a fair comparison, we compare multiple MHRWs to multiple RWs and multiple BFSs, all starting from the same set of initial nodes V_0.

Convergence Tests

Valid inferences from MCMC are based on the assumption that the samples are derived from the equilibrium distribution, which is true asymptotically. We will use the term convergence to refer to the state in which the sample has reached the equilibrium distribution. In order to correctly diagnose when convergence occurs, we use standard diagnostic tests developed within the MCMC literature [41].

We would like to use diagnostic tests to answer at least the following convergence questions: i) how many of the initial samples in each walk do we need to discard to lose dependence from the starting point (or burn-in) ? ii) how many samples do we need before we have collected a representative sample?

A standard approach is to run the sampling long enough and to discard a number of initial burn-in samples proactively. But from a practical point of view, the burn-in comes at a cost. In the case of `Facebook`, it is the consumed bandwidth (in the order of gigabytes) and measurement time (days or weeks). It is therefore crucial to assess the convergence of our MCMC sampling, and to decide on appropriate settings of burn-in and total running time.

Given that we do not have knowledge of the target distribution, we can only estimate convergence from the statistical properties of the walks as they are collected. Here we present two standard convergence tests, widely accepted and well documented in the MCMC literature, Geweke [75] and Gelman-Rubin [76], described below. In theory, convergence diagnostics should be used to indicate lack of convergence. In practice however, negative results in regard to non-convergence provide a degree of confidence that the sample has converged. In Section 4.4, we apply these tests on several node properties, including the node degree, userID, network ID and membership in a specific network; please see Section 4.4.1 for details. Below, we briefly outline the rationale of these tests and we refer the interested reader to the references for more details.

Geweke Diagnostic. The Geweke diagnostic [75] detects the convergence of a single Markov chain. Let X be a single walk of samples of our metric of interest. Geweke considers two subwalks of X, its beginning X_a (typically the first 10%), and its end X_b (typically the last 50%). Based on X_a and X_b, we compute the z-statistic

$$z = \frac{E(X_a) - E(X_b)}{\sqrt{Var(X_a) + Var(X_b)}}.$$

With increasing number of iterations, X_a and X_b move further apart, which limits the correlation between them. As they measure the same metric, they should be identically distributed when converged and, according to the law of large numbers, the z values become normally distributed with mean 0 and variance 1. We can declare convergence when most values fall in the $[-1, 1]$ interval.

Gelman-Rubin Diagnostic. Monitoring one long walk has some disadvantages. *E.g.,* if our walk stays long enough in some non-representative region of the parameter space, we might erroneously declare convergence. For this reason, Gelman and Rubin [76] proposed to monitor $m > 1$ walks. The Gelman-Rubin diagnostic [76] monitors convergence of the output of a MCMC process with m multiple walks whose starting points are overdispersed in the target distribution. It indicates convergence when the output from all walks is similar. The Gelman-Rubin diagnostic is applied to a single scalar metric in the walks and it is based on a comparison of within-walk and between-walk variances. More specifically, the variance of the equilibrium distribution is estimated in two ways different ways: i) the empirical variance from all walks combined, defined as V, using both between-walk and within-walk information. ii) the mean of all empirical variances within each walk, defined as W. Non-convergence is detected when these two variance estimates do not reach the same value. The rationale in the latter case is that the variance of the estimate V over all walks is expected to be higher, because of the selection of overdispersed starting points, while the variance of the within-walk estimate W is expected to be lower, since the individ-

ual walks have not mixed enough. On the other hand, if all m walks have converged, then both estimates of variance converge to the same value and the Gelman-Rubin diagnostic R approaches the value 1[1].

Assume that we have $m >= 2$ parallel walks of length n and let θ_j^i be our metric of interest for walk j and iteration i. The steps to calculate the Gelman-Rubin diagnostic are as follows.

1. Compute W, the mean of all empirical variances within each walk.

   $W = \frac{1}{m(n-1)} \sum_{j=1}^{m} \sum_{i=1}^{n} \left(\theta_j^i - \bar{\theta}_j\right)^2$

2. Compute B, the variance of the mean across the walks.

   $B = \frac{n}{m-1} \sum_{j=1}^{m} \left(\theta_j - \bar{\theta}\right)^2$

3. Compute the target variance by a weighted average of W and B.

   $\hat{\sigma}^2 = \left(\frac{n-1}{n}\right) W + \frac{1}{n} B$

4. The convergence diagnostic is based on the assumption that the target distribution is normal. A Bayesian credible interval can be constructed using a t-distribution with mean, variance estimated as below.

   $\hat{\mu} = \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} \theta_j^i$

   $\hat{V} = \hat{\sigma}^2 + \frac{B}{mn}$

5. Compute the convergence diagnostic R for all variables of interest.

   $R = \frac{\hat{V}}{W}$

   Values near 1 indicate convergence.

[1] Convergence is declared typically for values smaller than 1.02.

Finally, we note that even after convergence is diagnosed by the above tests and initial samples are discarded, strong correlation of consecutive samples in the walk may affect sequential analysis. This is typically addressed by thinning, *i.e.,* keeping only one every r samples. In our approach, instead of thinning, we do sub-sampling of nodes after burn-in, which has essentially the same effect.

4.2.3 Ground Truth: Uniform Sample (UNI)

Assessing the quality of any graph sampling method on an unknown graph, as it is the case when measuring real systems, is a challenging task. In order to have a "ground truth" to compare against, the performance of such methods is typically tested on artificial graphs (using models such as Erdös-Rényi, Watts-Strogatz or Barabási-Albert, etc.). This has the disadvantage that one can never be sure that the results can be generalized to real networks that do not follow the simulated graph models and parameters.

Fortunately, `Facebook` was an exception during the time period we performed our measurements. We capitalized on a unique opportunity to obtain a truly uniform sample of `Facebook` users by generating uniformly random 32-bit userIDs, and by polling `Facebook` about their existence. If the userID exists, we keep it, otherwise we discard it. This simple method is a textbook technique known as *rejection sampling* [77] and in general it allows to sample from any distribution of interest, which in our case is the uniform. In particular, it guarantees to select uniformly random userIDs from the existing `Facebook` users regardless of their actual distribution in the userID space, *i.e., even* if though the userIDs are *not* allocated sequentially or evenly across the userID space. For completeness, we derive this property of UNI sampling in Appendix A. We refer to this method as 'UNI', and use it as a ground-truth uniform sampler.

Although UNI sampling solves the problem of uniform node sampling in `Facebook`, our

methodology (and results) remain important. There are two necessary conditions for UNI to work. First, such an operation must be supported by the system. Facebook currently allows to verify the existence of an arbitrary userID and retrieve her list of friends; however, Facebook may remove this option in the future, *e.g.,* for security reasons. Second, the userID space must not be sparse for this operation to be efficient. During our data collection (April-May 2009) the number of Facebook users ($\sim 200 \times 10^6$) was comparable to the size of the userID space ($2^{32} \sim 4.3 \times 10^9$), resulting in about one user retrieved per 22 attempts on average. If the userID was 64 bit long or consisting of strings of arbitrary length, UNI would had been infeasible. To mention a few such cases at the same time frame: Orkut had a 64bit userID and hi5 used a concatenation of userID+Name. Additionally, we now know that within days to weeks after our measurements were completed, Facebook changed its userID allocation space from 32 bit to 64 bit [78]. We speculate that the main reason for such a change was to allocate more userID space but we do not preclude security reasons behind this change i.e. to hinder efforts of data collection. Section 4.4.2 contains more information about userID space usage in Facebook on April 2009.

In summary, we were fortunate to be able to obtain the ground truth, through uniform sampling of userIDs. This allowed us to demonstrate that our results perfectly agree with it. However, crawling friendship relations is a fundamental primitive available in all OSNs and, we believe, the right building block for designing sampling techniques in OSNs, in the long run.

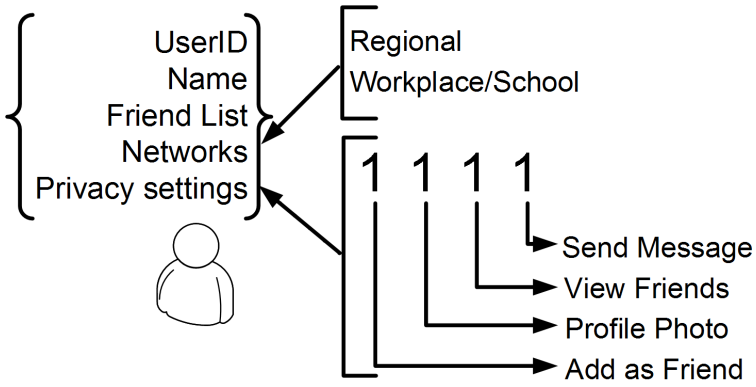


Figure 4.1: Information that we obtain about a user.

4.3 Data Collection

4.3.1 User properties of interest

Fig. 4.1 summarizes the information that we obtain about each user that we visit during our crawls.

Name and userID. Each user is uniquely defined by her userID, which is a 32-bit number [2]. Each user presumably provides her real name. The names do not have to be unique.

Friends list. A core idea in social networks is the possibility to declare friendship between users. In `Facebook`, friendship is always mutual and must be accepted by both sides. Thus the social network is undirected.

Networks. `Facebook` uses the notion of "networks" to organize its users. There are two types of networks. The first type is *regional* (geographical) networks[3]. There are 507 predefined regional networks that correspond to cities, regions, and countries around the world. A user can freely join any regional network but can be a member of only one regional network at a time. Changes are allowed, but no more than twice every 6 months

[2] `Facebook` changed to 64-bit user ID space after May 2009 [78]
[3] Regional networks were phased out starting from June 2009[79]

bit	attribute	explanation
1	Add as friend	=1 if w can propose to 'friend' u
2	Photo	=1 if w can see the profile photo of u
3	View friends	=1 if w can see the friends of u
4	Send message	=1 if w can send a message to u

Table 4.1: Basic privacy settings of a user u with respect to her non-friend w.

(April 2009).

The second type of networks contain user affiliations with colleges, workplaces, and high schools and have stricter membership criteria: they require a valid email account from the corresponding domain i.e. to belong in the UC Irvine network you have to provide a "@uci.edu" email account. On the other hand, a user can belong to many such networks.

For each sampled user we collect, at no additional cost, full network membership information of all her friends. (Section B.1 contains more details).

Privacy settings Q_v. Each user u can restrict the amount of information revealed to any non-friend node w, as well as the possibility of interaction with w. These are captured by four basic binary privacy attributes, as described in Table 4.1. We refer to the resulting 4-bit number as privacy settings Q_v of node v. By default, `Facebook` sets $Q_v = 1111$ (allow all).

Profiles. Much more information about a user can potentially be obtained by viewing her profile. Unless restricted by the user, the profile can be displayed by her friends and users from the same network.

In this work, we do not collect any profile, even if it is open/publicly available. We study only the basic information mentioned above.

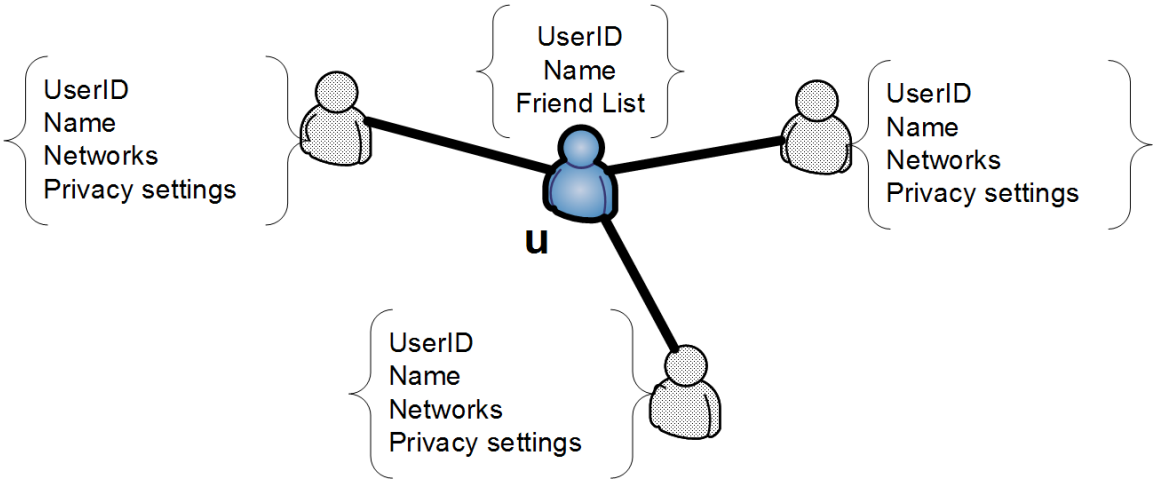


Figure 4.2: Basic node information collected when visiting a user u.

4.3.2 Collection Process

Crawling `Facebook` to collect this information faces several challenges, which we describe below, along with our solutions.

One node view. Fig. 4.2 shows the information collected when visiting the "show friends" web page of a given user u, which we refer to as *basic node information*. Because the Network and Privacy information of u are not directly visible, we collect these indirectly by visiting one of u's friends and using the "show friends" feature.

Invalid nodes. There are two types of nodes that we declare *invalid*. First, if a user u decides to hide her friends and to set the privacy settings to $Q_u = **0*$, the crawl cannot continue. We address this problem by backtracking to the previous node and continuing the crawl from there, as if u was never selected. Second, there exist nodes with degree $k_v = 0$; these are not reachable by any crawls, but we stumble upon them during the UNI sampling of the userID space. Discarding both types of nodes is consistent with our problem statement, where we already declared that we exclude such nodes (either not publicly available or isolated) from the graph we want to sample.

Implementation Details about the Crawls. In Section 4.2.2, we discussed the advantages of using multiple parallel walks both in terms of convergence and implementation. We ran $|V_0| = 28$ different independent crawls for each algorithm, namely MHRW, BFS and RW, all seeded at the same initial, randomly selected nodes V_0.

We let each independent crawl continue until exactly 81K samples are collected. We count towards this value all repetitions, such as the self-transitions of MHRW, and returning to an already visited state (RW and MHRW). In addition to the 28×3 crawls (BFS, RW and MHRW), we ran the UNI sampling until we collected 984K valid users, which is comparable to the 957K unique users collected with MHRW.

In terms of implementation, we developed a multi-threaded crawler in Python and used a cluster of crawling machines machines to reduce the data collection time. A crawler does HTML scraping to extract the basic node information (Fig. 4.2) of each visited node, as described in Appendix B.1. We also have a server that coordinates the crawls so as to avoid downloading duplicate information of previously visited users. This coordination brings many benefits: we take advantage of the parallel walks in the sampling methodology to speed up the process, we do not overload the FB platform with duplicate requests, and the crawling process continues in a faster pace since each request to FB servers returns new information.

Ego Networks. The sample of nodes collected by our method enables us to study many features of FB users in a statistically unbiased manner. However, more elaborate topological measures, such as clustering coefficient and assortativity, cannot be estimated based purely on a single-node view. For this reason, after finishing the BFS, RW, MHRW crawls, we decided to also collect a number of *extended ego nets* for a sub-sample of the MHRW dataset, which is representative of the whole `Facebook` population.

In the social network literature [20], the ego net consists of the ego, all alters (neighbors),

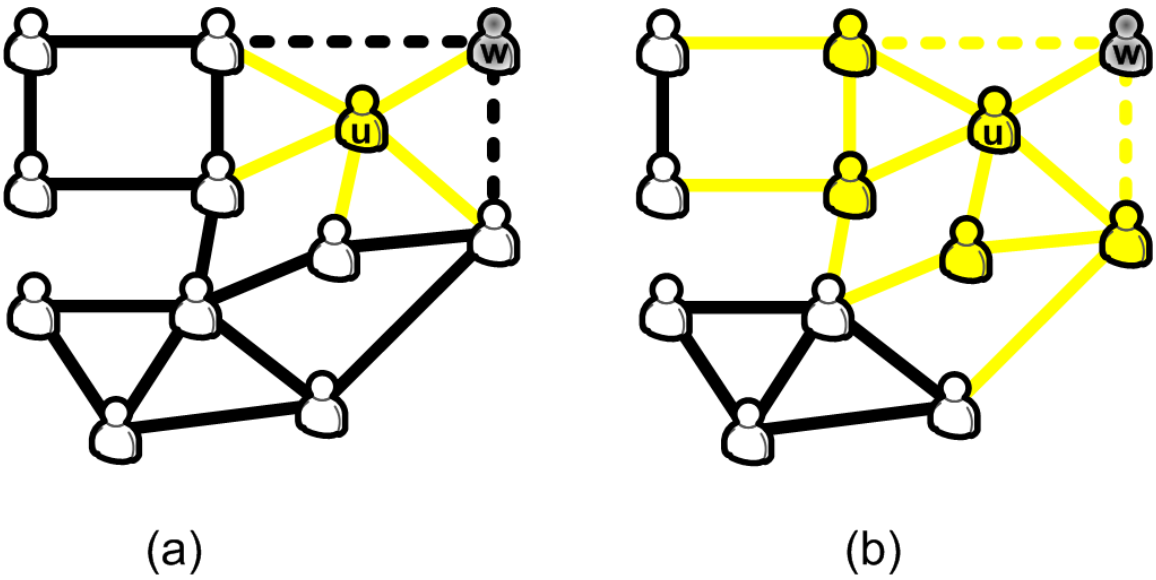


Figure 4.3: (a) Sampled user u with observed edges in yellow color. (b) The extended ego network of user u with observed nodes and edges in yellow color. Invalid neighbor w, whose privacy settings $Q_w = **0*$ do not allow friend listing, is discarded.

and all edges among alters. Here we collect the egonet plus all observed edges of the alters, since there is no additional sampling cost. We call it the extended ego net and we illustrate the concept in Fig 4.3. We use the extended egonet sample differently from the Random Node Neighbor (RNN) sample presented in [37]. We are interested in estimating properties of the ego nodes only whereas RNN [37] looks at the induced subgraph of all sampled nodes and is interested in estimating properties of the ego and all alters. Therefore, the sample of extended egonets, that we collect in this work, is expected to capture very well community structure properties (*i.e.,* clustering coefficient) of the whole Facebook population.

Collecting an egonet requires sampling 100 nodes per node (ego) on average, which is impossible to do for all visited nodes. For this reason, we collect the ego-nets only for $\sim$37K nodes, randomly selected from all nodes in MHRW (considering all 28 walks, after the 6000 'burn-in' period). This sub-sampling has the side advantage that it eliminates the correlation of consecutive nodes in the same crawl, as discussed in Section 4.2.2.

Number of egonets	37K
Number of neighbors	9.3M
Number of unique neighbors	5.8M
Crawling period	04/24-05/01
Avg Clustering coefficient	0.16
Avg Assortativity	0.233

Table 4.2: Ego networks collected for 37K nodes, randomly selected from the users in the MHRW dataset.

	MHRW	RW	BFS	UNI
Total number of valid users	28×81K	28×81K	28×81K	984K
Total number of *unique* users	957K	2.19M	2.20M	984K
Total number of *unique* neighbors	72.2M	120.1M	96M	58.4M
Crawling period	04/18-04/23	05/03-05/08	04/30-05/03	04/22-04/30
Avg Degree	95.2	338	323	94.1
Median Degree	40	234	208	38

Table 4.3: Collected datasets by different algorithms during April-May 2009. The crawling algorithms (MHRW, RW and BFS) consist of 28 parallel walks each, with the same 28 starting points. UNI is the uniform sample of userIDs.

4.3.3 Description of Datasets

Information about the datasets we collected is summarized in Tables 4.3, 4.4, and 4.2. This information refers to all sampled nodes, before discarding any "burn-in". The MHRW crawl contains 957K unique nodes, which is less than the $28 \times 81K = 2.26M$ iterations in all 28 random walks; this is because MHRW may reject a node transition and instead sample the same node during a walk. The number of rejected nodes in all MHRW walks, without repetitions, adds up to 645K nodes. In the BFS crawl, we observe that the overlap of nodes between the 28 different BFS instances is very small: 97% of the nodes are unique, which also confirms that the random seeding chose different areas of `Facebook`. In the RW crawl, 97% of the nodes are unique.

Table 4.4 shows that the percentage of common users between the MHRW, RW, BFS and UNI datasets is very small, as expected. The largest observed, but still objectively small, overlap is between RW and BFS and is probably due to the common starting points selected.

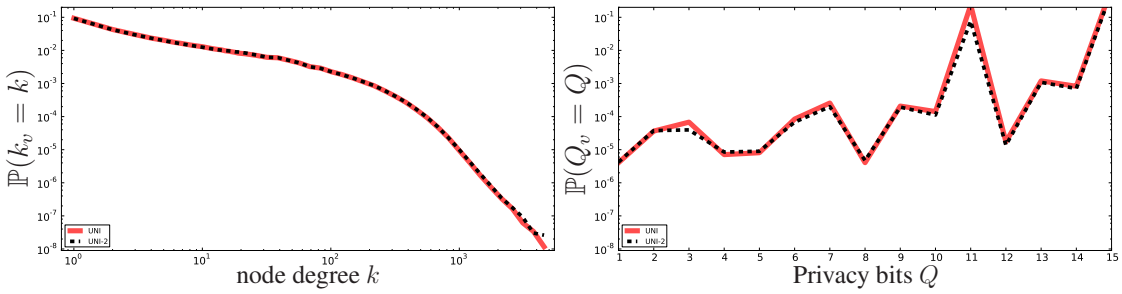
	MHRW	RW	BFS	UNI
MHRW	-	16.2K	15.1K	4.1K
RW			64.2K	9.3K
BFS				15.1K
UNI				

Table 4.4: The overlap between different datasets is small

To collect the UNI dataset, we checked $\sim$ 18.5M user IDs picked uniformly at random from $[1, 2^{32}]$. Out of them, only $1,216K$ users existed. Among them, 228K users had zero friends; we discarded these isolated users to be consistent with our problem statement. This results in a set of 984K valid users with at least one friend each. Considering that the percentage of zero degree nodes is unusually high, we manually confirmed that 200 of the discarded users have indeed zero friends.

To analyze topological characteristics of the `Facebook` population, we collected $\sim$ 37K egonets that contain basic node information (see Fig 4.2) for $\sim$ 5.8M unique neighbors. Table 4.2 contains a summary of the egonets dataset, including properties that we analyze in Section 4.5.

Overall, as a result of (i) the multiple crawls, namely BFS, RW, MHRW, UNI and (ii) the ego networks of a sub-sample of the Metropolis walk, we collected 11.6 million unique nodes with basic node information. As a result, the total number of unique users (including the sampled nodes and the neighbors in their egonets) for which we have basic privacy and network membership information becomes immense. In particular, we have such data for $\sim$ 172 million unique `Facebook` users. This is a significant sample by itself given that `Facebook` had reported having close to 200 million active users during the time of these measurements. We use this information in Section 4.5 to analyze topology dependent and privacy related properties.



(a) Node Degree

(b) Privacy Settings Q_v

Figure 4.4: PDF of two user properties for UNI samples obtained 45 days apart

4.3.4 Topology Change During Sampling Process

We treat the FB graph as static during the execution of our crawls, despite the fact that `Facebook` is growing. We believe that this assumption is a valid approximation in practice for several reasons. First, the FB characteristics change in longer timescales than the duration of our walks. During the period that we did our crawls (see table 4.3), `Facebook` was growing at a rate of $450K/day$ as reported by websites such as [80, 81]. With a population of $\sim$200M users during that period, this translates to a growth of 0.22% of users/day. Each of our crawls lasted around 4-7 days (during which, the total FB growth was 0.9%-1.5%); in fact, our convergence analysis shows that the process converged even faster, *i.e.,* in only one day. Therefore, the growth of `Facebook` was negligible during our crawls. Second, the FB social (not interaction) graph is much more static than P2P systems that are known to have high churn; in the latter case, dealing with dynamic graphs becomes important [34, 52]. Third, we obtained empirical evidence by comparing our metrics of interest between the UNI sample of Table 4.3 and a similarly sized UNI sample obtained 45 days later. The distributions we obtained were virtually identical; Fig 4.4 shows the distribution of node degree and privacy.

Thus, while issues of dynamics are important to consider when sampling changing graphs, they appear not to be problematic for this particular study.

