Understanding and Improving Online Social Interactions and Processes: Methods, Algorithms & Applications

by

Emmanouil Papangelis

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Understanding and Improving Online Social Interactions and Processes: Methods, Algorithms & Applications

Emmanouil Papangelis
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2015

Over the last years, the web witnesses a prolific growth largely due to the changing trends in the use of web technology that aims to enhance interconnectivity, self-expression and information sharing. These trends have led to the development and evolution of online social systems, but also to the creation of voluminous user-generated data. Even as these new online social media change the way we communicate, the social processes occurring remain governed by long-