4.3.5 Data Release

From the datasets we collected, we have made available to the research community the Metropolis Hasting Random Walk and UNI crawl types. They constitute some of the first representative samples of OSN users, appropriate for further statistical analysis. For each sampled user in the released datasets, we provide the number of times sampled, the total number of friends, the 4-bit privacy settings and network affiliations. To preserve user privacy, we have anonymized all userIDs and networkIDs. More details and downloading instructions are contained at `http://odysseas.calit2.uci.edu/research/osn.html`

4.4 Evaluation of Sampling Techniques

In this section, we evaluate all candidate methodologies, namely BFS, RW and RWRW, MHRW, in terms of convergence and estimation bias. First, in Section 4.4.1, we study in detail the convergence of the random walk methods, with respect to several properties of interest. We find a burn-in period of 6K samples, which we exclude from each independent crawl. The remaining 75K x 28 sampled nodes is our main sample dataset; for a fair comparison we also exclude the same number of burn-in samples from all datasets. Second, in Section 4.4.2 we examine the quality of the estimation based on each sample. Finally, in Section 4.4.3, we summarize our our findings and provide recommendations for the use of sampling methods in practice.

4.4.1 Convergence analysis

There are several crucial parameters that affect the convergence of MCMC. In this section we first study these parameters by (i) applying formal convergence tests and (ii) using simple (yet insightful) visual inspection of the related traces and histograms.

How to count

Counting samples in BFS is trivial since nodes are visited at most once. However, in the random walks, nodes can be revisited and repetitions *must* be included in the sample in order to ensure the desired statistical properties. To get more understanding, let us consider MHRW as an example, and let us first have a look at the typical chain evolution. At every iteration MHRW may either remain at the current user, or move to one of its neighbors. A sample path from a simulation is: Node-level: . . . 1, 1, 1, 1, 17, 1, 3, 3, 3, 1, 1, 1, 1, 2, 1, 1, 1, 2, 3, 9, 1. . . , where each number represents the number of consecutive rounds the chain remained at a given node. For RW, the same user cannot be immediately visited twice, but non-consecutive repetitions are possible. In practice, that happens infrequently in the RW sample (as can be seen from the number of unique nodes given in table 5.1).

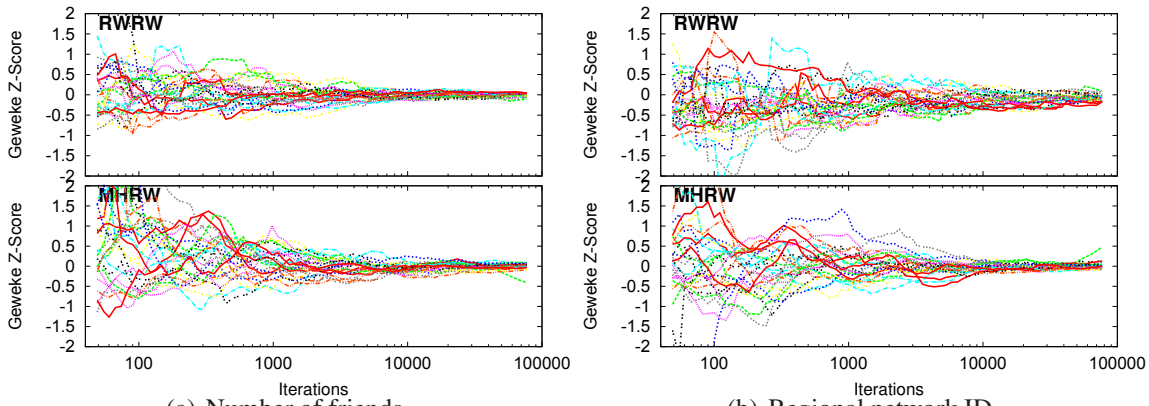
On the other hand, MHRW repeatedly samples some (typically low degree) nodes, a property which is intrinsic to its operation. For instance, if some node v_l has only one neighbor v_h, then the chain stays at (repeatedly samples) v_l for an average of k_{v_h} iterations (k_v is the degree of node v). Where k_{v_h} is large (e.g., $\mathcal{O}(10^2)$ or more), the number of repetitions may be locally large. While counter intuitive, this behavior is essential for convergence to the uniform distribution. In our MHRW sample, roughly 45% of the proposed moves are accepted (the *acceptance rate* in MCMC terms). As a result, a typical MHRW visits fewer unique nodes than a RW or BFS sequence of the same length. This raises the question: what is a fair way to compare the results of MHRW with RW and BFS? Since queries are

only made for new nodes, if $k_{v_l} = 1$ and MHRW stays at v_l for some $\ell > 1$ iterations when crawling an OSN, the bandwidth consumed is equal in cost to one iteration (assuming that we cached the visited neighbor of v_l). This suggests that an appropriate practical comparison should be based not on the total number of iterations, but rather on the number of visited unique nodes. In our subsequent comparisons, we will denote RW and MHRW indices as "RW-Fair" and "MHRW-Fair" when we compare using the number of visited unique nodes, as this represents the methods in terms of equivalent bandwidth costs.

Convergence

Burn-in. A decision we have to make is about the number of samples to discard to lose dependence from the initial seed point, for the random walk based methods. Since there is a cost for every user we sample, we would like to choose this value using formal convergence diagnostics so as not to waste resources. Here, we apply the convergence diagnostics presented in section 4.2.2 to several properties of the sampled nodes and choose as burn-in the maximum period from all tests.

The Geweke diagnostic is applied in each of the 28 walks separately and compares the difference between the first 10% and the last 50% of the walk samples. The intuition behind Geweke is that by allowing the walk to run long enough, we might reach different regions in regard to the metric we use. The Geweke diagnostic assumes an asymptotically standard degree distribution. Therefore if the walk has converged, 68% of the calculated z-scores should fall within 1 standard deviations of zero. Fig. 4.5 presents the results of the Geweke diagnostic for the user properties of node degree and regional network membership. We start at 50 iterations and plot 150 points logarithmically spaced. We observe that after approximately $500 - 2000$ iterations we have a z-score strictly between $[-1, 1]$. We can also see that RWRW and MHRW show similar results in regard to the Geweke diagnostic.



(a) Number of friends (b) Regional network ID

Figure 4.5: Geweke z score (0k..81k) for number of friends (left) and regional network affiliation (right). We declare convergence when all values fall in the $[-1, 1]$ interval. Each line shows the Geweke score for each of the 28 parallel walks.

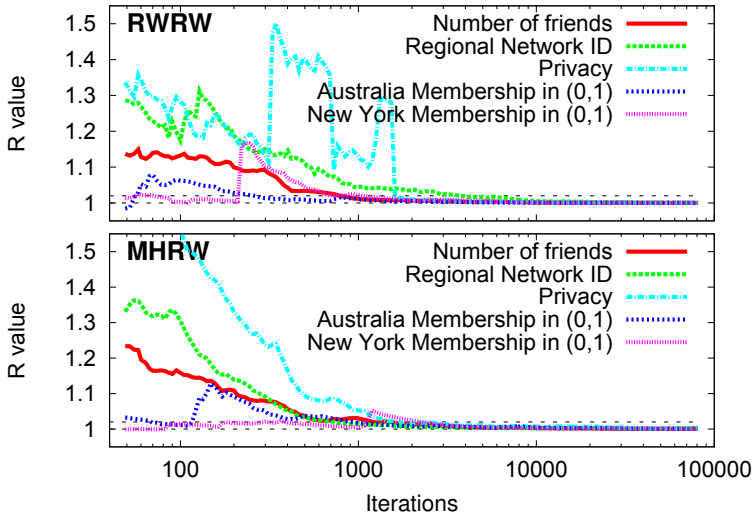


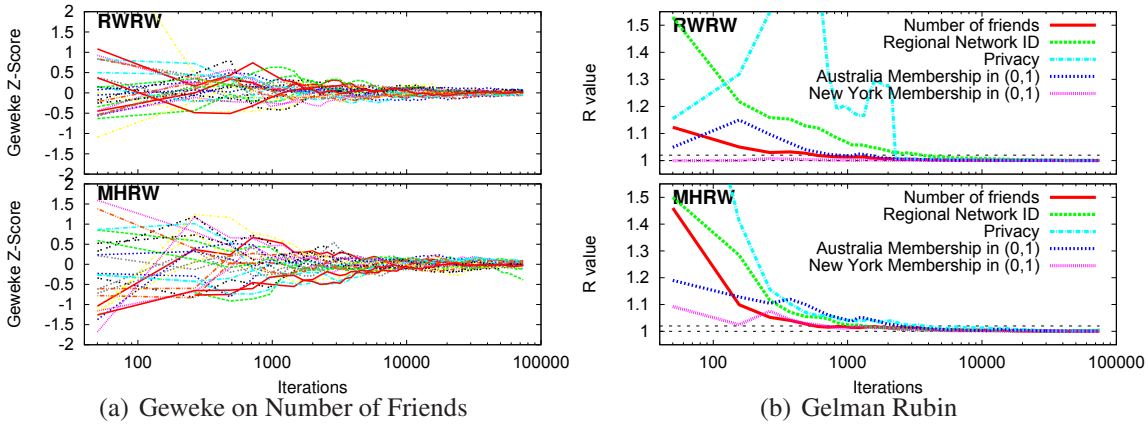
Figure 4.6: Gelman-Rubin R score (0k..81k) for five different metrics. Values below 1.02 are typically used to declare convergence.

The Gelman-Rubin diagnostic analyzes all the 28 walks at once by summarizing the difference of the between-walk variances and within-walk variances for all walks. In Fig 4.6 we plot the R score for the following metrics (i) number of friends (or node degree) (ii) networkID (or regional network) (iii) privacy settings Q_v (iv) membership in specific regional networks, namely Australia, New York and Colombia. The last user property is defined as follows: if the user in iteration i is a member of network x then the metric is set to 1, otherwise it is set to 0. We can see that the R score varies a lot in the initial hundred to

thousand iterations for all properties. To pick an example, we observe a spike between iterations $1,000$ and $2,000$ in the MHRW crawl for the New York membership. This is most likely the result of certain walks getting trapped within the New York network which is particularly large. Eventually, after 3000 iterations all the R scores for the properties of interest drop below 1.02, the typical target value used for convergence indicator.

We declare convergence when all tests have detected it. The Gelman-Rubin test is the last one at 3K nodes. To be even safer, in each independent walk we conservatively discard 6K nodes, out of 81K nodes total. In the remainder of the evaluation, we work only with the remaining 75K nodes per independent chain for RW, RWRW and MHRW.

Total Running Time and Thinning. Another decision we have to make is about the number of iterations for which we run the random walks, or the *walk length*, excluding the burn-in samples. This length should be appropriately long to ensure that we are at equilibrium. Here, we utilize multiple ways to analyze the collected samples, so as to increase our confidence that the collected samples are appropriate for further statistical analysis.



(a) Geweke on Number of Friends (b) Gelman Rubin

Figure 4.7: Online convergence diagnostics for samples 6k..81k (without burn-in). (left) Geweke z score for number of friends. (right) Gelman-Rubin score for five different metrics.

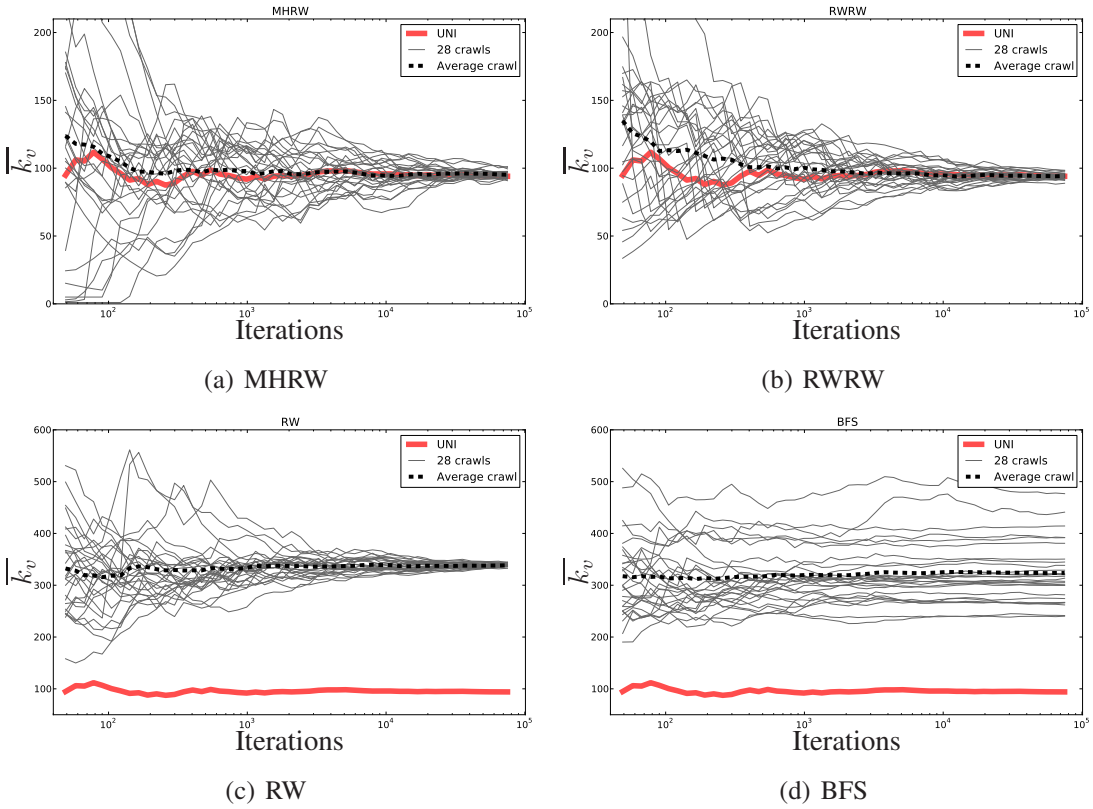
First, we apply formal convergence diagnostics that allow us to assess convergence online by indicating approximate equilibrium. Fig 4.7 shows the Geweke z-score for the number of friends (left) and the Gelman-Rubin R score for five different properties (right). These

results are obtained after discarding the burn-in samples (0k..6k). They show that convergence is attained with at least 3k samples per walk, similar to the section in which we determined the burn-in. This is an indication that the `Facebook` social graph is well connected and our random walks achieved good mixing with our initial selection of random seeds.

A visual way to check the convergence state of our sampling is to plot for each walk the running mean for a user property against the iteration number . The intuition here is that if convergence has been reached, the running mean of the property will not drastically change as the number of iterations increases. Fig 4.8 shows for each crawl type the running mean (i) for the node degree in the UNI sample, (ii) in each of the 28 walks individually, and (iii) in an average crawl that combines all 28 walks. It can be seen that in order to estimate the average node degree $\overline{k_v}$ based on only a single MHRW or RW walk, we should take at least 10k iterations to be likely to get within $\pm 10\%$ off the real value. In contrast, averaging over all 28 walks seems to provide similar or better confidence after fewer than 100 iterations per walk or $100 \times 28 \sim 3k$ samples over all walks. Additionally, the average MHRW and RW crawls reach stability within $350 \times 28 \sim 10k$ iterations. It is quite clear that the use of multiple parallel walks is very beneficial in the estimation of user properties of interest.

According to the formal diagnostics and the visual inspection, at minimum we need at least 3k samples per walk or $3k \times 28 \sim 84k$ over all walks. In our case, since we were not resource constrained during our crawling, we continued sampling users until we reached 81K per walk. One obvious reason is that more samples should decrease the estimation variance. Another reason is that more samples allow us to break the correlation between consecutive samples by thinning the set of sampled users. We use such a thinning process to collect egonets.

First, let us examine the effect of a larger sample on the estimation of user properties. Fig. 4.9 shows the percentage of sampled users with specific node degrees and network



(a) MHRW (b) RWRW

(c) RW (d) BFS

Figure 4.8: Average node degree $\overline{k_v}$ observed by each crawl, as a function of the number of iterations (or running mean).

affiliations, rather than the average over the entire distribution. A walk length of 75K (top) results in much smaller estimation variance per walk than taking 5K consecutive iterations from 50-55k (middle).

Fig.4.9 also reveals the correlation between consecutive samples, even after equilibrium has been reached. It is sometimes reasonable to break this correlation, by considering every ith sample, a process which is called *thinning*, as discussed at the end of Section 4.2.2. The bottom plots in Fig. 4.9 are created by taking 5K iterations per walk with a thinning factor of $i = 10$. It performs much better than the middle plot, despite the same total number of samples.

Thinning in MCMC samplings has the side advantage of saving space instead of storing all collected samples. In the case of crawling OSNs, the main bottleneck is the time and

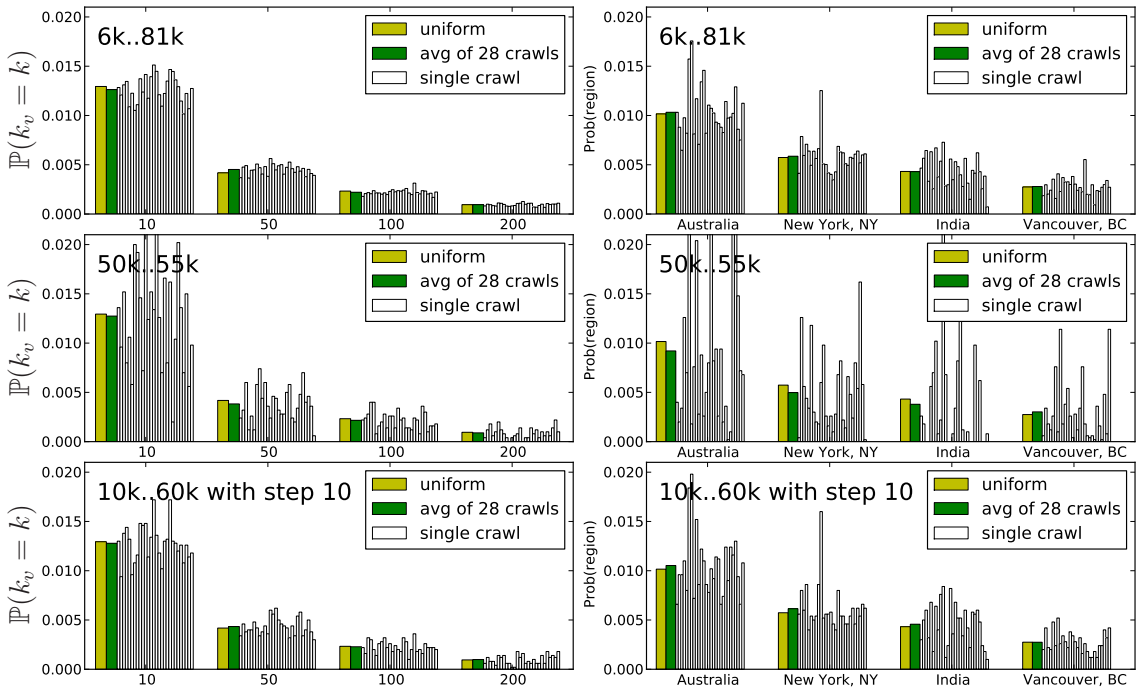
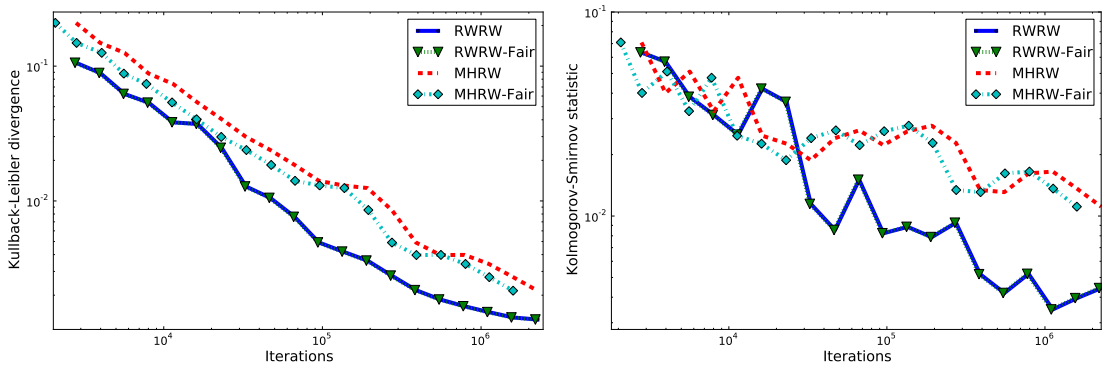


Figure 4.9: The effect of walk length and thinning on the results. We present histograms of visits at nodes with a specific degree $k \in \{10, 50, 100, 200\}$ and network membership (Australia, New York, India, Vancouver), generated under three conditions. (top): All nodes visited after the first 6k burn-in nodes. (middle): 5k consecutive nodes, from hop 50k to hop 55k. This represents a short walk length. (bottom): 5k nodes by taking every 10th sample (thinning).

bandwidth necessary to perform a single transition, rather than storage and post-processing of the extracted information. Therefore we did not apply thinning to our basic crawls.

However, we applied another idea (sub-sampling), that has a similar effect with thinning, when collecting the second part of our data - the egonets. Indeed, in order to collect the information on a single egonet, our crawler had to visit the user and all its friends, an average ~ 100 nodes. Due to bandwidth and time constraints, we could fetch only 37K egonets. In order to avoid correlations between consecutive egonets, we collected a random sub-sample of the MHRW (post burn-in) sample, which essentially introduced spacing among sub-sampled nodes.

Comparison to Ground Truth. Finally, we compare the random walk techniques in terms



(a) Kullback-Leibler divergence (b) Kolmogorov-Smirnov statistic

Figure 4.10: Efficiency of the random walk techniques (RWRW, MHRW) in estimating the degree distribution of `Facebook`, in terms of the Kullback-Leibler divergence and Kolmogorov-Smirnov statistic. We observe that (i) RWRW converges faster than MHRW and approximates UNI slightly better at the end (ii) RWRW-Fair is also more efficient than MHRW-Fair. The "Fair" versions of the algorithms count the real bandwidth cost of contacting a previously unseen neighbor, either for sampling (in RW) or to learn its degree (in MHRW), based on our measurements.

of their distance from the true uniform (UNI) distribution as a function of the iterations. In Fig. 4.10, we show the distance of the estimated distribution from the ground truth in terms of the KL (Kullback-Leibler) metric, that captures the distance of the 2 distributions accounting for the bulk of the distributions, and the Kolmogorov-Smirnov (KS) statistic, that captures the maximum vertical distance of two distributions. This comparison indicates that RWRW is more efficient than MHRW in regard to both statistics.

We should note here that the usage of distance metrics such as KL and KS cannot replace the role of the formal diagnostics which are able to determine convergence online and most importantly in the absence of the ground truth.

The choice of metric matters

MCMC is typically used to estimate some user property/metric, *i.e.,* a function of the underlying random variable. The choice of this metric can greatly affect the convergence time. The choice of metrics used in the diagnostics of the previous section was guided by

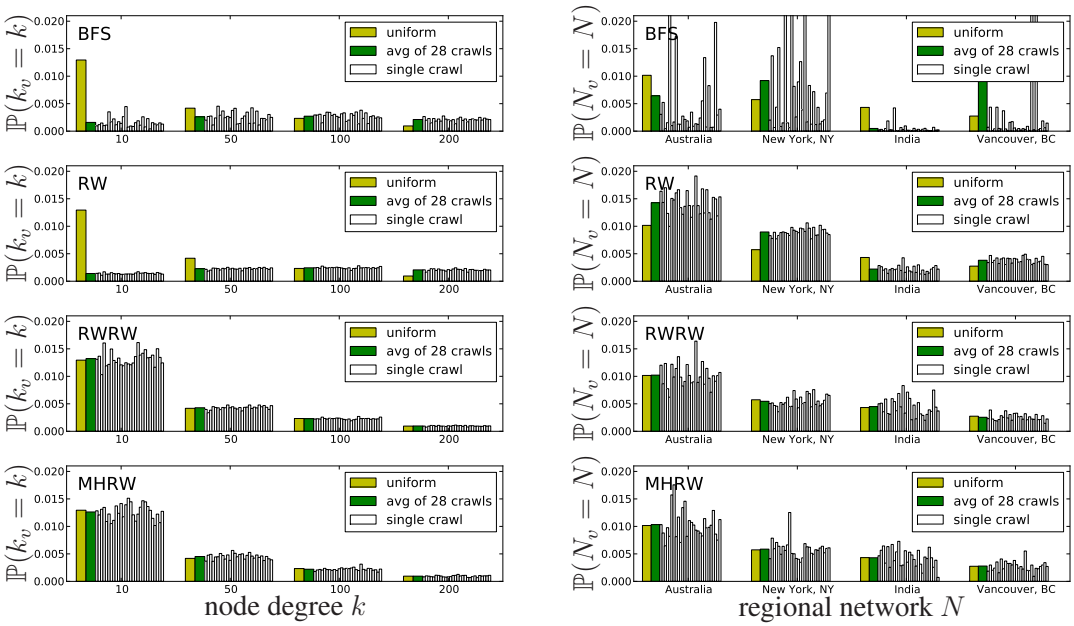


Figure 4.11: Histograms of visits at node of a specific degree (left) and in a specific regional network (right). We consider four sampling techniques: BFS, RW, RWRW and MHRW. We present how often a specific type of node is visited by the 28 crawlers ('crawls'), and by the uniform UNI sampler ('uniform'). We also plot the visit frequency averaged over all the 28 crawlers ('average crawl'). Finally, 'size' represents the real size of each regional network normalized by the the total `Facebook` size. We used all the 81K nodes visited by each crawl, except the first 6k burn-in nodes. The metrics of interest cover roughly the same number of nodes (about 0.1% to 1%), which allows for a fair comparison.

the following principles:

- We chose the node degree because it is one of the metrics we want to estimate; therefore we need to ensure that the MCMC has converged at least with respect to it. The distribution of the node degree is also typically heavy tailed, and thus is slow to converge.

- We also used several additional metrics (e.g. network ID, user ID and membership to specific networks), which are uncorrelated to the node degree and to each other, and thus provide additional assurance for convergence.

Let us focus on two of these metrics of interest, namely *node degree* and *sizes of geographical network* and study their convergence in more detail. The results for both metrics and all three methods are shown in Fig.4.11. We expected node degrees to not depend strongly on geography, while the relative size of geographical networks to strongly depend on geography. If our expectation is right, then (i) the degree distribution will converge fast to a good uniform sample even if the walk has poor mixing and stays in the same region for a long time; (ii) a walk that mixes poorly will take long time to barely reach the networks of interests, not to mention producing a reliable network size estimate. In the latter case, MHRW will need a large number of iterations before collecting a representative sample.

The results presented in Fig. 4.11 (bottom) indeed confirm our expectations. MHRW performs much better when estimating the probability of a node having a given degree, than the probability of a node belonging to a specific regional network. For example, one MHRW crawl overestimates the size of 'New York, NY' by roughly 100%. The probability that a perfect uniform sampling makes such an error (or larger) is $\sum_{i=i_0}^{\infty} \binom{i}{n} p^i (1-p)^i \simeq 4.3 \cdot 10^{-13}$, where we took $i_0 = 1k$, $n = 81K$ and $p = 0.006$. Even given such single-walk deviations, however, the multiple-walk average for the MHRW crawl provides an excellent estimate of the true population size.

4.4.2 Unbiased Estimation

This section presents the main results of this chapter. First, the MHRW and RWRW methods perform very well: they estimate two distributions of interest (namely node degree, regional network size) essentially identically to the UNI sampler. Second, the baseline algorithms (BFS and RW) deviate substantively from the truth and lead to misleading estimates.

Node degree distribution

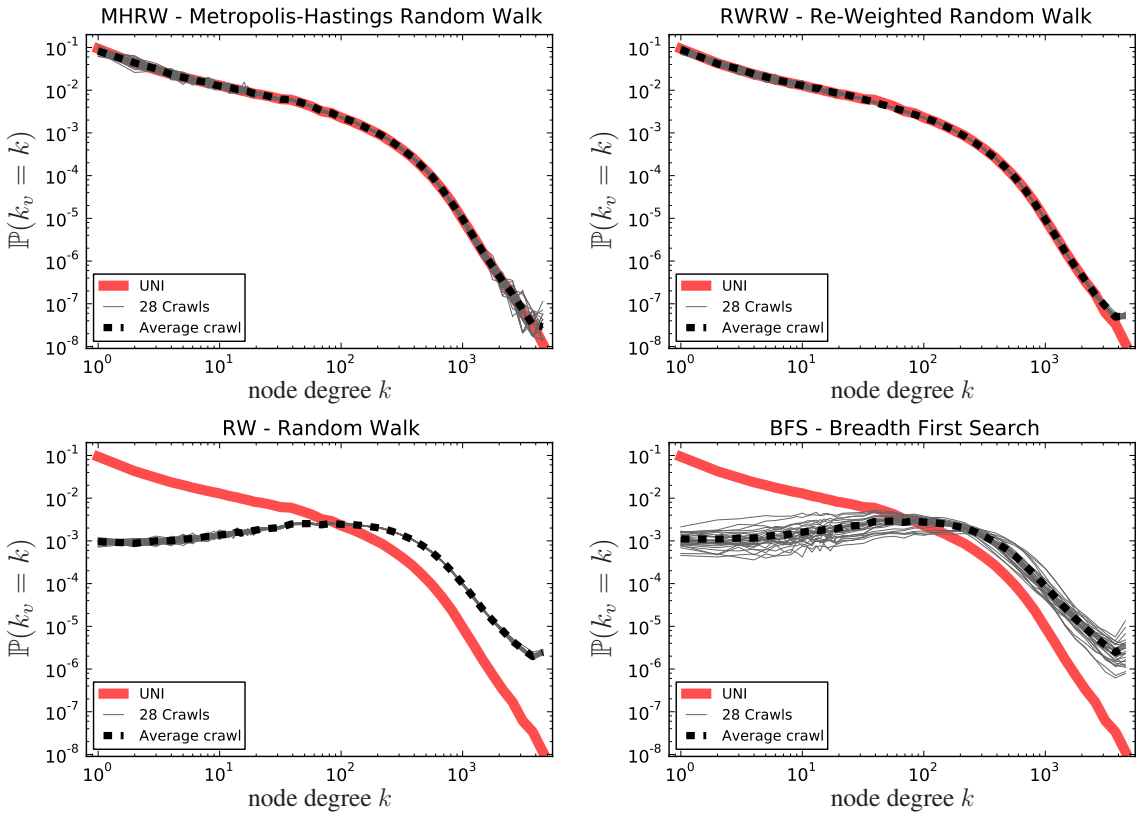


Figure 4.12: Degree distribution (pdf) estimated by the sampling techniques and the ground truth (uniform sampler). MHRW and RWRW agree almost perfectly with the UNI sample; while BFS and RW deviate significantly. We use log-log scale and logarithmic binning of data in all plots.

In Fig. 4.12 we present the degree distributions estimated by MHRW, RWRW, RW, and BFS. The average MHRW crawl's pdf, shown in Fig. 4.12(a) is virtually identical to UNI. Moreover, the degree distribution found by each of the 28 chains separately are almost identical. In contrast, RW and BFS shown in Fig. 4.12(c) and (d) introduce a strong bias towards the high degree nodes. For example, the low-degree nodes are under-represented by two orders of magnitude. As a result, the estimated average node degree is $\overline{k}_v \simeq 95$ for MHRW and UNI, and $\overline{k}_v \simeq 330$ for BFS and RW. Interestingly, this bias is almost the same in the case of BFS and RW, but BFS is characterized by a much higher variance. Notice that that BFS and RW estimate wrong not only the parameters but also the shape of

the degree distribution, thus leading to wrong information. Re-weighting the simple RW corrects for the bias and results to RWRW, which performs almost identical to UNI, as shown in 4.12(b). As a side observation we can also see that the true degree distribution clearly *does not* follow a power-law.

Regional networks

Let us now consider a geography-dependent sensitive metric, *i.e.,* the relative size of regional networks. The results are presented in Fig. 4.11 (right). BFS performs very poorly, which is expected due to its local coverage. RW also produces biased results, possibly because of a slight positive correlation that we observed between network size and average node degree. In contrast, MHRW and RWRW perform very well albeit with higher variance, as already discussed in Section 4.4.1.

The userID space

Finally, we look at the distribution of a property that is completely uncorrelated from the topology of `Facebook`, namely the user ID. When a new user joins `Facebook`, it is automatically assigned a 32-bit number, called userID. It happens before the user specifies its profile, joins networks or adds friends, and therefore one could expect no correlations between userID and these features. In other words, the degree bias of BFS and RW should not affect the usage of userID space. Therefore, at first sight we were very surprised to find big differences in the usage of userID space discovered by BFS, RW and MHRW. We present the results in Fig 4.13.

First, note that the userID space is not covered uniformly, probably for some historical reasons. BFS and RW are clearly shifted towards lower userIDs. The origin of this shift is probably historical. The sharp steps at $2^{29} \simeq 0.5\text{e}9$ and at $2^{30} \simeq 1.0\text{e}9$ suggest that `Facebook`

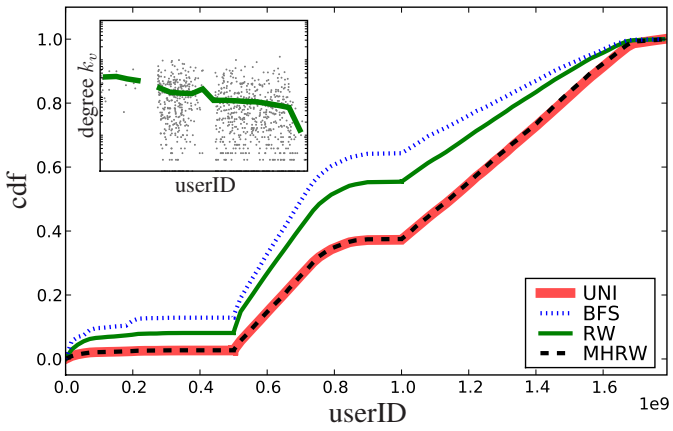


Figure 4.13: User ID space usage discovered by BFS, RW, MHRW and UNI. Each user is assigned a 32 bit long userID. Although this results in numbers up to $4.3e9$, the values above $1.8e9$ almost never occur. Inset: The average node degree (in log scale) as a function of userID.

was first using only 29 bit of userIDs, then 30, and now 31. As a result, users that joined earlier have the smaller userIDs. At the same time, older users should have higher degrees on average. If our reasoning is correct, userIDs should be negatively correlated with node degrees. This is indeed the case, as we show in the inset of Fig 4.13. This, together with the degree bias of BFS and RW, explains the shifts of userIDs distributions observed in the main plot in Fig 4.13. In contrast to BFS and RW, MHRW performed extremely well with respect to the userID metric.

Our observations are confirmed by internal sources within `Facebook` [82]. According to them, `Facebook`'s user ID assignment reflects the history of the website and has transitioned through several phases. Initially, userIDs were auto-incremented starting at 4. As more networks, such as colleges or high schools, were supported, customized userID spaces were assigned per college *i.e.,* Stanford IDs were assigned between 200000-299999. Finally, open registration to all users introduced scalability problems and made userID assignment less predictable.

4.4.3 Findings and Practical Recommendations

Choosing between methods. First and most important, the above comparison demonstrates that both MHRW and RWRW succeed in estimating several `Facebook` properties of interest virtually identically to UNI. In contrast, commonly used baseline methods (BFS and simple RW) fail, *i.e.,* deviate significantly from the truth and lead to substantively erroneous estimates. Moreover, the bias of BFS and RW shows up not only when estimating directly node degrees (which was expected), but also when we consider other metrics seemingly uncorrelated metrics (such as the size of regional network), which end up being correlated to node degree. This makes the case for moving from "1st generation" traversal methods such as BFS, which have been predominantly used in the measurements community so far [16, 21, 24], to more principled, "2nd generation", sampling techniques whose bias can be analyzed and/or corrected for. The random walks considered, RW, RWRW and MHRW, are well-known in the field of Monte Carlo Markov Chains (MCMC). We apply and adapt these methods to `Facebook`, for the first time, and we demonstrate that, when appropriately used, they perform remarkably well on real-world OSNs.

Adding convergence diagnostics and parallel crawls. A key ingredient of our implementation - not previously employed in network sampling - was the use of formal *online* convergence diagnostic tests. We tested these on several metrics of interest within and across chains, showing that convergence was obtained within a reasonable number of iterations. We believe that such tests can and should be used in field implementations of walk-based sampling methods to ensure that samples are adequate for subsequent analysis. Another key ingredient of our implementation, which we recommend, was the use of parallel crawlers/chains (started from several random independent starting points, unlike [35, 42] who use a single starting point), which both improved convergence and decreased the duration of the crawls.

MHRW vs. RWRW. Both MHRW and RWRW performed excellently in practice. When comparing the two, RWRW is slightly more efficient for the applications considered here, consistent with the findings in [35, 42]; this appears to be due to faster mixing in the latter Markov chain, which (unlike the former) does not require large numbers of rejections during the initial sampling process. However, when choosing between the two methods there are additional trade-offs to consider. First, MHRW yields an asymptotically uniform sample, which requires no additional processing for subsequent analysis. By contrast, RWRW samples are heavily biased towards high-degree nodes, and require use of appropriate re-weighting procedures to generate correct results. For the creation of large data sets intended for general distribution (as in the case of our `Facebook` sample), this "ready to use" aspect of MHRW has obvious merit; for example our released data sets are intended to be used by people that are not necessarily experts in the re-weighting methods, for whom the potential for erroneous misuse is high. A second advantage of MHRW is the ease of online testing for convergence to the desired target (uniform) distribution. In contrast, in RWRW, we test for convergence on a different distribution and then re-weight, which can introduce distortion. It is in principle possible to diagnose convergence on re-weighted statistics with RWRW. However, this requires appropriate use of re-weighting during the convergence evaluation process, which can increase the complexity of implementation. Finally, simple re-weighting is difficult or impossible to apply in the context of many purely data-analytic procedures such as multidimensional scaling or hierarchical clustering. Simulated Importance Resampling [83] provides a useful alternative for RWRW samples, but suffers from well-known problems of asymptotic bias (see [74] for a discussion and some palliatives). This is of less concern for applications such as moment estimation, for which re-weighting is both simple and effective.

Ultimately, the choice of RWRW versus MHRW is thus a trade-off between efficiency during the initial sampling process (which often favors RWRW) and simplicity/versatility of use for the resulting data set (which often favors MHRW). For our present purposes,

these trade-offs favor MHRW, and we employ it here for producing a uniform ready-to-use sample of users. However, both approaches are viable alternatives in many settings, thus we present and analyze both in this work.

4.5 Facebook Characterization

In the previous sections, we sampled Facebook and demonstrated convergence and a true uniform sample of $\sim$ 1M Facebook users. In this section, we use this unbiased sample and the extended egonets dataset to study some topological and non-topological features of Facebook. In contrast to previous works, which focused on some particular regions [18, 56] or used biased sampling methods [21, 24], our results are representative of the entire Facebook graph.

4.5.1 Topological characteristics

We first focus on purely topological aspects of the graph of Facebook.

Degree distributions

In Fig. 4.14, we present the true node degree distribution of Facebook, the pdf (left) and the corresponding CCDF (right). Interestingly, and unlike previous studies of crawled datasets in online social networks in [24, 21, 16], we conclude that the node degree distribution of Facebook *does not* follow a power law distribution. Instead, we can identify two regimes, roughly $1 \leq k < 300$ and $300 \leq k \leq 5000$, each following a power law with exponent $\alpha_{k<300} = 1.32$ and $\alpha_{k\geq 300} = 3.38$, respectively.[4] We note, however, that the

[4]Exponents were computed with the help of formula (5) in [84].

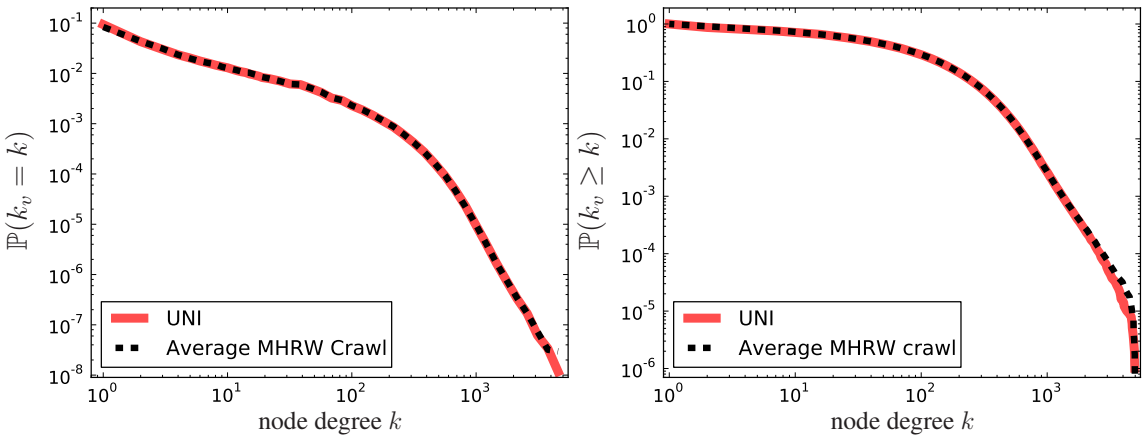


Figure 4.14: Degree distribution estimated by MHRW and the ground truth (UNI). (left) Probability Distribution Function (right) Complementary Cumulative Distribution Function. We use log-log scale and logarithmic binning of data in all plots.

regime $300 \leq k \leq 5000$ covers only slightly more than one decade.

Assortativity

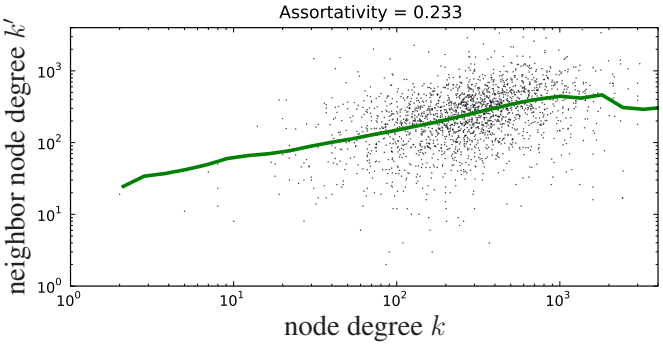


Figure 4.15: Assortativity - correlation between degrees of neighboring nodes. The dots represent the degree-degree pairs (randomly subsampled for visualization only). The line uses log-binning and takes the average degree of all nodes that fall in a corresponding bin.

Depending on the type of complex network, nodes may tend to connect to similar or different nodes. For example, in most social networks high degree nodes tend to connect to other high degree nodes [85]. Such networks are called *assortative*. In contrast, biological and technological networks are typically *disassortative*, *i.e.,* they exhibit significantly more high-degree than low-degree connections.

In Fig. 4.15 we plot the node degree vs. the degrees of its neighbors. We observe a positive correlation, which implies assortative mixing and is in agreement with previous studies of social networks. We can also summarize this plot by calculating the Pearson correlation coefficient, or *assortativity coefficient* r. The assortativity coefficient of `Facebook` is $r = 0.233$. This value is higher than $r' = 0.17$ reported by Wilson et al in [24]. A possible explanation of this difference is that [24] uses the Region-Constrained BFS to sample `Facebook`. It stops at a boundary of a regional network and thus misses many connections to, typically high-degree, nodes outside the network.

Clustering coefficient

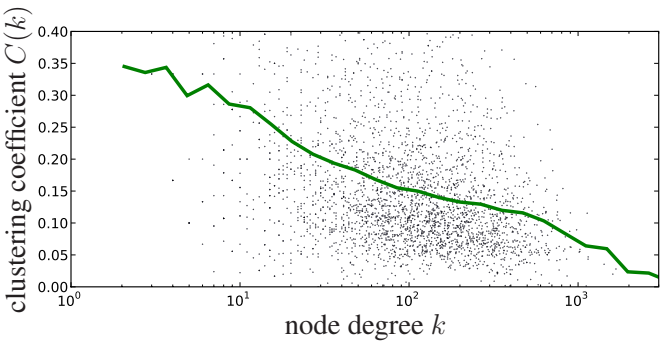


Figure 4.16: Clustering coefficient of `Facebook` users as function of their degree.

In social networks, it is likely that two friends of a user are also friends to each other. The intensity of this phenomenon can be formally captured by the *clustering coefficient* C_v of a node v, defined as the relative number of connections between the nearest neighbors of v. In other words, $C_v = \frac{2m_v}{k_v(k_v-1)}$, where m_v is the total number of edges between the nearest neighbors of v, and k_v is the degree of node v. The clustering coefficient of a network is just an average value $C = \frac{1}{n}\sum_v C_v$, where n is the number of nodes in the network. We find the average clustering coefficient of `Facebook` to be $C = 0.16$, similar to that reported in [24].

Since the clustering coefficient tends to depend strongly on the node's degree k_v, it makes

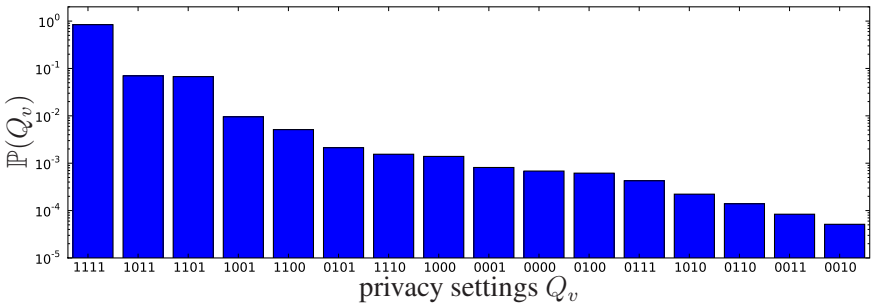


Figure 4.17: The distribution of the privacy settings among $\sim$ 172M `Facebook` users. Value $Q_v = 1111$ corresponds to default settings (privacy not restricted) and covers 84% of all users.

sense to study its average value $C(k)$ conditioned on k_v. We plot C_v as a function of k_v in Fig. 4.16. Comparing with [24], we find a larger range in the degree-dependent clustering coefficient ([0.05, 0.35] instead of [0.05, 0.18]).

4.5.2 Privacy awareness

PA	Network n	PA	Network n
0.08	Iceland	...	...
0.11	Denmark	0.22	Bangladesh
0.11	Provo, UT	0.23	Hamilton, ON
0.11	Ogden, UT	0.23	Calgary, AB
0.11	Slovakia	0.23	Iran
0.11	Plymouth	0.23	India
0.11	Eastern Idaho, ID	0.23	Egypt
0.11	Indonesia	0.24	United Arab Emirates
0.11	Western Colorado, CO	0.24	Palestine
0.11	Quebec City, QC	0.25	Vancouver, BC
0.11	Salt Lake City, UT	0.26	Lebanon
0.12	Northern Colorado, CO	0.27	Turkey
0.12	Lancaster, PA	0.27	Toronto, ON
0.12	Boise, ID	0.28	Kuwait
0.12	Portsmouth	0.29	Jordan
...	...	0.30	Saudi Arabia

Table 4.5: Regional networks with respect to their privacy awareness $PA = \mathbb{P}(Q_v \neq 1111 \mid v \in n)$ among $\sim$ 172M `Facebook` users. Only regions with at least 50K users are considered.

Recall from Section 4.3 that our crawls collected, among other properties, the privacy set-

tings Q_v for each node v. Q_v consists of four bits, each corresponding to one privacy attribute. By default, Facebook sets these attributes to 'allow', i.e., $Q_v = 1111$ for a new node v. Users can freely change these default settings of Q_v. We refer to the users that choose to do so as 'privacy aware' users, and we denote by PA the level of privacy awareness of a user v, *i.e.,* privacy aware users have $PA = \mathbb{P}(Q_v \neq 1111)$.

In Fig. 4.17, we present the distribution of privacy settings among Facebook users. About 84% of users leave the settings unchanged, *i.e.,* $\mathbb{P}(Q_v{=}1111) \simeq 0.84$. The remaining 16% of users modified the default settings, yielding $PA = 0.16$ across the entire Facebook. The two most popular modifications are $Q_v = 1011$ ('hide my photo') and $Q_v = 1101$ ('hide my friends'), each applied by about 7% of users.

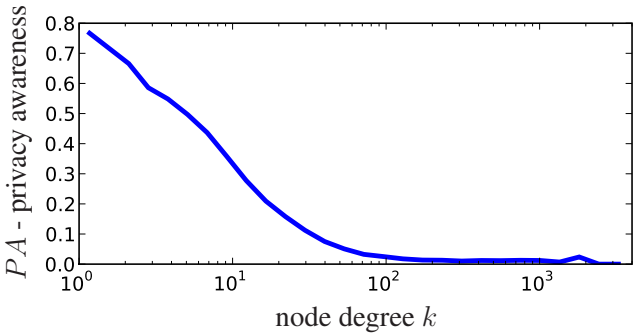


Figure 4.18: Privacy awareness as a function of node degree in the egonets dataset. We consider only the nodes with privacy settings set to '**1*', because only these nodes allow us to see their friends and thus degree. So here $PA = \mathbb{P}(Q_v{\neq}1111 \mid k_v = k,\ Q_v{=}**1*)$.

The privacy awareness PA of Facebook users depends on many factors, such as the geographical location, node degree or the privacy awareness of friends. For example, in Table 4.5 we classify the regional networks with respect to PA of their members. Note the different types of countries in the two extreme ends of the spectrum. In particular, many Facebook users in the Middle East seem to be highly concerned about privacy. Interestingly, Canada regions show up at both ends, clearly splitting into English and French speaking parts.

Another factor that affects the privacy settings of a user is the node degree. We present the

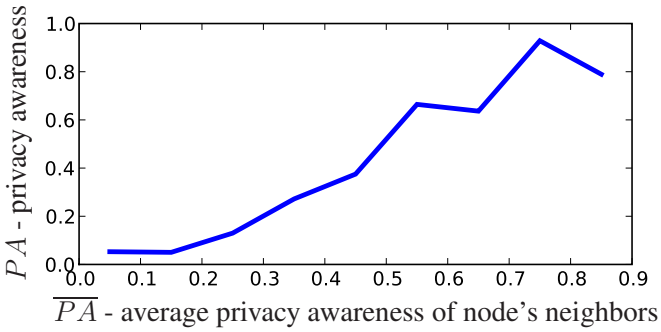


Figure 4.19: Privacy awareness as a function of privacy awareness of node's neighbors in the egonets dataset. We consider only the nodes with privacy settings set to '**1*', because only these nodes allow us to see their friends, so $PA = \mathbb{P}(Q_v{\neq}1111 \mid \overline{PA},\ Q_v{=}**1*)$.

results in Fig. 4.18. Low degree nodes tend to be very concerned about privacy, whereas high degree nodes hardly ever bother to modify the default settings. This clear trend makes sense in practice. Indeed, to protect her privacy, a privacy concerned user would carefully select her `Facebook` friends, *e.g.,* by avoiding linking to strangers. At the other extreme, there are users who prefer to have as many 'friends' as possible, which is much easier with unrestricted privacy attributes.

Finally, in Fig. 4.19 we show how privacy awareness of a user depends on the privacy awareness of her friends. We observe a clear positive correlation.

4.6 Summary

In this chapter, we obtained for the first time representative (*i.e.,* approximately uniform) samples of `Facebook` users. To perform this task, we implemented and compared several crawling methods. We demonstrated that two principled approaches (MHRW and RWRW) perform remarkably well (almost identical to the ground truth) while the more traditional methods (BFW, RW) lead to substantial bias. We also give practical recommendations about the use of these methods for sampling OSNs in practice, including online

convergence diagnostics and the proper use of multiple chains. The collected samples were validated against a true uniform sample, as well as via formal convergence diagnostics, and were shown to have good statistical properties. The datasets are accessible at [19]. Second, and based on our unbiased sample and on a sub-sample of egonets, we also studied some interesting properties of `Facebook`. Some of our findings agree with previous studies, some disagree and reveal a substantive bias of prior sampling techniques, and some are new to the best of our knowledge. .

Chapter 5

Multigraph sampling

5.1 Overview

Key to current sampling schemes is the fact that OSN users are, by definition, connected to one another via some relation, referred to here as the "social graph". Specifically, samples of OSN users can be obtained by crawling the OSN social graph. In the previous chapter we showed state-of-the-art techniques for probability sampling of users of online social networks (OSNs), based on random walks on a single social relation. Using such techniques we can achieve an asymptotic probability sample of users by online or a posteriori correction for the (known) bias induced by the crawling process. While random walk sampling can be very effective, its success is ultimately dependent on the connectivity of the underlying social graph. More specifically, random walks can yield a representative sample of users only if the social graph is fully connected. Furthermore, the speed with which the random walks converge to the target distribution strongly depends on characteristics of the graph, *e.g.,* clustering.

In this chapter, we propose *multigraph sampling*, a novel methodology for collecting an

asymptotic probability sample of OSN users by crawling not only on one, but on multiple graphs. Each individual graph connects the same set of users via a different set of edges, each capturing a different relation. These relations may include direct interpersonal ties, *e.g.,* friendship, but can also include indirect connections such as group co-membership, event co-participation, use of common applications, etc. We consider the union of these multiple graphs (which we refer to as "union multigraph", or simply "multigraph"), and we perform sampling via a random walk on this multigraph. Intuitively, multigraph sampling can potentially overcome two limitations of *single graph sampling*. First, the multigraph may be connected, even in cases when each individual graph is not. For example, multigraph sampling can discover users that have no friends, which would be impossible by crawling the friendship graph. Second, the multigraph may have better mixing properties, even when the individual graphs are highly clustered.

The naive way to approximate multigraph sampling with single graph sampling would be to run many random walks, one per each individual graph, and then combine the collected samples. However, in cases of fragmented graphs, each such walk would be restricted to the connected component around its starting point and thus would never converge. This would result in an arbitrary non-converged sample, dependent on the initial seed.

Although the idea of utilizing multiple relations to explore the social graph is conceptually simple and intuitively beneficial, it may be computationally difficult in practice if naively implemented. For example if we were to use the union graph instead of the union multigraph, we would have to enumerate all neighbors at every iteration. That would be costly in time and bandwidth. One of our contributions is that we design an efficient algorithm that walks on different graphs as the chain evolves, mixing across possible relations so as to avoid their individual limitations. We prove that the algorithm achieves convergence to the proper equilibrium distribution where the union multigraph is connected.

We demonstrate the benefits of our approach in two settings: (i) by simulation of synthetic

random graphs and (ii) by measurements of `Last.fm`- an Internet website for music with social networking features. We chose `Last.fm` as an example of a network that is highly fragmented with respect to the friendship graph as well as other relations. We show that multigraph sampling can obtain a representative sample when each individual graph is disconnected. Along the way, we also give practical guidelines on how to efficiently implement multigraph sampling for OSNs.

5.2 Sampling Methodology

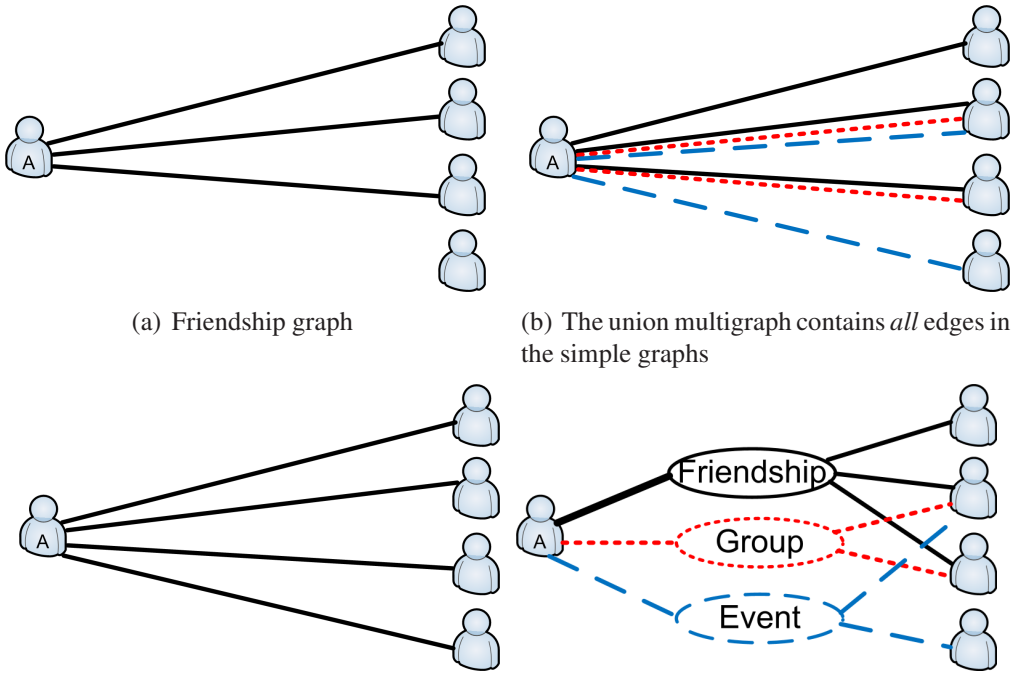
5.2.1 Terminology and Examples

We consider different sets of edges $\mathcal{E} = \{E_1, \ldots, E_Q\}$ on a common set of users V. Each E_i captures a symmetric relation between users, such as friendship or group co-membership. (V, E_i) defines an undirected graph G_i on V. We make no assumptions of connectivity or other special properties of each G_i. $\mathcal{E}$ can be used to construct several types of combined structures on V, of which we will employ the following two:

Definition 1. *Let the* union multigraph $G = (V, E)$ *of* $G_1, \ldots, G_Q$ *be the multigraph on* V *whose edges are given by the multiset* $E = \biguplus_{i=1}^{Q} E_i$. ■

Definition 2. *Let the* union graph $G' = (V, E')$ *of* $G_1, \ldots, G_Q$ *be the graph on* V *whose edges are given by the set* $E' = \cup_{i=1}^{Q} E_i$ *(*i.e., *s.t.* $\{u, v\} \in E'$ *iff* $\{u, v\} \in E$*).* ■

Note that the union multigraph G can contain multiple edges between a pair of nodes, while the union graph G' can be obtained from G by dropping any such redundant edges. Our focus here is on the union multigraph, also referred to as simply *multigraph*, because it allows us to implement more efficiently sampling on multiple properties; but we also use the union graph as a helpful conceptual tool.



(a) Friendship graph

(b) The union multigraph contains *all* edges in the simple graphs

(c) Union Graph

(d) An equivalent way of thinking the multigraph as "mixture" of simple graphs.

Figure 5.1: Example of different relations on a set of OSN users, the simple graphs they induce and their multigraph. (a) Each relation $i = 1, ..Q$ defines a different set of edges E_i, and thus a different simple graph $G_i = (V, E_i)$ on a common user set V. In this example, we present $Q = 3$ relations: Friendship (solid black edges, and shown separately in (a)), co-membership in a Group (red dotted edges), and co-participating in an Event (blue dashed edges). (b) The union multigraph, according to Definition 1 contains all the edges of all relations. (c) The union graph according to Definition 2 can be obtained from the multigraph by dropping redundant edges between a pair of nodes. (c) The multigraph can also be thought of as a "mixture" across the simple graphs. Node A has degrees $d_1(A)$=3, $d_2(A)$=2 and $d_3(A)$=2 in the Friendship, Group and Event graphs, respectively. Its total degree in the union multigraph is $d(A) = d_1(A) + d_2(A) + d_3(A) = 7$. Algorithm 2 moves from node A to the next sampled node as follows. First, it decides on which graph G_i to walk, with probability $\frac{d_i(A)}{d(A)}$. Next, it picks a random neighbor in the selected graph G_i, *i.e.,* with probability $\frac{1}{d_i(A)}$.

Example 1. *Fig 5.1 shows an example of different relations in the context of OSNs. Let us consider two users. If they are friends, then they are are connected through an edge in the friendship graph, depicted in Fig.5.1(a). If they are members of a common group or event, then they are connected through an edge on this particular group's graph. These edges appear as distinct edges in the multigraph G depicted in Fig.5.1(b). The union graph G'*

(not shown in the figure) can be obtained by dropping redundant edges between pairs of nodes. □

As noted above, a major motivation for considering multiple relations is connectivity. Consider the following example.

Example 2. *Fig. 5.2 shows five i.i.d. random (N, p) graphs with expected mean degree 1.5 on the same 50 nodes. One can immediately see that each individual graph is highly fragmented. However, their union graph, and thus the union multigraph, is fully connected.* □

5.2.2 Algorithm for Multigraph Sampling

We seek to draw a sample of the nodes in V, so that the draws are independent and that sampling probability of each node is known up to a constant of proportionality. There are several ways to achieve this goal using multiple graphs. One way is to form the union graph G' of $G_1, \ldots, G_Q$ and then sample using standard random walk methods on the union graph. Another way is to obtain the multigraph G of $G_1, \ldots, G_Q$ and employ a random walk that moves from one vertex to another by selection of random edges, as illustrated in Fig. 5.1(b). A potential practical difficulty with these approaches is the query cost when using high-degree, heavily clustered relations such as group co-membership, for which the calculation of neighborhood unions can be quite expensive. Indeed, assume that we are at a node and ready to choose the next sample. If we select an adjacent edge uniformly at random from all possible edges, then we must list all neighbors in each graph, which ends up being very costly (depending on the number of graphs in the multigraph). Instead we propose to take advantage of our knowledge of the size of each graph in the multigraph.

We alleviate this problem using the following two-stage procedure [1] , which is described

[1] In practice, this procedure can often increase efficiency, because degree information in OSNs is fre-

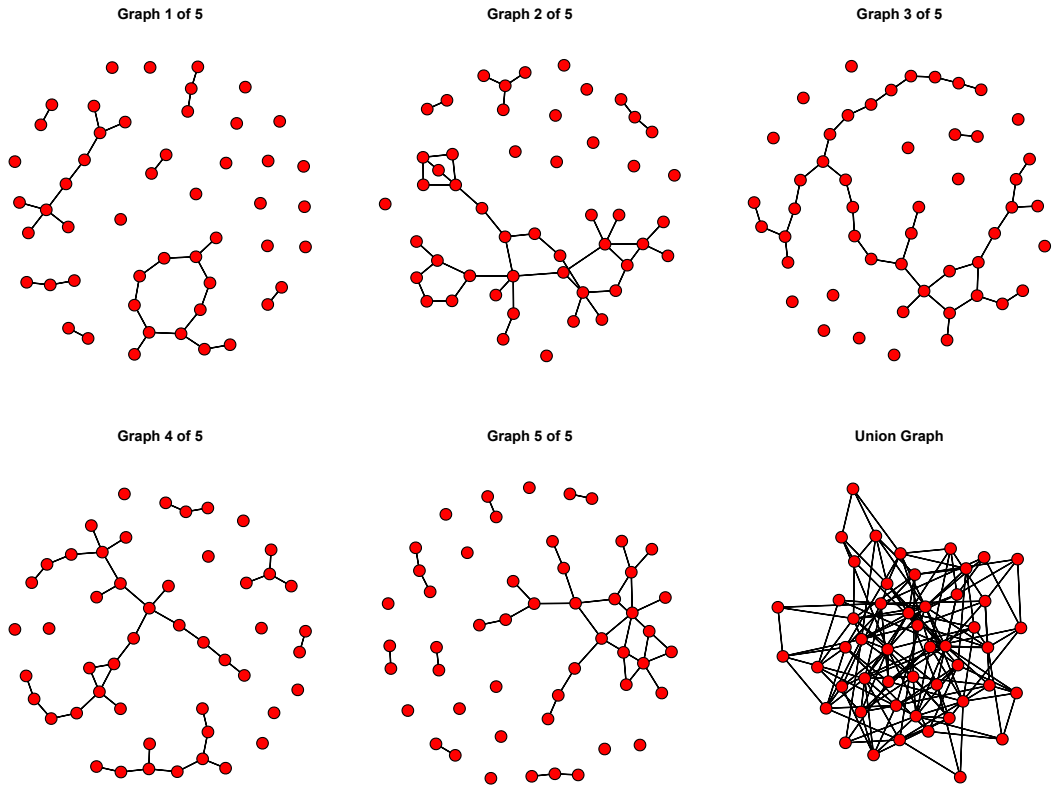


Figure 5.2: Example of multiple graphs vs. their union. Five draws from a random (N, p) graph with expected degree 1.5 are depicted in (a)-(e). Each simple graph is disconnected, while their union graph G', depicted in (f), is fully connected. The union multigraph G (not shown here) is also connected: it has - possibly multiple - edges between the exact same pairs of nodes as G'.

in Algorithm 2 and depicted in Fig.5.1(c). Denote by $d_i(v)$ the degree of node v in graph G_i, and by $d(v) = \sum_{i=1}^{Q} d_i(v)$ its degree in the union multigraph. First, we select a graph G_i with probability $\frac{d_i(v)}{d(v)}$. Second, we pick uniformly at random an edge of v within the selected G_i (*i.e.,* with prob. $\frac{1}{d_i(v)}$), and we follow this edge to v's neighbor. This procedure is equivalent to selecting an edge of v uniformly at random in the union multigraph, because $\frac{d_i(v)}{d(v)} \cdot \frac{1}{d_i(v)} = \frac{1}{d(v)}$.

Proposition 5.2.1. *If G is connected and contains at least one complete subgraph of order*

quently available without enumeration, *e.g.,* via API calls. This procedure may also be helpful in certain offline applications involving human respondents (*e.g.,* RDS [86]), in which selection of random neighbors is possible but enumeration is not.

Algorithm 2 Multigraph Sampling Algorithm

Require: $v_0 \in V$, simple graphs $G_i, i = 1 \dots Q$
  Initialize $v \leftarrow v_0$.
  while NOT CONVERGED **do**
    Select graph G_i with probability $\frac{d_i(v)}{d(v)}$
    Select uniformly at random a neighbor v' of v in G_i
    $v \leftarrow v'$
  end while
  return all sampled nodes v and their degrees $d(v)$.

3, then Algorithm 2 leads to equilibrium distribution $\pi(v) = \frac{d(v)}{\sum_{u \in V} d(u)}$.

Proof. Let $d(v, u) = d(u, v)$ be the number of edges between nodes v and u in the union multigraph G. The sampling process of Algorithm 2 is a Markov chain on V with transition probabilities $P_{vu} = \frac{d(v,u)}{d(v)}, u, v \in V$. So long as G is finite and connected, this random walk is irreducible and positive recurrent; the presence of a K_3 within G further guarantees aperiodicity. A Markov chain of this type is equivalent to a random walk on a finite undirected weighted graph with edge weights $w(v, u) = d(v, u)$. It is a standard result that this process has equilibrium distribution $\pi(v) = \frac{w(v)}{\sum_{u \in V} w(u)}$, where $w(v) = \sum_u w(v, u)$ (*e.g.,* see [87, Example 4.32]), and the proof follows immediately by substitution. □

5.2.3 Re-Weighting the Random Walk

Per Proposition 5.2.1, the random walk on the multigraph is inherently biased towards high degree nodes: the probability of sampling a node v is proportional to its degree $d(v)$ in the multigraph G. Therefore, the resulting sample requires re-weighting. We use the Hansen-Hurwitz estimator as shown in Section 4.2.1.

5.2.4 Multiple Walks and Convergence Diagnostics

In the previous chapter, we recommended the use of multiple, independent random walks to reduce the chance of obtaining samples that overweight non-representative regions of the graph. We also recommended the use of formal convergence diagnostics to assess sample quality in an online fashion. These diagnostics help to determine when a set of walks is in approximate equilibrium, and hence when it is safe to stop sampling. Use of both multiple walks and convergence diagnostics are critical to effective sampling of OSNs, as our sample case (Section 5.4) illustrates.

Here, we re-use the convergence diagnostics introduced in section 4.2.2. First, we track the running means for various scalar parameters of interest as a function of the number of iterations. Second, we use the Geweke [75] diagnostic within each random walk, which verifies that mean values for scalar parameters at the beginning of the walk (here the first 10% of samples) does not differ significantly from the corresponding mean at the end of the walk (here the last 50%). Third, we use the Gelman-Rubin [76] diagnostic to verify convergence across walks, by ensuring that the parameter variance between walks matches the variance within walks.

5.3 Evaluation in Synthetic Graphs

In this section, we use synthetic graphs to demonstrate two key benefits of the multigraph approach, namely (i) improved connectivity of the union multigraph, even when the underlying individual graphs are disconnected, and (ii) improved mixing time, even when the individual graphs are highly clustered. The former is necessary for the random walk to converge. The latter determines the speed of convergence.

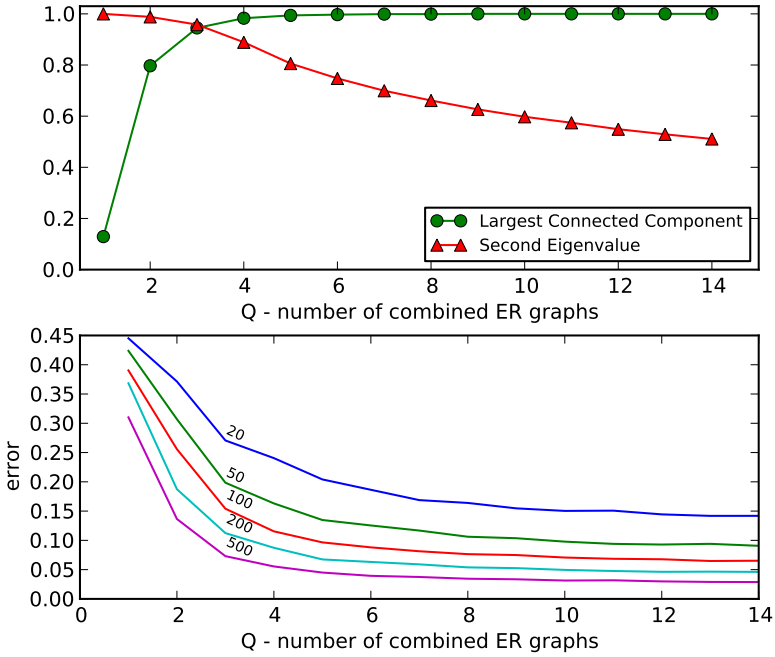


Figure 5.3: Multigraph that combines from several Erdos-Renyi graphs. We generate a collection $G_1, .., G_Q$ of Q random Erdos-Renyi (ER) graphs with $|V| = 1000$ nodes and expected $|E| = 500$ edges each. (top) We show two properties of multigraph G as a function of Q. (1) Largest Connected Component (LCC) fraction (f_{LCC}) is the fraction of nodes that belong to the largest connected component in G. (2) The second eigenvalue of the transition matrix of random walk on the LCC is related to the mixing time. (bottom) We also label a fraction $f = 0.5$ of nodes within LCC and run random walks of lengths $20 \ldots 500$ to estimate f. We show the estimation error (measured in the standard deviation) as a function of Q (x axis) and walk length (different curves).

Erdos-Renyi graphs. In Example 2 and Fig. 5.2, we noted that even sparse, highly fragmented graphs can have well-connected unions. In Fig. 5.3, we generalize this example and quantify the benefit of the multigraph approach. We consider here a collection $G_1, .., G_Q$ of Q Erdos-Renyi random graphs (N, p) with N=1000 nodes and p=$1/1000$, i.e., with the expected number of edges $|E| = 500$ each. We then look at properties of their multigraph G with increasing numbers of simple graphs Q.

In order to characterize the connectivity of G, we define f_{LCC} as the fraction of nodes that belong to the largest connected component in G. For Q=1 we have $f_{LCC} \simeq 0.15$, which means that each simple ER graph is heavily disconnected. Indeed, at least 999 edges are

necessary for connectivity. However, as Q increases, f_{LCC} increases. With a relatively small number of simple graphs, say for $Q = 6$, we get $f_{LCC} \simeq 1$, which means that the multigraph is almost fully connected. In other words, combining several simple graphs into a multigraph allows us to reach (and sample) many nodes otherwise unreachable.

Note that this example illustrates a more general phenomenon. Given Q independent random graphs with N nodes each and with expected densities $p_1, \ldots, p_Q$, the probability that an edge $\{u, v\}$ belongs to their union graph G' is $p^* = 1 - \prod_{i=1}^{Q}(1 - p_i)$. For p_i approximately equal, this approaches 1 exponentially fast in Q. Asymptotically, the union graph will be almost surely connected where $(N-1)p^* > \ln N$ [88, pp413–417], in which case the union multigraph is also trivially connected. Thus, intuitively, a relatively small number of sparse graphs are needed for the union to exceed its connectivity threshold.

In order to characterize the mixing time, we plot the second eigenvalue λ_2 of the transition matrix of the random walk (on the LCC). λ_2 is well-known to relate to the mixing time of the associated Markov chain [44]: the smaller λ_2, the faster the convergence. In Fig. 5.3(top), we observe that λ_2 significantly drops with growing Q. (However, that adding a new edge to an existing graph does not always guarantee the decrease of λ_2. It is possible to design examples where λ_2 increases.)

To further illustrate the connection between λ_2 and the speed of convergence, we conducted an experiment with a simple practical goal: apply random walk to estimate the size of a community in the network. We labeled as community members a fraction $f = 0.5$ of nodes within LCC (these nodes were selected with the help of a randomly initiated BFS to better imitate a community). Next, we ran 100 random walks of lengths $20 \ldots 500$ within this LCC, and we used them to estimate f. In Fig. 5.3 (bottom), we show the standard error of this estimator, as a function of Q (x axis) and walk length (different curves). This error decreases not only with the walk length, but also with the number Q of combined graphs. This means that by using the multigraph sampling approach we improve the quality of our

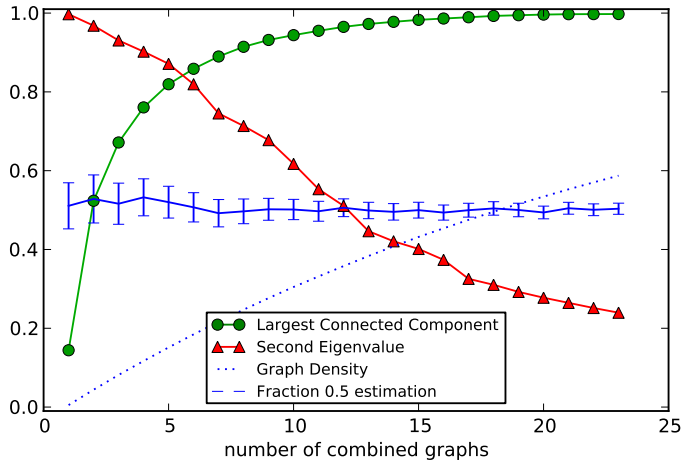


Figure 5.4: Multigraph resulting from combining one ER graph (with $|V| = 200$ nodes and $|E| = 100$ edges) with a set of $k - 1$ cliques (of size 40 randomly chosen nodes each).

estimates. Alternatively, we may think of it as a way to decrease the sampling cost. For example, in Fig. 5.3, a random walk of length 500 for Q=3 (*i.e.,* when LCC is already close to 1) is equivalent to a walk of length 100 for $Q \simeq 8$, which results in a five-fold reduction of the sampling cost.

ER Graphs Plus Cliques. One may argue that ER graphs are not good models for capturing real-life relations. Indeed, in practice, many relations are highly clustered; *e.g.,* a friend of my friend is likely to be my friend. In an extreme case, all members of some community may form a clique. This is quite common in OSNs, where we are often able to browse all members of a group, or all participants of an event.

Interestingly, the multigraph technique is efficient also under the presence of cliques. In 5.4, we consider one ER graph, combined with an increasing number of random cliques. We plot the same three metrics as in Fig 5.3, and we obtain qualitatively similar results. This robustness is a benefit of the multigraph approach.

5.4 Multigraph Sampling of Last.fm

In this section, we apply multigraph sampling to `Last.fm`- a music-oriented OSN that allows users to create communities of interest that include both listeners and artists. `Last.fm` is built around an Internet radio service that compiles a preference profile for each listener and recommends users with similar tastes. In June 2010, `Last.fm` was reported to have around 30 million users and was ranked in the top 400 websites in Alexa. We chose `Last.fm` to demonstrate our approach because it provides an example of a popular OSN that is fragmented with respect to the friendship graph as well as other relations. For example, many `Last.fm` users mainly listen to music and do not use the social networking features, which makes it difficult to reach them through crawling the friendship graph; likewise, users with similar music tastes may form clusters that are disconnected from other users with very similar music tastes. This intuition was confirmed by our empirical observations. Despite these challenges, we show that multigraph sampling is able to obtain a fairly representative sample in this case, while single graph sampling on any specific relation fails.

5.4.1 Crawling Last.fm

We sample `Last.fm` via random walks on several individual relations as well as on their union multigraph. Fig 5.5 shows the information collected for each sampled user.

Walking on Relations

We consider the following relations between two users:

- *Friends:* This refers to mutually declared friendship between two users.

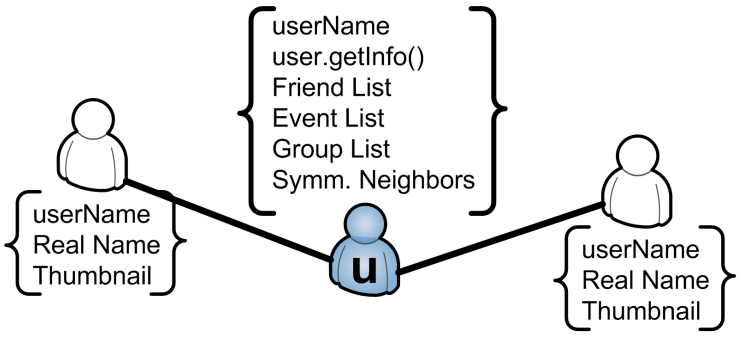


Figure 5.5: Information collected for a sampled user u. (a) userName and user.getInfo: Each user is uniquely identified by her userName. The API call user.getInfo returns : real Name, userID, country, age, gender, subscriber, playcount, number of playlists, bootstrap, thumbnail, and user registration time. (b) Friends list: List of mutually declared friendships. (c) Event list. List of past and future events that the user indicates she will attend. We store the eventID and number of attendees. (d) Group list. List of groups of which the user is a member. We store the group name and group size. (e) Symmetric neighbors. List of mutual neighbors.

- *Groups:* Users with something in common are allowed to start a group. Membership in the same group connects all involved users.

- *Events:* `Last.fm` allows users to post information on concerts or festivals. Attendees can declare their intention to participate. Attendance in the same event connects all involved users.

- *Neighbors:* `Last.fm` matches each user with up to 50 similar neighbors based on common activity, membership, and taste. The details of neighbor selection are proprietary. We symmetrize this directed relation by considering only mutual neighbors as adjacent.

More details about how we fetch data for each relation are provided in Appendix B.2.

First, we collect a sample of users by a random walk on the graph for each individual relation, that is Friends, Groups, Events, and Neighbors. Then, we consider sets of relations (namely: Friends-Events, Friends-Events-Groups, and Friends-Events-Groups-Neighbors) and we perform a random walk on the corresponding union multigraph. In the rest of the

Single relation crawl types

Dataset	Friends	Events	Groups	Neighbors
# Total Users	$5{\times}50K$	$5{\times}50K$	$5{\times}50K$	$5{\times}50K$
% Unique users	71.0%	58.5%	74.3%	53.1%
# Users kept	245K	245K	245K	245K
Crawling period	07/13-07/16	07/13-07/18	07/13-07/17	07/13-07/17
Avg # friends	10.7	18.0	15.8	12.2
Avg # groups	2.40	4.71	5.22	2.90
Avg # events (past/future)	2.44/0.17	7.49/0.56	3.96/0.28	2.94/0.27

Multiple relation crawl types and Uniform

Dataset	Friends-Events	Friends-Events-Groups	Friends-Events-Groups-Neighbors	UNI
# Total Users	$5{\times}50K$	$5{\times}50K$	$5{\times}50K$	500K
% Unique users	59.4%	75.5%	75.6%	99.1%
# Users kept	200K	187K	200K	500K
Crawling period	07/13-07/18	07/13-07/18	07/13-07/21	07/13-07/16
Avg # friends	9.8	6.8	6.6	1.2
Avg # groups	2.47	0.71	0.67	0.30
Avg # events (past/future)	2.30/0.17	0.74/0.05	0.73/0.04	0.28/0.02

Table 5.1: Summary of collected datasets in July 2010. The percentage of users kept is determined from convergence diagnostics. Averages shown are after re-weighting which corrects sampling bias.

section, we refer to random walks on different simple graphs or multigraphs as *crawl types*.

Uniform Sample of userIDs (UNI)

`Last.fm` usernames uniquely identify users in the API and HTML interface. However, internally, `Last.fm` associates each username with a userID, presumably used to store user information in the internal database. We discovered that it is possible to obtain usernames from their userIDs, a fact that allowed us to obtain a uniform, "ground truth" sample of the user population. Examination of registration and ID information indicates that `Last.fm` allocates userIDs in an increasing order. Fig. 5.6 shows the exact registration

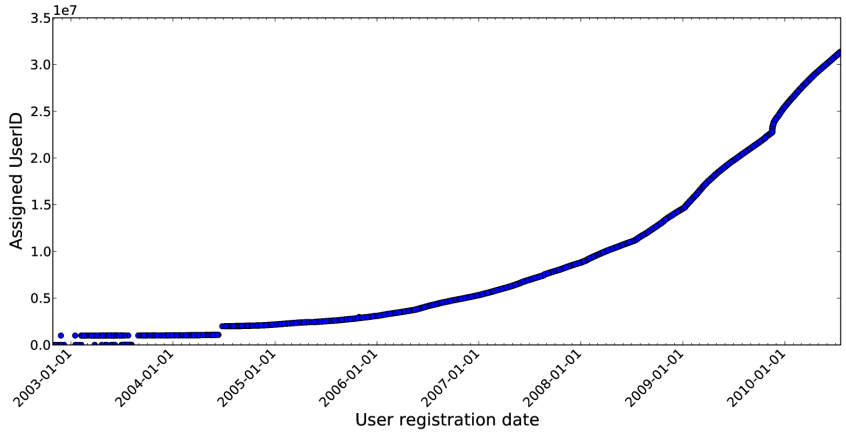


Figure 5.6: `Last.fm` assigns userIDs in increasing order after 2005: userID vs registration time (in seconds since 1970).

date and the assigned userID for each sampled user in our crawls obtained through exploration. With the exception of the first $\sim 2M$ users (registered in the first 2 years of the service), for every $userID_1 > userID_2$ we have $registration\_time(userID_1) > registration\_time(userID_2)$. We also believe that userIDs are assigned sequentially because we rarely observe non-existent userIDs after the $\sim 2,000,0000$ threshold. We conjecture that the few non-existent userIDs after this threshold are closed or banned accounts. At the beginning of the crawl, we found no indication of user accounts with IDs above $\sim 31,200,000$. Just before the crawls, we registered new users that were assigned user IDs slightly higher than the latter value.

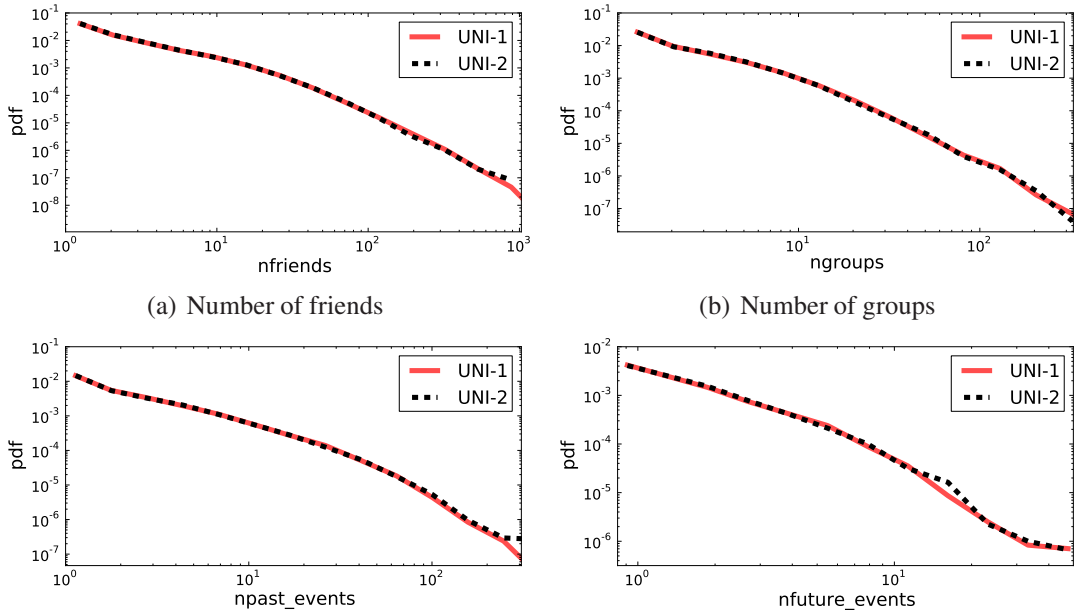
Using the userID mechanism, we obtained a reference sample of `Last.fm` users by uniform rejection sampling [77]. Specifically, each user was sampled by repeatedly drawing uniform integers between 0 and 35 million (*i.e.,* the maximum observed ID plus a $\sim$4 million "safety" range) and querying the userID space. Integers not corresponding to a valid userID were discarded, with the process being repeated until a match was obtained. IDs obtained in this way are uniformly sampled from the space of user accounts, irrespective of how IDs are actually allocated within the address space (see Appendix A). We employ this procedure to obtain a sample of 500K users, referred to here as "UNI."

Although UNI sampling currently solves the problem of uniform node sampling in `Last.fm` and is a valuable asset for this study, it is not a general solution for sampling OSNs. Such an operation is not generally supported by OSNs. Furthermore, the userID space must not be sparse for this operation to be efficient. In the `Last.fm` case, the small userID space makes this possible at the time of this writing; however, a simple increase of the userID space to 48 or 64 bits would render the technique infeasible. In summary, we were able to obtain a uniform sampling of userIDs and use it as a baseline for evaluating the sampling methods of interest against the target distribution. Our ability to do this capitalizes on unusual circumstances, however, while the walk-based sampling methods are broadly applicable to a wide range of OSNs.

Estimating Last.fm population size

In addition to the UNI sample presented in Table 5.1, we obtained a second UNI sample of the same size one week later. We then applied the capture-recapture method [89] to estimate the Last.fm user population during the period of our crawling. According to this method, the population size is estimated to be : $P_{\texttt{Last.fm}} = \frac{N_{UNI1} \times N_{UNI2}}{R}$

where $N_{UNI1} = N_{UNI2} = 500K$ and R is the number of valid common userIDs sampled during the first and second UNI samples. $P_{\texttt{Last.fm}}$ turned out to be 28.5 million, consistent with our observations of the maximum userID space and close to the perceived size of `Last.fm` on various Internet websites. We will use this estimation to comment on the topology change during our crawls and to estimate the pdf of the artist/track charts from the number of listeners we obtain through data scraping in Section 5.4.2.



(a) Number of friends

(b) Number of groups

(c) Number of past events

(d) Number of future events

Figure 5.7: Probability distribution function (PDF) of UNI samples obtained one week apart from each other. Results are binned.

Topology Change During Sampling Process

While substantial change could in theory affect the estimation process, `Last.fm` evolves very little during the duration of our crawls. The increasing-order userID assignment allows us to infer the maximum growth of `Last.fm` during this period, which we estimate at $25K$/day on average. Therefore, with a population increase of 0.09%/day, the user growth during our crawls (2-7 days) is calculated to range between $0.18\% - 0.63\%$ per crawl type, which is quite small. Furthermore, the comparison between the two UNI samples revealed almost identical distributions for the properties studied here, as shown in Fig 5.7.

For this study, then, we assume that any changes in the `Last.fm` network during the crawling period can be ignored. This is unlike the context of dynamic graphs, where considering the dynamics is essential, *e.g.,* see [42, 52, 53].

Implementation Details

We developed in `Python` a multi-threaded crawler, which is employed for all crawl types. The crawler uses a combination of API calls and HTML data scraping to traverse the multigraphs and has built-in mechanisms to obey API limits defined by `Last.fm`. HTML scraping in the crawler is required for two reasons. First, in some cases it is the only way to get information for some data that are inaccessible via the API but are visible in the website interface. Examples include the listing of group sizes, and the listing of group membership for users. Thus, scraping makes possible the inclusion of more multigraph properties in our sampling, *e.g.,* groups. Second, even if the data is available through the API, there are cases where scraping is the only way to efficiently implement multigraph sampling (Algorithm 2). Appendix B.2 provides more details about the way we crawled `Last.fm`.

We used a cluster of machines to execute all crawl types under comparison simultaneously. For each crawl type, we run $|V_0| = 5$ different independent walks. The starting points for the five walks, in each crawl type, are randomly selected users identified by the web site as listeners of songs from each of five different music genres: country, hip hop, jazz, pop and rock. This set was chosen to provide an overdispersed seed set, while not relying on any special-purpose methods (*e.g.,* UNI sampling). To ensure that differences in outcomes do not result from choice of seeds, the same seed users are used for all crawl types. We let each independent crawl continue until we determine convergence per walk and per crawl, using online diagnostics as introduced in [90] and described in Section 5.2.4. Eventually, we collected exactly 50K samples for each random walk crawl type. Finally, we collect a UNI sample of 500K users, a total sample at least as large as the number of samples in each crawl type.

Crawl Type	Friends Graph	Events Graphs	Groups Graphs	Neighbors Graphs
Friends	100%	0%	0%	0%
Events	0%	100%	0%	0%
Groups	0%	0%	100%	0%
Neighbors	0%	0%	0%	100%
Friends-Events	2.2%	97.8%	0%	0%
Friends-Events-Groups	0.3%	5.4%	94.3%	0%
Friends-Events-Groups-Neighbors	0.3%	5.5%	94.2%	0.02%

Table 5.2: Percentage of time a particular graph (edges corresponding to this graph) is used during the crawl by Algorithm2

Crawl Type	Friends Isolates	Future Events Isolates	Past Events Isolates	Groups Isolates
Friends	0%	93.7%	73.2%	60.4%
Events	19.2%	78.2%	4.5%	41.7%
Groups	21.2%	89.9%	62.0%	0.0%
Neighbors	40.4%	89.5%	71.2%	62.4%
Friends-Events	6.2%	93.5%	69.9%	61.6%
Friends-Events-Groups	5.5%	98.15%	88.1%	85.3%
Friends-Events-Groups-Neighbors	7.4%	98.3%	86.7%	86.3%
UNI	87.9%	99.2%	96.1%	93.8%

Table 5.3: Percentage of sampled nodes that are isolates (have degree 0) w.r.t. to a particular (multi)graph.

Summary of Collected Datasets

Table 5.1 summarizes the collected datasets. Each crawl type contains $5 \times 50K = 250K$ users. We observe that there is a large number of repetitions in the random walks of each crawl type, ranging from 25% (in Friends-Events-Groups-Neighbors) to 47% (in Neighbors). This appears to stem from the high levels of clustering observed in the individual networks. It is also interesting to note that the crawling on the multigraph Friends-Events-Groups-Neighbors is able to reach more unique nodes than any of the single graph crawls.

Table 5.3 shows the fraction of Markov chain transitions using each individual relation.

The results for the single-graph crawl types Friends, Events, Groups, and Neighbors are as expected: they use their own edges 100% of the time and other relations' 0%. Besides that, we see that Events relations dominate Friends when they are combined in a multigraph, and Groups dominate Friends, Events, and Neighbors when combined with them. This occurs because many groups and events are quite large (hundreds or thousands of users), leading participants to have very high relationship-specific degree for purposes of Algorithm 2, and thus for the Group or Event relations to be chosen more frequently than low-degree relations like Friends. In the crawl types obtained through a random walk, the highest overlap of users is observed between Groups and Friends-Events-Groups-Neighbors (66K) while the lowest is between Neighbors and Friends-Events (5K). It is noteworthy that despite the dominance of Groups and the high overlap between Groups and Friends-Events-Groups-Neighbors, the aggregates for these two crawl types in Table 5.1 lead to very different samples of users.

5.4.2 Evaluation Results

Convergence

Burn-in. To determine the burn-in for each crawl type in Table 5.1, we run the Geweke diagnostic separately on each of its 5 chains, and the Gelman-Rubin diagnostic across all 5 chains at once, for several different properties of interest. The Geweke diagnostic shows that first-order convergence is achieved within each walk after approximately 500 iterations at maximum. For the single relation crawl types (Friends, Events, Groups, and Neighbors), the Gelman-Rubin diagnostic indicates that convergence is attained within 1000 iterations per walk (target value for R below 1.02 and close to 1) as shown in Fig. 5.8.

On the other hand, multigraph crawl types take longer to reach equilibrium. Fig. 5.9 presents the Gelman-Rubin (R) score for three multigraph crawl types (namely Friends-

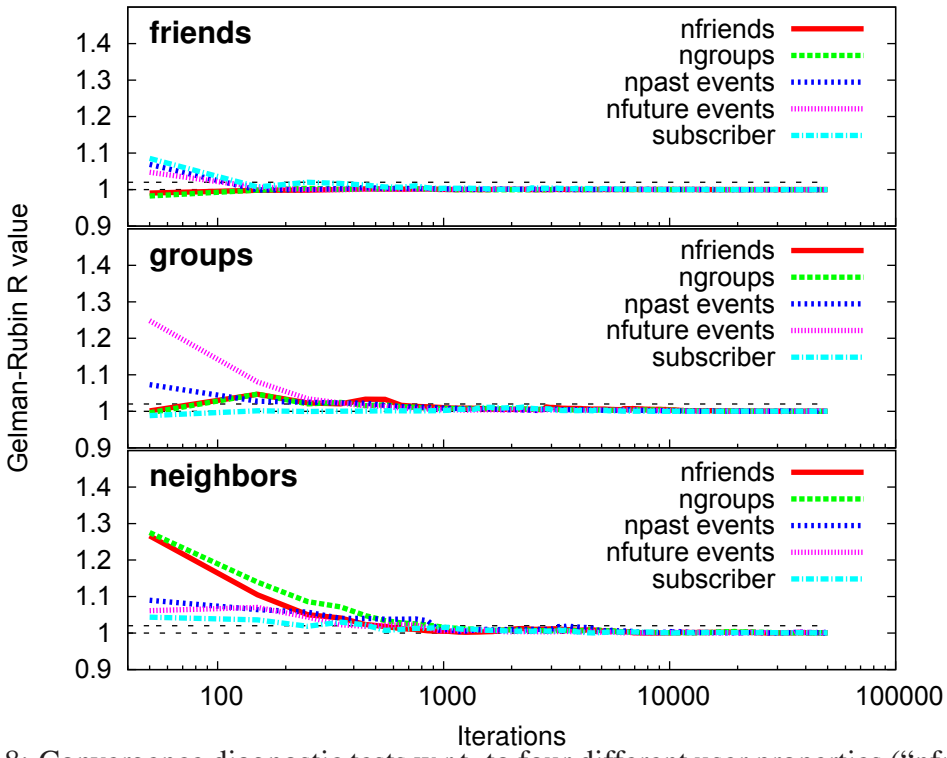


Figure 5.8: Convergence diagnostic tests w.r.t. to four different user properties ("nfriends": number of friends, "ngroups": number of groups, "npast_events": number of past events, "nfuture_events": number of future events, and "subscriber") and three different crawl types (Friends, Groups, Neighbors).

Events, Friends-Events-Groups, Friends-Events-Groups-Neighbors) and five user properties (namely number of friends, number of groups, number of past/future events, and subscriber - a binary value which indicates whether the user is a paid customer). We observe that it takes 10K, 12.5K and 10K samples for each crawl type correspondingly to converge. However, as we show next, they include more isolated users and better reflect the ground truth, while the single graph sampling methods fail to do so. This underscores an important point regarding convergence diagnostics: while useful for determining whether a random walk sample approximates its equilibrium distribution, they cannot reliably identify cases in which the equilibrium itself is biased (*e.g.,* due to non-connectivity). For the rest of the analysis, we discard the number of samples each crawl type needed to converge.

Total Running Time. Before we analyze the collected datasets, we verify that the remaining walk samples, after discarding burn-in, have reached stationary distribution. Table 5.1

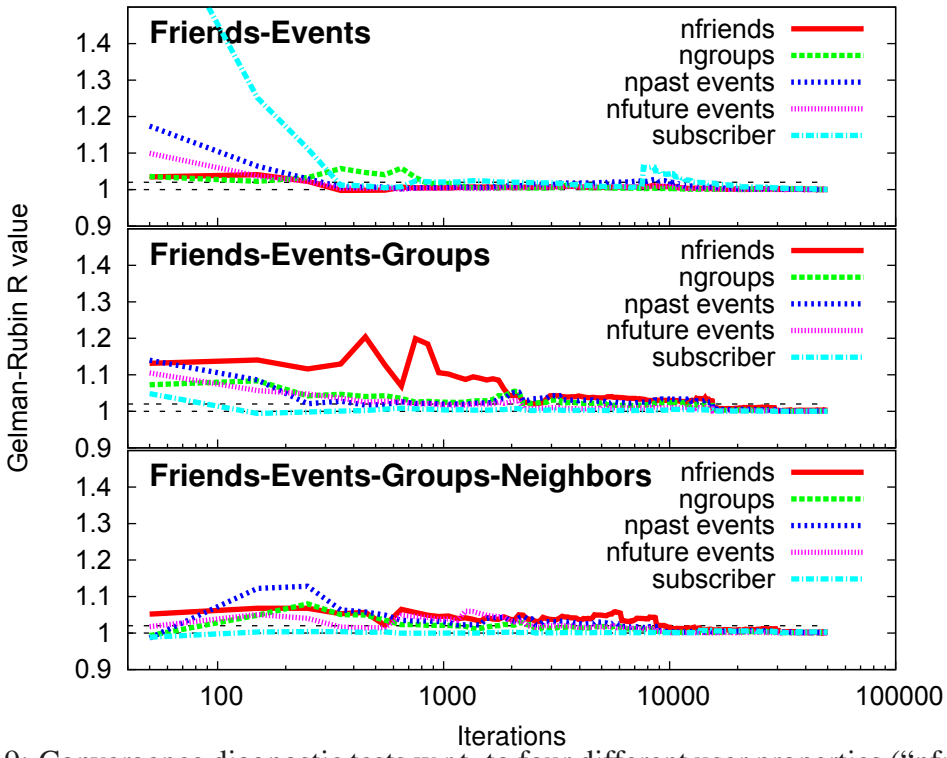


Figure 5.9: Convergence diagnostic tests w.r.t. to four different user properties ("nfriends": number of friends, "ngroups": number of groups, "npast_events": number of past events, "nfuture_events": number of future events, and "subscriber") and three different crawl types (Friends-Events, Friends-Events-Groups, Friends-Events-Groups-Neighbors).

contains the "Number of users kept" for each crawl type. We use the convergence diagnostics on the remaining samples to assess convergence formally. The results are qualitatively similar to the burn-in determination section. We also perform visual inspection of the running means in Fig 5.10 for four different properties, which reveals that the estimation of the average for each property stabilizes within 2-4k samples per walk (or 10k-20k over all 5 walks).

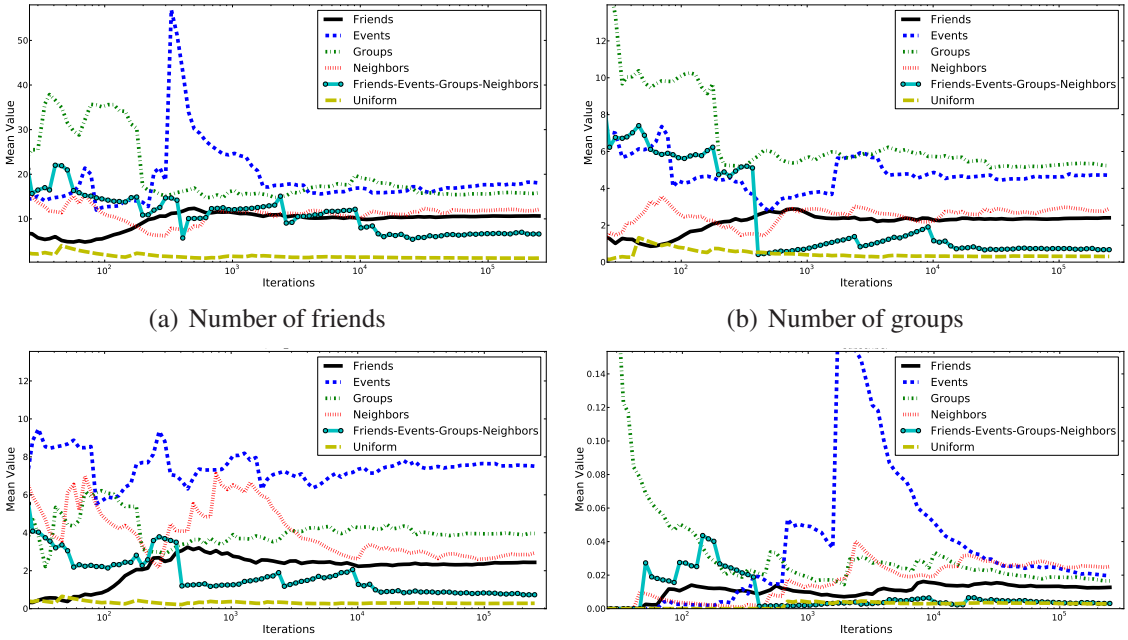
Discovering Isolated Components

As noted above, part of our motivation for sampling `Last.fm` using multigraph methods stems from its status as a fragmented network with a rich multigraph structure. In particular, we expected that large parts of the user base would not be reachable from the largest

connected component in any one graph. Such users could consist of either isolated individuals or highly clustered sets of users lacking ties to rest of the network. We here call *isolate*, any user that has degree 0 in a particular graph relation. Walk-based sampling on that particular graph relation has no way of reaching those isolates, but a combination of graphs might be able to reach them, assuming that a typical user participates in different ways in the network (*e.g.,* a user with no friends may still belong to a group or attend an event).

In the right-hand columns of Table 5.3, we report the percentage of nodes in each crawl type that are estimated to be isolates, and compare this percentage to the UNI sample. Observe that there is an extremely high percentage of isolate users in any single graph: *e.g.,* UNI samples are 88% isolates in the Friends relation, 96-99% isolates in the Events relation, and 93.8% isolates in the Groups relation. Such isolates are not necessarily inactive: for instance, 59% of users without friends have either a positive playcount or playlist value, which means that they have played music (or recorded their offline playlists) in `Last.fm`, and hence are or have been active users of the site. This confirms our expectation that `Last.fm` is indeed a fragmented graph.

More importantly, Table 5.3 allows us to assess how well different crawl types estimate the % of users that are isolates with respect to a particular relation or set of relations. We observe that the multigraph that includes all relations (Friends-Events-Groups-Neighbors) leads to the best estimate of the ground truth (UNI sample - shown in the last row). The only exception is the friends isolates, where the single graph Neighbors gives a better estimate of the percentage of isolates over all other crawl types. The multigraph crawl type Friends-Events-Groups-Neighbors uses the Neighbors relation only 0.02% of of the time, and thus does not benefit as much as might be expected (though see below). A weighted random walk that put more emphasis on this relation (or use of a relation that is less sparse) could potentially improve performance in this respect.



(a) Number of friends

(b) Number of groups

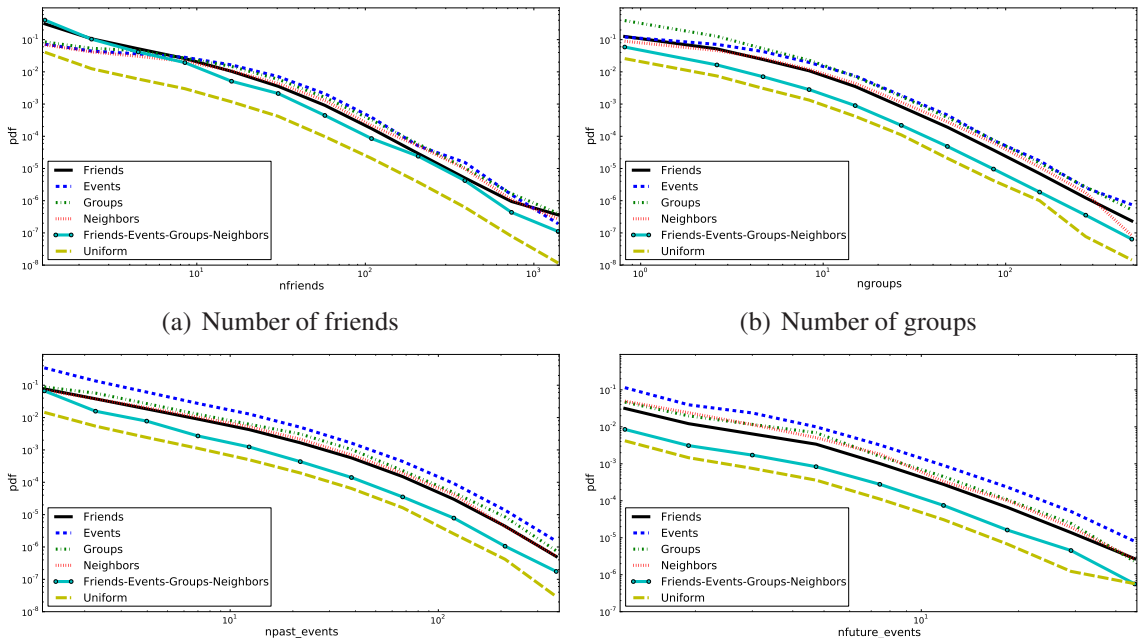
(c) Number of past events

(d) % of Subscribers

Figure 5.10: Single graph vs multigraph sampling. Sample mean over the number of iterations, for four user properties (number of friends, number of groups, number of past events, % subscribers), as estimated by different crawl types.

Comparing Samples to Ground Truth

I. Comparing to UNI. In Table 5.3, we saw that multigraph sampling was able to better approximate the percentage of isolates in the population. Here we consider other user properties (namely number of friends, past events, and groups a user belongs to, and whether he or she is a subscriber to `Last.fm`). In Fig 5.10, we plot the sample mean value for four user properties across iteration number, for all crawl types and for the ground truth (UNI). One can see that crawling on a single graph, *i.e.,* Friends, Events, Groups, or Neighbors alone, leads to poor estimates. This is prefigured by the previous results, as single graph crawls undersample individuals such as isolates on their corresponding relation, who form a large portion of the population. We also notice that Events and Groups alone consistently overestimate the averages, as these tend to cover the most active portion of the user base.



(a) Number of friends

(b) Number of groups

(c) Number of past events

(d) Number of future events

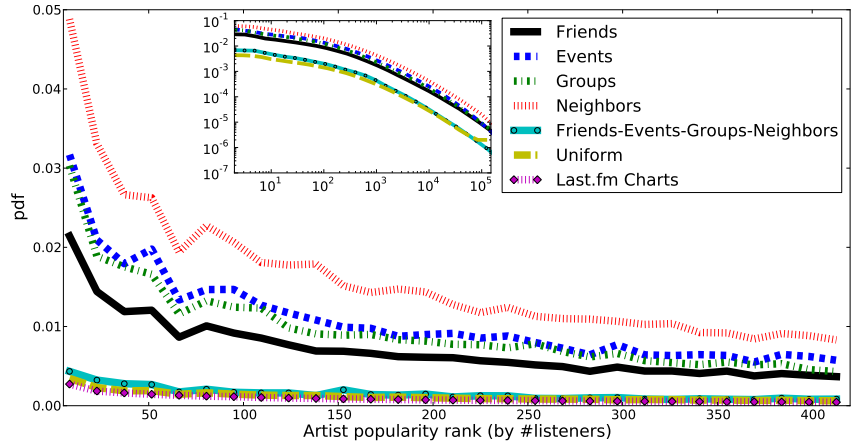
Figure 5.11: Single graph vs multigraph sampling. Probability distribution function (pdf) for three user properties (number of friends, number of groups, number of past events), as estimated by different crawl types.

However combining them together with other relations helps considerably. The multigraph that utilizes all relations, Friends-Events-Groups-Neighbors, is the closest to the truth. For example, it approximates very closely the avg number of groups and % of paid subscribers (Figs 5.10(b), 5.10(d)).

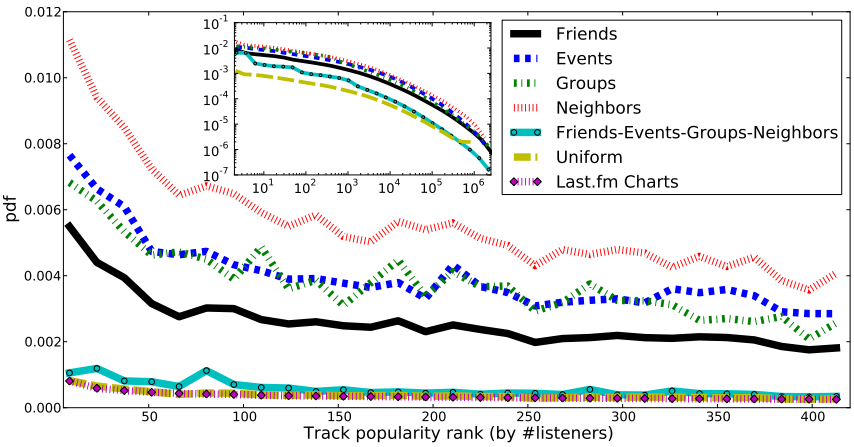
In Fig 5.11, we plot the probability distributions for four user properties of interest. Again, the crawl type Friends-Events-Groups-Neighbors is closest to the ground truth, in terms of shape of the distribution and vertical distance to it. Nevertheless, we observe that in both the probability distribution and the running mean plots, there is sometimes a gap from UNI, which is caused by the imperfect approximation of the % of isolates. That is the reason that the gap is the largest for the number of friends property (Fig 5.10(a), 5.11(a)).

II. Comparing to Weekly Charts. Finally, we compare the estimates obtained by different crawl types, to a different source of the ground truth - the weekly charts posted by

Last.fm. This is useful as an example of how one can (at least approximately) validate the representativeness of a random walk sample in the absence of a known uniform reference sample.



(a) Popularity of artists



(b) Popularity of tracks

Figure 5.12: Weekly Charts for the week 07/04-07/11. Artists/tracks are retrieved from "Last.fm Charts" and remain the same for all crawl types. Data is linearly binned (30 points). Inset: Artist/track popularity rank and percentage of listeners for all artists/tracks encountered in each crawl type.

Last.fm reports on its website weekly music charts and statistics, generated automatically from user activity. To mention a few examples, "Weekly Top Artists" and "Weekly Top Tracks" as well as "Top Tags", "Loved Tracks" are reported. Each chart is based on the actual number of people listening to the track, album or artist recorded either through an Audioscrobbler plug-in (a free tracking service provided by the site) or the Last.fm

radio stream. To validate the performance of multigraph sampling, we estimate the charts of "Weekly Top Artists" and "Weekly Top Tracks" from our sample of users for each of the crawl types in Table 5.1, and we compare it to the published charts for the week July 04-July 11 2010, *i.e.,* the week just before the crawling started. To generate the charts from our user samples, we utilize API functions that allow us to fetch the exact list of artists and tracks that a user listened during a given date range. Fig. 5.12 shows the observed artist/track popularity rank and the percentage of listeners for the top 420 tracks/artists (the maximum available) from the `Last.fm` Charts, with the estimated ranks and percentage of listeners for the same tracks/artists in each crawl type. As can be seen, the rank curve estimated from the multigraph Friends-Events-Groups-Neighbors tracks quite well the actual rank curve. Additionally, the curve that corresponds to the UNI sample is virtually lying on top of the "`Last.fm` Charts" line. On the other hand, the single graph crawl types Friends, Events, Groups, and Neighbors are quite far from actual charts. Here, as elsewhere, combining multiple relations gets us much closer to the truth than would reliance on a single graph.

5.5 Summary

In this chapter, we have introduced a family of methods for random walk sampling of OSNs using multiple underlying relations. *Multigraph sampling* generates probability samples in the same manner as conventional random walk methods, but is more robust to poor connectivity and clustering within individual relations. As we demonstrate using the `Last.fm` service, multigraph methods can give reasonable approximations to uniform sampling even where the overwhelming majority of users in each underlying relation are isolates, thus making single-graph methods fail. Our experiments with synthetic graphs also suggest that multigraph sampling can improve the coverage and the convergence time for partitioned or

highly clustered networks. Given these advantages, we believe multigraph sampling to be a useful addition to the growing suite of methods for sampling OSNs.

Remaining open questions in this approach include the selection of the relations to use when sampling so as to optimize the multigraph sampler performance. Intuitively, it seems reasonable to suppose that negatively correlated relations (i.e., those that tend to be mutually exclusive) will prove most effective. Gaining intuition into this problem would be particularly helpful in designing optimal OSN sampling schemes.

Chapter 6

Facebook applications

6.1 Overview

In a successful attempt to enhance user experience and increase the site's appeal, in May 2007, Facebook made a key innovation: they opened their platform to third-party developers [91]. Developers were given the opportunity to create Facebook applications that augment Facebook's functionality or act as front-end to third party web-based services. The Facebook application paradigm is unique because the risk of development and promotion investment in third party applications is smaller than the risk of investing in stand-alone web applications. This is due to the simplicity of the Facebook application API and to the inherent capabilities of the platform. In particular, Facebook notifies its members about the applications their friends install and use, and applications themselves prompt the user to invite friends to install them.

In mid-February 2008, there were approximately 866M installations of 16.7K distinct Facebook applications, and 200K developers are utilizing the platform. 138 applications had more than 1 million installations on February 25, 2008. Within a few months,

more than 100 OSN application development companies were founded and Facebook application-based advertising campaigns have been surprisingly successful. [92].

Motivated by this unprecedented success, we became interested in studying the popularity (both its distribution and change over time), and adoption dynamics (how applications are installed by users) of Facebook applications. An in-depth understanding of these characteristics is important both for engineering and marketing reasons. Understanding the popularity and adoption dynamics can assist advertisers and investors to form strategies for acquiring applications or purchasing advertising real-estate, so as to reach a large portion of the targeted user-base at a low cost. At the same time, determining which applications tend to attract more users, can help to better engineer the applications' user interface and features and to better provision the OSN system.

For this study, we collected and analyzed two data sets pertaining to Facebook applications. The first data set consists of data obtained from Adonomics [93], a service based on statistics reported by Facebook [94], for a period of 6 months from Sept. 2007 until Feb. 2008. It provides the number of installations for each application and the number of distinct users that engage each application at least once during a day, called *Daily Active Users (DAU)*. The second data set consists of a sample of publicly available Facebook user profiles, crawled for 1 week in Feb. 2008, pertaining to approximately 300K users and 13.6K applications. Based on the above data, we are interested in the following questions.

Aggregate Application Popularity. We first ask whether the continuous growth of Facebook applications' install base translates to increasing user engagement. We find that although the total number of installations increased linearly over the entire 6-month period, the total number of daily active users increased in the first three months but subsequently dropped and eventually stabilized.

Popularity of Individual Applications. A natural question is how skewed is the distribution

of popularity across different applications. We examine the popularity of applications in terms of DAU and number of installations and we find that the distributions of both metrics are highly skewed. The next question is whether popularity depends on the type of application. To this end, we classify `Facebook` applications based on their functionality and identify the most popular categories and applications in terms of DAU.

User Coverage. The popularity of applications in itself does not tell us much about how applications are distributed among users. More detailed information is needed if user coverage is of interest, i.e. given a set of applications how many unique users have installed one or more applications from that set. For example, an advertiser may attempt to increase the reach of a campaign by acquiring two popular applications. However, this reach is diminished if there is significant overlap in the set of users that have installed these applications. To this end we use the crawled data, which essentially represent a bipartite graph of users and the applications they have installed. We derive statistics about this graph and user coverage thereof. We simulate and validate a "preferential installation" process, according to which the probability of a user installing a new application is proportional to a power of the number of applications she has already installed. The simulation takes as input the applications, their popularity, and the number of users. It outputs which applications each user has installed.

6.2 Data Sets and Collection

The results of this chapter are based on two data sets pertaining to third-party applications, which are summarized in Table 6.1. Because we are interested in third-party applications, we exclude in-house `Facebook` applications, such as "Groups", "Gifts", "Photos" etc. In the rest of this section, we describe the collection process we used for obtaining these data sets. But first let us give some brief background on how the `Facebook` platform works.

Data Set	Source	Period	Data Element
I	Adonomics (Facebook analytics)	08/29/07-02/14/08	(date, application, # installations, # active users)
II	Facebook Public User Profiles	02/20/08-02/27/08	(user, list of applications)

Table 6.1: Data Sets under Study

6.2.1 Background on the Facebook Platform

The Facebook Platform is a standards-based web service with methods for accessing and contributing Facebook data [91]. It comprises the following parts: (i) the Facebook API, which uses a REST-based interface. Facebook method calls are made over the internet by sending HTTP GET or POST requests to a REST server. The API enables developers to add social context to their application by utilizing data regarding the user's profile, its friends, its pictures, and events; (ii) the Facebook Query Language (FQL), which resembles an SQL interface to access the same data that one can access through the Facebook API; iii) Markup (FBML) and Java Script (FBJS), which allow developers to build applications that integrate into a user's Facebook experience through the Profile, Profile Actions, Canvas, News Feed and Mini-Feed.

6.2.2 Data Set I: Crawling Facebook Analytics

Data Set I consists of the daily number of installations and daily active users (DAU) for every application, for every day of a 170 day period. It was obtained by crawling the Adonomics [93] data sets. The rationale is as follows.

Collection Process. Facebook reports application statistics in its application directory [94]. It employs an application ranking system that is based on user engagement. From

12:00am to 11:59pm each day, they measure how many distinct users *engaged* the application at least once, i.e. performed one of the following actions: a) view its Canvas; b) clicked on FBML links; c) performed an AJAX form submission; and d) activated a click-to-play Flash. Facebook expresses DAU as an integer percentage of the total number of installations,which can be used to extrapolate the total number of installations of an application.

Facebook does not report historical data on DAU and installations, only statistics for the current day. On the other hand, Adonomics [93], continuously processes the above statistics page to provide Facebook daily statistics and analysis over long periods of time.

In order to determine the reliability of the statistics reported by Adonomics, we randomly sampled their statistics on DAU and cross-referenced it with what Facebook reported in its own application directory during February 2008. We confirmed that their application statistics were the same as in the original Facebook reports. Therefore, to allow for a rapid analysis over a longer period of time, we decided to scrape Adonomics instead of the raw Facebook Application Directory.

6.2.3 Data Set II: Crawling User Profiles

We also crawled a sample of publicly available Facebook user profiles for 1 week in Feb. 2008 and logged the applications that each user had installed. This constitutes our *Data Set II*.

Collection Process. Crawling user profiles is a more difficult task than crawling user friend lists (see data collection process in Chapter 4). That is because, in addition to Facebook defenses against automated data mining, users have a limited view of other profiles outside their (one hop) immediate circle of friends. As a result, we were severely constrained in regard to the data we could collect.

To partly work around these constraints and efficiently obtain user-specific information of interest *i.e.,* application installations per user, we exploited the fact that Facebook's privacy model revolves around Facebook networks, which were described in section 4.3.1. To begin with, a user that is a member of a network can browse the profiles of all other users in the same network, under default privacy settings. Another network-centric feature at the time, was the ability to "display 10 random people from a joined network". Fortunately for us, in the period during our data collection, a Facebook user could freely join any regional network, one at a time, with the only limitation being the frequency of network changes.

Therefore to proceed with the collection of user profiles, we created 20 new Facebook user accounts and we had these users join a regional network from the following list: Orange County, New York, Toronto, London, France, Greece, Mexico, India and Australia. By repeatedly requesting 10 random users from each user account in a specific network, we were able to mine the profiles of 4K-7K distinct users at each network after approximately 2K-4K requests for random people. Furthermore, we crawled the profiles of the friends of those distinct users that belong in the same network, acquiring 20K to 60K additional users at each network.

We note that sampling random user profiles from a Facebook network by randomly generated userIDs is not a plausible technique. That is because, as we observed, userIDs for users affiliated with a network are not necessarily in a given range of userID space.

We automated all the above procedures using Python scripts, which we made available online [95]. We note that all our 20 accounts were eventually banned by Facebook due to excessive activity. In total, we crawled approximately 300K users that had installed 13.6K distinct applications in total.

Limitations. Our crawling technique has some limitations that stem mainly from the rela-

tively restrictive Facebook data access policies. First, our sampling methodology missed profiles of users that restricted access by other users in the same Facebook network. Second, we did not capture privacy-conscious users that choose not to place any or all applications in their profile.

6.2.4 Representativeness of Data Sets

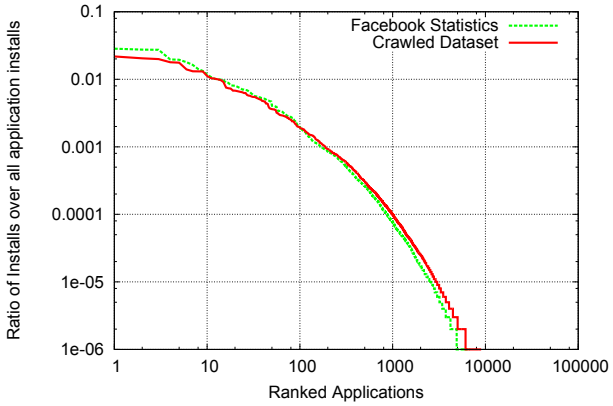


Figure 6.1: Probability Distribution Function (PDF) of total installations per application in Data Set I (Facebook statistics) and II (Crawled dataset).

Since dataset I is a full dataset with daily information for all existing Facebook applications, representativeness is not an issue. On the other, it is unclear whether Dataset II, which contains a sample of user profiles, is representative of the whole Facebook population. This is due to the fact that the inner-workings of the "10 random users" feature are unknown. In addition, by sampling the friends of the randomly returned users, it is possible to skew the distribution of application installations; e.g. some applications may be more popular among a group of friends than in the entire Facebook.

Clarifying the issue of the sample representativeness for all user properties found in a profile would be tricky. In our case, we only plan to use Dataset II to answer questions regarding the user coverage of applications in Facebook (in Section 6.4). Therefore determining that our sample is representative in regard to user application installations would

be adequate.

We have three indications that this is indeed the case. First, the distribution of application installations follows the same distribution as in the complete network, as shown in Fig. 6.1. Second, Data Set II has ∼13.6K distinct applications, which matches the applications with $\frac{DAU}{\#Installations} > 0\%$ in all `Facebook` from Dataset I. Third, we found that the top 50 most installed applications are common in both sets.

6.2.5 Data Release

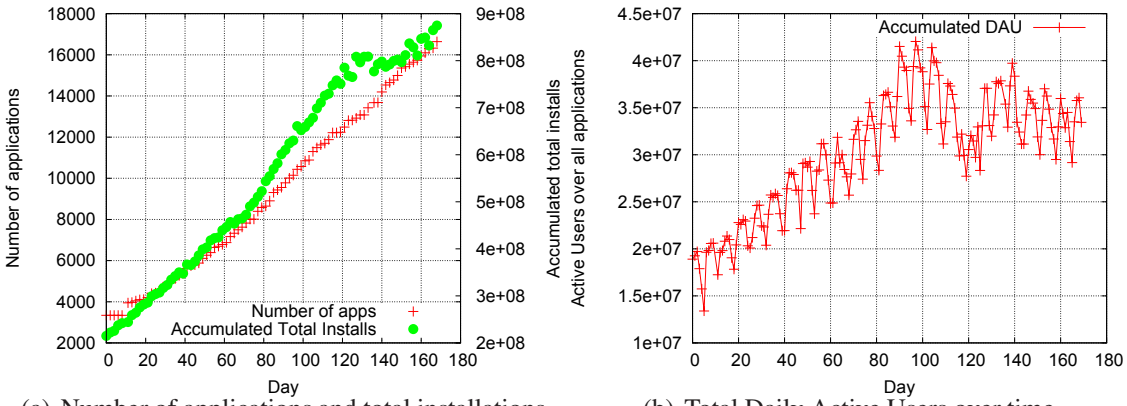
We have made available to the research community Dataset II, which contains the list of application installations for ∼ 300K users. To preserve user privacy, we have anonymized all userIDs. More details and downloading instructions are contained at `http://odysseas.calit2.uci.edu/research/osn.html`

6.3 Data Analysis

In this section, we analyze the two data sets and provide statistics about the popularity of `Facebook` applications. We begin the analysis of `Facebook` application statistics by classifying `Facebook` application and ranking categories based on popularity. We proceed with obtaining statistics with respect to their growth. We investigate the power-law distribution in application popularity in our Adonomics data set.

6.3.1 Aggregate Facebook Application Statistics

We start by looking at all third-party `Facebook` applications together as an aggregate. Fig. 6.2(a) shows that the total number of applications and the total number of installations increases almost linearly over the 170 days of Data Set I.



(a) Number of applications and total installations (b) Total Daily Active Users over time

Figure 6.2: Evolution of `Facebook` applications in aggregate. (left) Number of applications and total installations over time. (right) Total Daily Active Users over time.

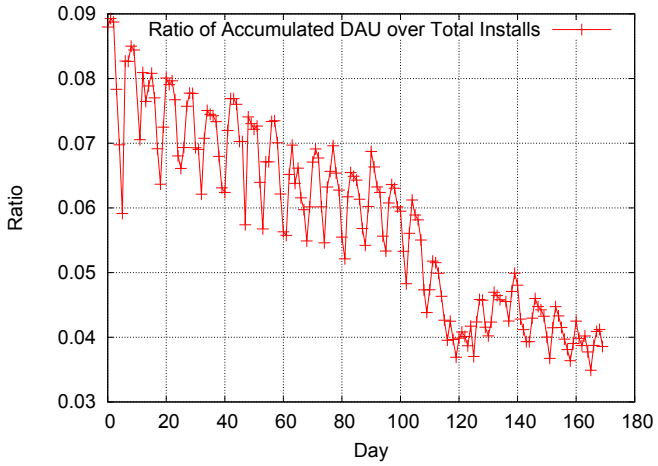


Figure 6.3: Evolution of `Facebook` applications in aggregate. Ratio of Daily Active Users over total installs

We then look at how many of these installed applications are actually active. Fig. 6.2(b) plots the number of active users over time: initially this number follows the growth of total installations, indicating that most users engage their installed applications. However, after day 104 the number of active users saturates at 42M and subsequently drops and stays at

around 35M. This indicates a particular user excitement during the months August 2007 to November 2007 which eventually fades out as some users realize that several of the applications do not satisfy their needs.

In Fig. 6.3, we look at the ratio of total daily active users (from Fig. 6.2(b)) over the total installations per day (Fig.6.2(a)). This ratio continuously decreases from 9% down to 4%. This indicates that an increasing number of applications competes for user attention that has plateaued.

Another observation from Fig. 6.2(b) and (c) is the weekly usage pattern. Although not clearly visible in these plots, our data analysis reveals that Tuesday and Wednesday were typically the most active and Saturday and Sunday were the least active days of the week.

6.3.2 Popularity of Individual Applications

We now turn our attention from `Facebook` as an aggregate to individual applications. We find that the popularity distribution is highly skewed; this is expected since popular applications tend to be more visible and solicit more invitations. Notice however that, as discussed above, the results can be quite different depending on whether we consider the number of *installations* or the number of *daily active users (DAU)* as the measure of popularity.

Fig. 6.1 shows the distribution of number of installations per application as found in Data Sets I and II. 10% of the top ranked applications account for 98% of total installations. The distribution of installations per application is approximated by a log-normal distribution with mean $\mu = 7.13$ and standard deviation $\sigma = 2.63$.

In Fig 6.4, we plot the number of daily active users per application distribution as a complementary cumulative probability function (CCDF). We are interested in determining whether

this distribution obeys a power law. In that case, the probability of an application having k installations would be $P(k) \sim k^{-\gamma}$, where γ is the power law coefficient. $\gamma \leq 3$ would indicate that there is a relatively small number of applications with a very large number of daily active users.

We use the techniques presented in [96] to fit a power-law distribution to the data. We use the discrete maximum likelihood estimator to compute the fitting power-law scaling parameter α, along with the Kolmogorov-Smirnov-based approach to estimate the lower cutoff x_{min} for the scaling region. We also use the Kolmogorov-Smirnov statistic D to compute the goodness-of-fit and the Kolmogorov-Smirnov test to compute a p-value for the estimated power-law fit. The estimated power-law fit gives parameters $\alpha = 1.57$, $x_{min} = 514$, $D = 0.0418$ which seems to best approximate the distribution among other known distributions we tried (including exponential, Weibull, and log-normal). We find that excluding the applications with less than $x_{min} = 514$ DAU, 20% of the top ranked applications account for 89.6% of total user engagements. However, we find that $p = 0$ for the given power-law fit parameters, which means that the power-law hypothesis is rejected in the strict statistical sense.

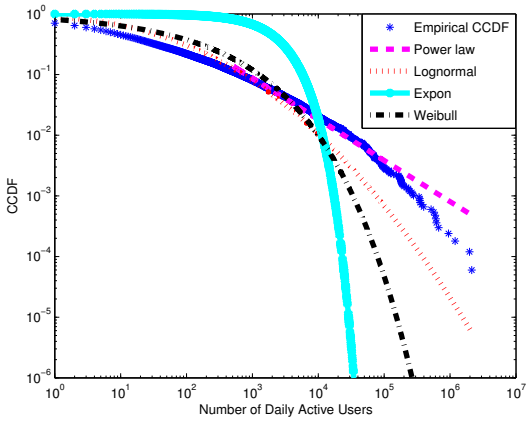


Figure 6.4: Complementary Cumulative Distribution Function (CCDF) of Daily Active Users per application.

The main observation is that the popularity distribution is highly skewed, with regards to

	Application Category	Most Popular applications
1	Friend Comparison	Top Friends, Compare People
2	Casual Communication	Super Wall, Fun Wall
3	Rating, Taste Matching Recommending	Movies, iLike,
4	Gestures	Super Poke, Hug me
5	Self Expression	Bumper Sticker, Green Patch
6	Gifts	Growing Gifts, Free Gifts
7	Meeting People	Are You Interested?, Zoosk
8	Casual Gaming	Scrabulous,Friends for Sale
9	Multimedia	Entourage, Friend Block
10	News and alerts	Birthday Calendar,Horoscopes Birthday Alert
11	Generic Utility	Mobile, Who's Online

Table 6.2: `Facebook` application classification. We also list the 2 most popular applications in each category.

both metrics of popularity. However, understanding the underlying process that leads to this property and finding the appropriate distribution fit is part of ongoing work.

6.3.3 The Effect of Application Category

There are several factors that may affect the popularity of an application. One such factor seems to be the type/category of application. In this section, we classify applications in thematic categories and look closer at the statistics and evolution of particular categories.

In its application directory [94], `Facebook` features an application classification. However, that is in many cases nonsensical (e.g. "Social Profile", which we list as "Casual Communication", is listed under the vague "Political" category). Thus, we use our own classification, which is inspired in great part by the one found in [97].

Table 6.2 summarizes our classification and lists the most popular applications in each

category. Here, we give a high level description of each category. We call the first category "Friend Comparison": it includes applications that allow users to declare best friends and compare friend traits. The second category is "Casual Communication", which includes applications that allow users to exchange messages and write on each other's wall. We call the third category "Rating, Taste Matching and Recommendations": it enables users to review, compare and recommend items spanning from music to restaurants. The fourth category is "Gestures": it includes applications that allow users to perform virtual gestures (poke, bite to convert to a zombie etc). The fifth category is called "Self Expression" and enables users to express moods, political opinions etc. The sixth category is "Gifting", enabling users to exchange virtual gifts. It is notable that the "Gifting" category is now making the transition from the virtual to the real world, providing a `Facebook` interface to online real gift shops. Early versions of these "Gifting" applications required users to pay a small amount in order to give a virtual gift but the category is now dominated by free services. The seventh category is "Meeting People", which is of particular interest to online dating services. The eighth category is 'Casual Gaming" with entries such as virtual versions of Scramble. Next comes the "Multimedia" category, which consists of applications that enable a user to listen to music, view photos and videos from within its `Facebook` page e.g. "YouTube Video Box". Another category is "News and Alerts", with applications such as "Google News". We call the last category "Generic Utility". It covers all the applications that do not fall under any of the above categories and instead serve a distinct useful functionality. In many cases they are front-ends to other web-based services, e.g. PayPal.

Comparing to Adonomics, our classification includes all applications and not only the most popular ones. We place the "Movies" and "Music" categories under the "Rating, Taste Matching and Recommendations" one. We removed the "Monsters" category and placed its applications under the "Casual Communication". We place "Fans" under "Self Expression". We added the "News and Alerts", "Multimedia" and "Generic Utilities" categories.

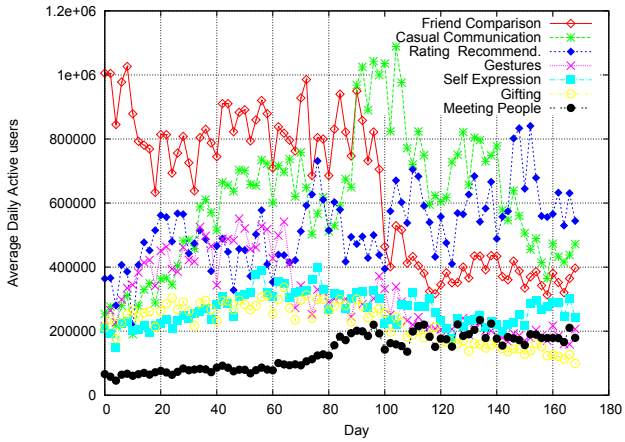


Figure 6.5: Average Daily Active Users (DAU) per application category. Categories are listed in the order of their total DAU rank.

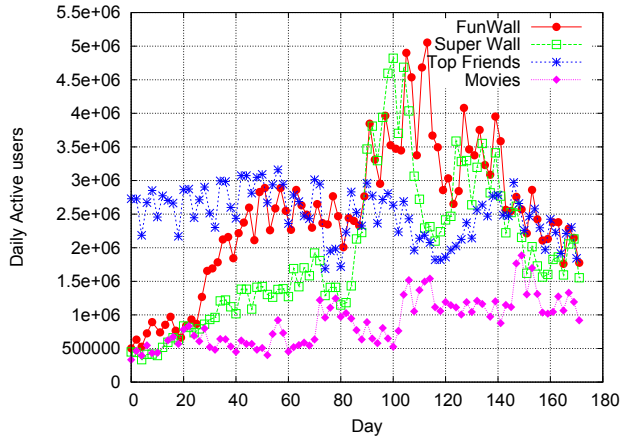


Figure 6.6: Daily active users of top 4 applications.

We are now ready to look closer at the average popularity of application categories and at the popularity of individual applications. In particular, we are interested in the most popular among them.

First, we ask whether popularity depends on the application category. We select the 100 most active applications on 03/05/2008 and we classify them in the aforementioned categories. We compute the average DAU of the applications in each category for every day in a 170-day period. In Fig. 6.6 we include the 7 top categories ranked according to the sum of their average DAU over all days. It turns out that the "Friend Comparison" and "Casual Communication" categories are the most popular. This indicates that `Facebook`

best serves the need of users to declare how they feel about their friends and the need to exchange messages. Interestingly, the fourth most popular category, "Gestures", was initially very popular but was subsequently matched by the "Self Expression" and "Meeting People" categories. A possible explanation is that the initial craze with applications of the likes of "Vampires", eventually turned into annoyance, while the applications in the other categories became more useful. We also observe that, in general, the user activity of the top ranked categories follows the same trends as the total user activity shown in Fig. 6.2(b): activity peaks at around the same time (95th-105th) day and drops afterward.

Second, we looked at the 5 most popular individual applications every day in the 170-day period. Interestingly, there were only 17 unique applications among them, which is much smaller than the number $5 \cdot 170 = 850$ that would correspond to a different top-5 popular applications every day. This indicates that the most popular applications remain popular throughout the entire period. Furthermore, we observed that 16 out of these 17 applications were present from the beginning of the period. Fig 6.6 shows the DAU evolution over time for the four most popular applications. We observe that all four applications belong in the three most popular categories. It is notable that the two most popular applications, which belong in the "Casual Communications" category, exhibited impressive viral growth. E.g. "Super Wall" grew from $\sim$1.1M to $\sim$4.8M DAU in 20 days.

6.4 User Coverage Simulation

In this section, we develop a simulation model that generates the bipartite graph between the users and the applications installed. The input to the simulator is the list of applications, the number of installations per application and the number of users, all easily obtainable from `Facebook`'s application directory [94]. The output of the simulator is the bipartite graph (which we cannot obtain without crawling). Based on this graph we can compute

several metrics of interest such as: the distribution of number of applications installed per user, the number of applications needed to cover all users, etc.

Such a simulator would be useful to those interested in reaching users via applications, such as advertisers. For example, an advertiser might be interested in which applications she should purchase to cover a certain set of users; other constraints, such as cost, could also be taken into account in these optimization problems. User coverage strategies can be studied if the bipartite graph is available. However, this is not a trivial task in practice. No `Facebook` analytics service offers statistics that can be directly used to infer the coverage of more than one application. In addition, privacy-conscious `Facebook` application operators are likely not to release information on which users have installed their applications. On the other hand, developers and `Facebook` already release statistics about application usage and user demographics. This can be then used as input to our simulator, in the absence of detailed crawled data. (e.g. what percentage of the application's users is in the age range 25 to 30).

6.4.1 Preferential Installation

The skewed distribution of application popularity motivated us to investigate whether rich-get-richer types of mechanisms can apply to the process according to which individual users install applications. At the heart of our simulator lies a preferential installation process, according to which users that have already several applications installed are more likely to install even more new applications. We call this mechanism "preferential installation", in reference to the "preferential-attachment" model for random networks [98].

In particular, our simulation proceeds as follows. Consider the users as bins and the applications as balls of different colors. The number of installations of an application are considered as balls of the same corresponding color. At each step of the simulation, a

ball is selected, starting from the color (application) with the most balls (installations), and proceeding to the next color once all balls of the current color are exhausted. Once a ball is selected, it must be assigned with a certain probability to one of the bins that does not contain a ball of the same color (assuming that a user installs a certain application only once).

The probability that a ball chooses a certain bin specifies the behavior of the installation model and eventually the statistics we will observe. For example, a ball could choose uniformly at random among the bins; this turned out to be a very bad model for our data, as shown in Fig. 6.7. We then explored preferential installation models that assign more probability to bins that have already several balls; this captures the intuition that a user that has already several applications installed is more prone to install new applications as well. In particular, the probability of a ball (application) to be installed at a bin (user) i is calculated as

$$P_{bin}(i) = \frac{balls(i)^{\rho} + init}{\Sigma_{j \in B}(balls(j)^{\rho} + init)} \tag{6.1}$$

where $balls(i)$ is the number of balls that bin i contains prior to this installation, B is the set of bins without a ball of the same color and ρ is an exponent that can magnify the effect of preferential attachment. The parameter $init$ defines the initial probability $P_{bin}(i)$ of a bin without any installations, and controls the significance of the number of balls in a bin in the early steps of the simulation. In the 1^{st} iteration of the simulation, all bins are equally likely to be chosen w.p. $1/|B|$; in the 2^{nd} iteration the ones with already 1 ball have $\frac{init+1}{init}$ times higher probability than the ones with 0 balls.

After careful tuning of the parameters ($\rho = 1.6$ and $init = 5$) this process results in the same statistics as those found in Data Set II. Fig. 6.7 shows the distribution of the number of installations per user as found in Data Set II. It also shows that a very similar

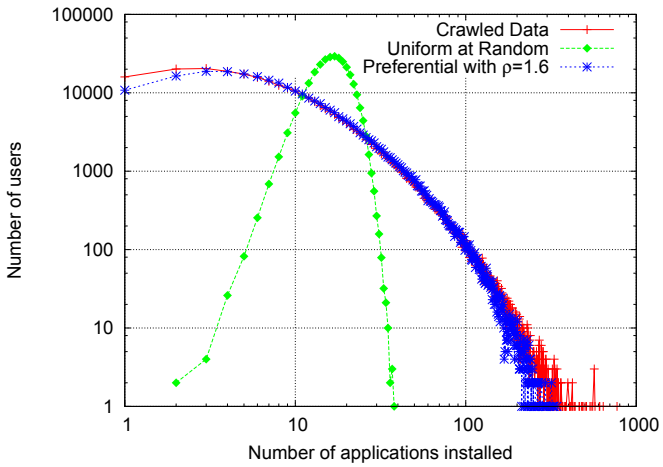


Figure 6.7: Distribution of per-user number of installed applications in the crawled data (Data Set II) and in the simulation-generated data.

distribution is generated by our simulation model with the above parameters and random order of application installations. In the same figure, we show that a uniform installation process yields a very different distribution. We also ran simulations in which applications are installed in increasing or decreasing order of popularity and observed similar results.

Limitations. Our simulator captures accurately the preferential installation of applications to users with already installed applications. However, it does not currently capture other factors that may affect the probability of installing an application such as: (i) demographics the user belongs to; (ii) previously installed applications, e.g. with similar function; (iii) installations of the user's friends. Furthermore, our current model is based on the number of installations not DAU. We plan to extend and refine this model in future work to include these considerations.

6.4.2 User Coverage Analysis

For validation, we compare user coverage statistics obtained by the model and the crawled data set (we have determined in Section 6.2.4 that the crawled data set is sufficiently representative of `Facebook`). The simulation can be used to predict user coverage of a target

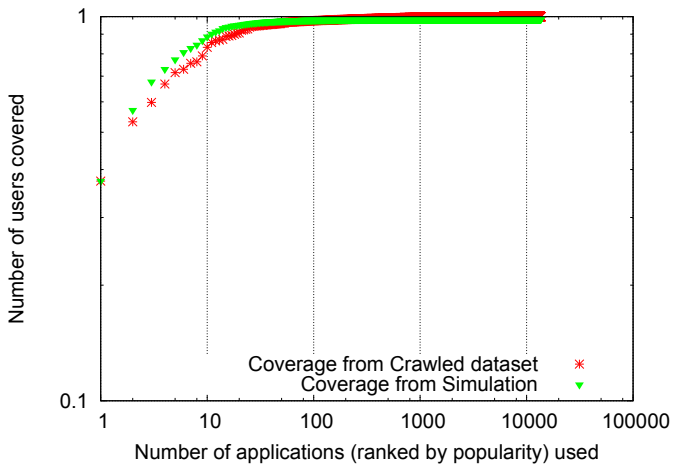


Figure 6.8: User coverage. The x axis is fraction of applications (ordered in decreasing popularity). The y axis is the fraction of users that are covered.

demographic given the number of installations per application. Fig. 6.8 shows a close match between user coverage in the crawled data and in the output of our simulator when run for the same popularity distribution. (The PDF is shown in Fig. 6.1: it is the same for the sample, Data Set II, and the full Data Set I and can be modeled well as log-normal.) Table 6.3 shows another example of how our simulator can be used to study user coverage. We randomly selected 5 applications and looked at their user coverage, which was 51.5% of all users in Data Set II and 51.1% in the simulation-generated data. Such information would be particularly useful to an advertiser. For example, the two applications that rank 1 and 2 may cover 50% of the users in the sample but they may cost a lot more to acquire than all 5 listed applications combined. We repeated this process 50 times, randomly selecting 5-20 applications and found that the sample coverage was within $\pm 4\%$ of the simulated coverage.

6.5 Summary

We have presented the first measurement-based characterization of the popularity and usage of third-party `Facebook` applications. Among other findings, our analysis reveals that the

Application Popularity Rank	# Installations	Coverage Real(%)	Coverage Simulation(%)
5	87609	30.2	30.2
15	45396	41.6	39.8
46	19504	43.9	42.6
99	9685	44.9	43.6
12	50825	51.5	51.1

Table 6.3: User coverage statistics. The first column corresponds to the rank of (five randomly chosen) applications according to their number of installations in Data Set II.

distribution of popularity among `Facebook` applications is highly skewed with a small percentage of applications accounting for the largest portion of total installations and user activity. In addition, we show that the distribution of application to users can be modeled by a preferential installation process.

Chapter 7

Conclusion

The popularity of online social networks has increased dramatically in the recent years, counting more than 1 billion users in 2010. Unlike past web applications, OSNs are user-centered and offer a variety of tools to facilitate information sharing and communication between their users. They have become an important phenomenon on the Internet and their existence has far reaching implications, beyond providing rich datasets in large scale to researchers. For example, the structure of online social networks has been used in the past to improve existing or build new Internet systems *i.e.,* by leveraging the trust between users or offering personalized services and recommendations.

Our study is motivated by the fact that complete OSN datasets are typically not available and OSN operators are usually unwilling to share data. Therefore, obtaining data by sampling is essential for practical estimation of OSN properties. In this thesis we presented a measurement study of online social networks and implemented efficient crawlers that addressed the challenges involved in crawling their users. We presented statistically principled approaches to collect representative samples of OSN users, appropriate for further statistical analysis. We performed Internet measurements of large scale online social net-

works, Facebook and Last.fm, which can be challenging in practice. Finally, using our representative collected datasets, we performed characterization studies of Facebook users, Last.fm users, and Facebook applications.

In Chapter 3, we described the challenges involved in crawling online social networks and our approach in addressing them. We built efficient crawlers that decrease multifold our data collection time. We coordinate a distributed crawling process and use multiple layers of parallelism in each individual machine.

In Chapter 4, we implemented and compared several crawling methods for the purpose of collecting a representative sample of Facebook users. We showed that two random walk based methods (MHRW, RWRW) performed remarkably well. Our recommended approach consists of (i) running multiple random walks in parallel (ii) correcting the inherent bias of the random walks either during the walk or at the end of the walk (iii) ensuring convergence using appropriate online diagnostics run on several metrics of interest. We demonstrated that, for all practical purposes, both methods achieve a perfectly random sample of users while traditional alternative techniques (BFS and RW) introduce signicant bias on degree distribution and other metrics. The sampling approach we recommend is principled, effective, and applicable to any OSN, since it is based on crawling the friendship relation which is a fundamental primitive in any OSN. Based on one of our unbiased samples of Facebook users, we presented a study of topological features and privacy awareness. Unlike previous studies, our results are representative of the entire social graph.

In Chapter 5, we introduced multigraph sampling, a novel methodology for collecting an representative sample of OSN users by crawling not only on one, but on multiple relations. Intuitively, multigraph sampling can potentially overcome two limitations of single graph sampling. First, the multigraph may be connected, even in cases when each individual graph is not. For example, multigraph sampling can discover users that have no friends, which would be impossible by crawling the friendship graph. Second, multigraph may

have better mixing properties, even when the individual graphs are highly clustered. We designed an efficient two stage algorithm for multigraph sampling that avoids enumerating a large number of users in each relation. We demonstrated the benefits of multigraph sampling by simulation of synthetic random graphs and by measurements of `Last.fm`.

In Chapter 6, we presented the first study to characterize the statistical properties of OSN applications. We collected and analyzed two data sets pertaining to `Facebook` applications. In addition, we proposed a simple and intuitive method to simulate the process with which users install applications. Using this method we showed how to determine the user coverage from the popularity of applications, without detailed knowledge of how applications are distributed among users.

Bibliography

[1] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[2] D.M. Boyd and N.B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2008.

[3] A.C. Weaver and B.B. Morrison. Social networking. *Computer*, 41(2):97–100, 2008.

[4] List of social networking websites. `http://en.wikipedia.org/wiki/List_of_social_networking_websites`, July 2010.

[5] W. Kim, O.R. Jeong, and S.W. Lee. On social Web sites. *Information Systems*, 35(2):215–236, 2010.

[6] Nielsen statistics. `http://blog.nielsen.com/nielsenwire/online_mobile/social-media-accounts-for-22-percent-of-time-online/`, June 2010.

[7] Alexa traffic statistics for facebook. `http://www.alexa.com/siteinfo/facebook.com`, June 2010.

[8] J.M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine (s) that could: Scaling online social networks. *ACM SIGCOMM Computer Communication Review*, 40(4):375–386, 2010.

[9] S. Agarwal. Social networks as Internet barometers for optimizing content delivery networks. In *Advanced Networks and Telecommunication Systems (ANTS), 2009 IEEE 3rd International Symposium on*, pages 1–3. IEEE, 2010.

[10] A. Nazir, S. Raza, D. Gupta, C.N. Chuah, and B. Krishnamurthy. Network level footprints of facebook applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 63–75. ACM, 2009.

[11] Exploiting Social Networks for Internet Search. In *HotNets*, 2006.

[12] A. Mislove, A. Post, P. Druschel, and K.P. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 15–30. USENIX Association, 2008.

[13] Michael Sirivianos, Kyungbaek Kim, and Xiaowei Yang. Introducing Social Trust to Collaborative Spam Mitigation. In *INFOCOM*, 2011.

[14] M. Sirivianos, K. Kim, and X. Yang. FaceTrust: Assessing the Credibility of Online Personas via Social Networks. In *Proceedings of the 4th USENIX conference on Hot topics in security*, page 2. USENIX Association, 2009.

[15] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. In *Science*, 2002.

[16] Y.Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of Topological Characteristics of Huge Online Social Networking Services. In *Proc. of WWW*, 2007.

[17] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470. ACM, 2008.

[18] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis. Tastes, ties, and time: A new social network dataset using Facebook.com. Social Networks, 2008.

[19] Uniform sampling of facebook users: Publicly available datasets. http://odysseas.calit2.uci.edu/fb/, 2009.

[20] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[21] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Samrat Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proc. of IMC*, 2007.

[22] Alan Mislove, Hema Swetha Koppula, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Growth of the flickr social network. In *Proc. of WOSN*, 2008.

[23] Alan Mislove, Hema Swetha Koppula, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Growth of the flickr social network. In *Proceedings of the 1st ACM SIGCOMM Workshop on Social Networks (WOSN'08)*, August 2008.

[24] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proc. of EuroSys*, 2009.

[25] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.

[26] S. H. Lee, P.-J. Kim, and H. Jeong. Statistical properties of sampled networks. *Phys. Rev. E*, 73:016102, 2006.

[27] L.Becchetti, C.Castillo, D.Donato, and A.Fazzone. A comparison of sampling techniques for web graph characterization. In *LinkKDD*, 2006.

[28] Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. In *STOC*, 2005.

[29] M. Kurant, A. Markopoulou, and P. Thiran. On the bias of breadth first search (bfs) and of other graph sampling techniques. In *Proc. of ITC '22*, 2010.

[30] S. Ye, J. Lang, and F. Wu. Crawling online social graphs. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pages 236–242. IEEE, 2010.

[31] L. Lovasz. Random walks on graphs. a survey. In *Combinatorics*, 1993.

[32] Monika R. Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. On near-uniform url sampling. In *Proc. of WWW*, 2000.

[33] Eda Baykan, Monika Rauch Henzinger, Stefan F. Keller, Sebastian De Castelberg, and Markus Kinzler. A comparison of techniques for sampling web pages. In *STACS*, 2009.

[34] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proc. of IMC*, 2006.

[35] Amir Rasti, Mojtaba Torkjazi, Reza Rejaie, Nick Duffield, Walter Willinger, and Dan Stutzbach. Respondent-driven sampling for characterizing unstructured overlays. In *INFOCOM Mini-Conference*, April 2009.

[36] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks. In *Proc. of INFOCOM*, 2004.

[37] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proc. of ACM SIGKDD*, 2006.

[38] D.D. Heckathorn. Respondent-driven sampling: A new approach to the study of hidden populations. *Social Problems*, 44:174199, 1997.

[39] M.J. Salganik and D.D. Heckathorn. Sampling and estimation in hidden populations using respondent-driven sampling. *Sociological Methodology*, 34:193239, 2004.

[40] N. Metropolis, M. Rosenblut, A. Rosenbluth, A. Teller, and E. Teller. Equation of state calculation by fast computing machines. *J. Chem. Physics*, 21:1087–1092, 1953.

[41] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, 1996.

[42] Amir Rasti, Mojtaba Torkjazi, Reza Rejaie, Nick Duffield, Walter Willinger, and Dan Stutzbach. Evaluating sampling techniques for large dynamic graphs. Technical report, University of Oregon, Sept 2008.

[43] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *Proc. of ACM SIGCOMM WOSN*, 2008.

[44] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM review*, 46(4):667–689, 2004.

[45] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *arXiv:1002.1751*, 2010.

[46] W. R. Gilks and G. O. Roberts. Strategies for improving MCMC. In *Markov Chain Monte Carlo In Practice*, pages 89–114. Chapman and Hall/CRC, Boca Raton, FL, 1996.

[47] S.K. Thompson. Adaptive cluster sampling. *Journal of the American Statistical Association*, 85(412):1050–1059, 1990.

[48] S.K. Thompson. Stratified Adaptive Cluster Sampling. *Biometrika*, 78(2):389–397, 1991.

[49] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proc. of ACM SIGKDD*, 2006.

[50] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *Proc. of ACM SIGKDD*, 2006.

[51] P. Sarkar and A.W. Moore. Dynamic social network analysis using latent space models. *ACM SIGKDD Explorations Newsletter*, 7(2):31–40, 2005.

[52] Walter Willinger, Reza Rejaie, Mojtaba Torkjazi, Masoud Valafar, and Mauro Maggioni. Osn research: Time to face the real challenges. In *HotMetrics*, 2009.

[53] U.G. Acer, P. Drineas, and A.A. Abouzeid. Random walks in time-graphs. In *Proceedings of ACM MobiOpp '10*, 2010.

[54] M.P.H. Stumpf, C. Wiuf, and R.M. May. Subnets of scale-free networks are not scale-free: sampling properties of networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(12):4221, 2005.

[55] B. Krishnamurthy. A measure of online social networks. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–10. IEEE, 2009.

[56] Amanda L. Traud, Eric D. Kelsic, Peter J. Mucha, and Mason A. Porter. Community structure in online collegiate social networks. arXiv:0809.0960, 2008.

[57] Balachander Krishnamurthy and Craig E. Wills. Characterizing privacy in online social networks. In *Proc. of WOSN*, 2008.

[58] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano. Eight friends are enough: social graph approximation via public listings. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 13–18. ACM, 2009.

[59] L. A. Adamic, O. Buyukkokten, and E. Adar. A Social Network Caught in the Web. In *First Monday*, 2003.

[60] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proc. of IMC*, 2007.

[61] A. Nazir, S. Raza, and C.N. Chuah. Unveiling facebook: a measurement study of social network based applications. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 43–56. ACM, 2008.

[62] I. Konstas, V. Stathopoulos, and J.M. Jose. On social networks and collaborative recommendation. In *Proc. of ACM SIGIR '09*, 2009.

[63] R. Schifanella, A. Barrat, C. Cattuto, B. Markines, and F. Menczer. Folks in folksonomies: Social link prediction from shared metadata. In *Proc. of WSDM '10*, 2010.

[64] N. Baym and A. Ledbetter. Tunes that bind? Predicting friendship strength in a music-based social network. *Information, Communication and Society*, 12(3), 2009.

[65] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. In *Proc. of the National Academy of Sciences*, 2005.

[66] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.

[67] About facebook's security and warning systems. `http://www.w3.org/TR/xpath20/`, 2010.

[68] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[69] Memcached. `http://www.memcached.org/`.

[70] Marc Najork and Janet L. Wiener. Breadth-first search crawling yields high-quality pages. In *Proc. of WWW*, 2001.

[71] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. In *Nature*, 1999.

[72] M.Hansen and W.Hurwitz. On the theory of sampling from finite populations. *Annals of Mathematical Statistics*, 14, 1943.

[73] Erik Volz and Douglas D. Heckathorn. Probability based estimation theory for respondent-driven sampling. *Journal of Official Statistics*, 2008.

[74] O. Skare. Improved sampling-importance resampling and reduced bias importance sampling. Scandin. journ. of stat., theory and apps, 2003.

[75] J. Geweke. Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In *Bayesian Statist. 4*, 1992.

[76] A. Gelman and D. Rubin. Inference from iterative simulation using multiple sequences. In *Statist. Sci. Volume 7*, 1992.

[77] Alberto Leon-Garcia. *Probability, Statistics, and Random Processes For Electrical Engineering*. Prentice Hall, 2008.

[78] Facebook announcement on 64 bit userids. `http://developers.facebook.com/blog/post/226`, May 2009.

[79] The facebook blog: Growing beyond regional networks. `http://blog.facebook.com/blog.php?post=91242982130`, June 2009.

[80] Fb statistics. http://facebook.com/press/info.php?statistics, July 2009.

[81] Inside facebook. http://www.insidefacebook.com/2009/07/02/facebook-now-growing-by-over-700000-users-a-day-updated-engagement-stats/.

[82] Facebook's id numbering system. `http://www.quora.com/What-is-the-history-of-Facebooks-ID-numbering-system`.

[83] D. B. Rubin. Using the SIR algorithm to simulate posterior distributions. In *Bayesian Statistics 3*. 1988.

[84] Mark Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics 46*, 2005.

[85] Mark Newman. Assortative mixing in networks. In *Phys. Rev. Lett. 89*.

[86] D.D. Heckathorn. Respondent-driven sampling ii: Deriving valid estimates from chain-referral samples of hidden populations. *Social Problems*, 49:11–34, 2002.

[87] Sheldon. Ross. *Introduction to Probability Models*. Academic Press, 8 edition.

[88] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, 1996.

[89] G.A.F. Seber. The estimation of animal abundance and related parameters. *New York*, 1982.

[90] Minas Gjoka, Maciej Kurant, Carter Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proc. of INFOCOM*, 2010.

[91] Facebook platform. http://developers.facebook.com/.

[92] Business Week. Building a Brand with Widgets. http://www.businessweek.com/technology/content/feb2008/tc20080303_000743.htm.

[93] Adonomics. http://www.adonomics.com, 2008.

[94] Facebook application directory. http://www.facebook.com/apps.

[95] Minas Gjoka. Scripts for Crawling Facebook. http://www.ics.uci.edu/~mgjoka/facebook/.

[96] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law Distributions in Empirical Data. In *http://arxiv.org/abs/0706.1062v1*, Jun 2007.

[97] Building the Social Suite of Category Killer Apps for Facebook. http://blog.adonomics.com/2008/02/22/building-the-social-suite-of-category-killer-apps-for-facebook/, 2008.

[98] A.L. Barabassi and R. Albert. Emergence of scaling in random networks. In *Science*, 1999.

[99] E.D. Kolaczyk. Sampling and Estimation in Network Graphs. *Statistical Analysis of Network Data*, pages 1–30, 2009.

[100] Facebook warnings - help center. http://www.facebook.com/help/?page=421, 2010.

[101] Facebook - automated data collection terms. http://www.facebook.com/apps/site_scraping_tos.php, April 2010.

[102] Last.fm api. http://www.last.fm/api/.

Appendices

A Uniform Sample of Users with Rejection Sampling

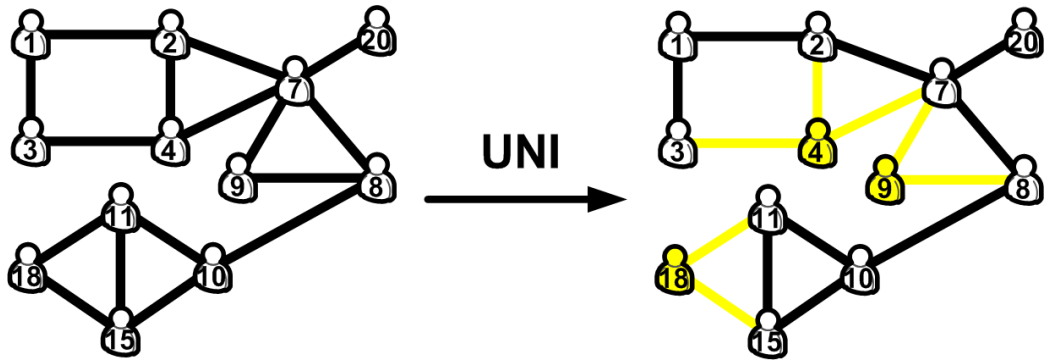


Figure A.1: Example of UNI: yellow nodes are sampled and all adjacent edges and nodes are observed

In Sections 4.2.3 and 5.4.1, we used UNI to obtain a uniform sample of users for `Facebook` and `Last.fm` respectively. There are two requirements to implement UNI, which were easily satisfied for the OSNs we crawled. First, we needed to know the range of userIDs assigned to users. Knowledge of the actual distribution of userIDs in the userID space is not needed *i.e.,* userIDs can be allocated non-sequentially or unevenly across the userID space. Second, we needed to be able to query the OSN to return data for the selected userID, if it exists, or return an error message if it does not exist. This is a commonly supported feature on all OSNs.

UNI is a labeled star sampling scheme [99], because we keep all sampled users and ob-

served adjacent edges and nodes. We implement UNI by repeating the following two steps: (i) we generate a userID uniformly at random in the known range of userID space (ii) we query the OSN for user data (user properties and user friend list) for the chosen userID. If the OSN does not return an error message, we include the chosen user in our sample, otherwise we discard it. Fig A.1 shows an example of a graph with a userID space in the range of 1-20, with three nodes sampled.

Proposition: UNI yields a uniform sample of the *existing (allocated)* user IDs in an Online Social Network for *any allocation policy* (*e.g.,* even if the userIDs are not evenly allocated in the userID address space of the Online Social Network).

Proof. Denote by U the set of all possible user IDs, *i.e.,* the set of all integers in $[0, maxuserID-1]$. Let $A \subset U$ be the set of allocated user IDs in `Facebook`. We would like to sample the elements in A uniformly, *i.e.,* with pdf $f_A(x) = \frac{1}{|A|}\sum_{y\in A}\delta(y)$, where $\delta(y)$ is the Dirac delta. The difficulty is that we do not know the allocated IDs A beforehand. However, we are able to verify whether id x exists ($x \in A$) or not, for any x.

To achieve this goal, we apply rejection sampling [77] as follows. Choose uniformly at random an element from U (which is easy), *i.e.,* with pdf $f_U(x) = \frac{1}{|U|}\sum_{y\in U}\delta(y)$. Let $K = \frac{|U|}{|A|}$ s.t. $f_A(x) \leq K \cdot f_U(x)$ for any x. Now, draw x from $f_U(x)$ and accept it with probability $\frac{f_A(x)}{K\cdot f_U(x)} = 1_{x\in A}$, *i.e.*, always if $x \in A$ (ID x exists/is allocated) and never if $x \notin A$ (ID x is not allocated). The resulting sample follows the distribution $f_A(x)$, *i.e.,* is taken uniformly at random from A (the set of *allocated* user IDs). □

The above is just a special case of *rejection sampling* [77], when the distribution of interest is uniform. It is presented here for completeness, given the importance of UNI sampling as "ground truth" in Chapters 4, 5.

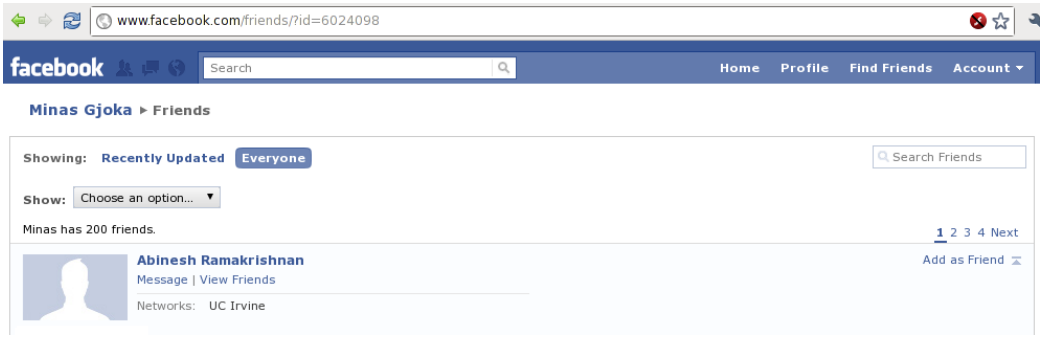


Figure B.2: Crawling the social graph by scraping `Facebook`'s web interface

B Data Collection

B.1 Crawling Facebook

In this section we describe the process we followed to explore `Facebook`'s social graph.

Fetching Friend List

We first looked into the ability to use `Facebook` API calls to list a user's friends, given a userID, and thus explore `Facebook`'s social graph. That would allow us to incur minimum overhead in terms of bandwidth. `Facebook` has the API call `friends.get` which returns the userIDs of a user's friends. To annotate the information we collect for each user, we could then feed the list of userIDs to the API call `users.getInfo`. The latter returns a wide array of user-specific information for each userID passed, limited by the view of the current user.

Unfortunately, the API call friends.get is constrained to bring the friend list of the currently authenticated user only. `Facebook` applications commonly use this call to retrieve the friends of the users that have installed and given permission to a specific application. Therefore it cannot be used to access the neighborhood of any user, which is a necessary

primitive to explore the social graph. Since this option is not available, we resort to data scraping. The web page that contains the friend list for a user, with a given numerical userID, is at `http://www.facebook.com/friends/?id=userID`. Fig B.2 contains a screenshot of such a page rendered in a web browser. We parse this page to extract the basic node information shown in Fig 4.2.

`Facebook` is known to implement defense mechanisms against automated data collection and mining. `Facebook` pages such as [100] mention reasons for blocked accounts and give general guidelines to avoid getting blocked. However, no specific limits are mentioned. Recently, `Facebook` updated their crawling policy and allows whitelisting for crawling purposes [101].

Data Scraping Overhead

The list of friends is split into pages of 400 friends at most. Therefore multiple HTML requests might be required to get a user's list of friends. With very rough calculations, the size of each HTML page that contains list of friends is $55KB + N \times 1.75KB$ where N is the number of friends inside the page.

Graph API

In April 2010, `Facebook` launched a new set of API calls, the Graph API, to replace the old REST API. The Graph API provides a simple and consistent view of the social graph, uniformly representing objects (*e.g.,* users, groups, events) and the connections between them (*e.g.,* friendships, likes, tags). This study predates the introduction of the Graph API.

B.2 Crawling Last.fm

In this section, we describe the process we follow to implement multigraph sampling and to obtain the weekly charts.

Fetching Multigraph Relations

Similarly to `Facebook`, `Last.fm` contains a rich API that enables developers to write programs that utilize its data. [102] lists the available API methods for every type of object (*e.g.,* users, groups, tracks, artists) in `Last.fm`. In comparison, `Last.fm` is more open than `Facebook` regarding the sharing of data. Privacy controls are less restrictive *i.e.,* users can only choose to withhold release of recent activity and real time data in their profiles.

To collect data from `Last.fm`, we use a combination of API calls and data scraping. The relations between users that we used in our study are "Friends", "Groups", "Events", and "Neighbors". Consider that we are sampling user u. For efficient implementation of multigraph sampling we proceed in two stages, as shown in Fig 5.1(d). In the first stage, we discover the graphs of user u (on the relations we have chosen) and obtain their corresponding size. In our study, we use the API calls `user.getfriends`, `user.getneighbors`, `user.getpastevents`, and `user.getevents` to collect the list of friends, neighbors, past events, and future events respectively and the corresponding size. Due to a lack of an API call that lists the groups of a user, we use data scraping to collect the list of groups and corresponding size for each group. We treat each individual group and event as a different graph in the multigraph. We also consider the set of friends and neighbors to comprise the friends and neighbors graph respectively in the multigraph. At the end of the first stage, we select one of the graphs G_i we have discovered, proportionally to the size of the graph, in accordance with algorithm 2 in Section 5.2.

We should note that at the end of the first stage, we have not enumerated any user from any of the groups and events graphs. Each of these graphs is quite large (up to tens of thousands of users) and depending on the user, there are many groups or events per user (up to thousands). On the other hand, we have enumerated users of friends and neighbors since knowledge of their size is equivalent to enumeration [1] of users within them. Overall, our two stage approach saves us bandwidth and time by avoiding the enumeration of users for graphs that we are not going to sample from at each iteration.

In the second stage, we pick uniformly at random one of the nodes from the graph G_i, selected at the end of the first stage. If the graph G_i is a graph of a group or an event, we need to carefully implement this action to be efficient. More specifically, we do not need to enumerate all group members or event attendants from a group or event graph. Instead, we can take advantage of the pages functionality that OSNs often provide and only fetch the page that corresponds to the user selected uniformly at random. In our study, to fetch group members we use the API call `group.getmembers`, which returns 50 users per page. To fetch event attendants we use data scraping[2], which also returns 50 users per HTML page.

Fetching Weekly Charts

Last.fm generates weekly charts of the top 420 artists and tracks, compiled from weekly song plays by users. The ranking on these charts are based on the total number of individual listeners and not the number of actual plays. However, for the top 20 artists/tracks, numbers are provided for both unique listeners and actual plays. Fig B.3 shows the web page in `Last.fm` that displays the top artists for the week ending on July 11, 2010.

[1] There might be workarounds to enumerating friends but they are not necessarily more efficient. For example, we could extract the number of friends by data scraping. In general, in another setting we could do away with any kind of enumeration in the first stage.

[2] We prefer data scraping to the API call `event.getattendees` because the API call i) is not paged ii) does not return users that marked "maybe" for the event iii) is very slow for large events.

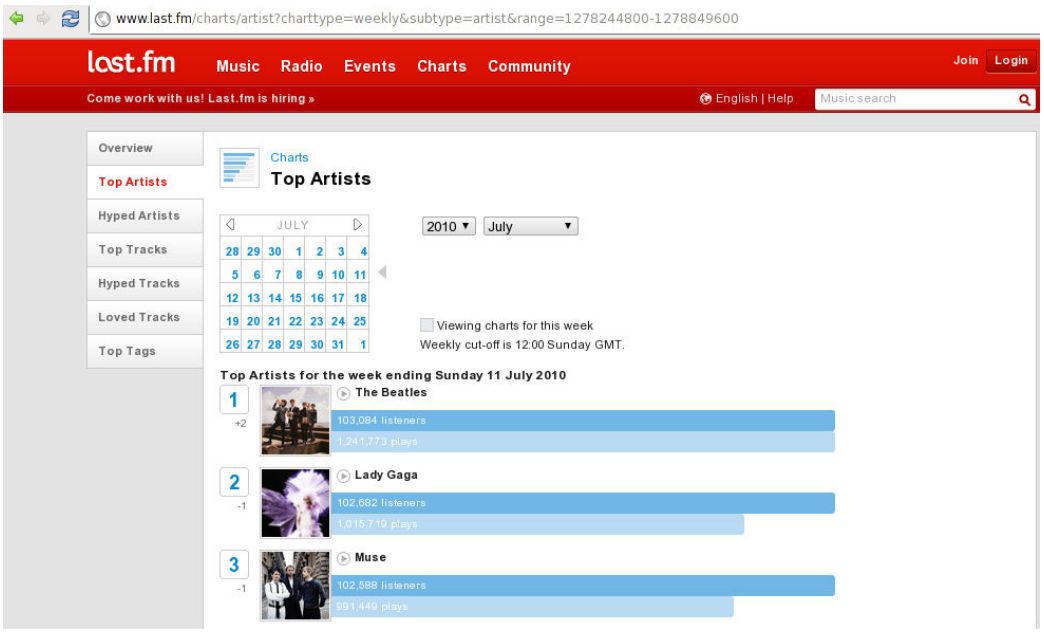


Figure B.3: Top Artist Charts aggregated weekly by `Last.fm`

In Section 5.4, we used the weekly charts as a source of ground truth and as an application of our multigraph sampling. More specifically, we used the collected user samples from the single graph and multigraph crawl types to produce chart rankings for artists and tracks. We then compared the inferred charts to the real charts made publicly available by `Last.fm`. To fetch the weekly artist and track plays for a user, we execute the API calls `user.getWeeklyArtistChart` and `user.getWeeklyTrackChart` respectively, for a given date range. We then obtain the real chart rankings by data scraping the `Last.fm` page at Fig B.3, for the same date range.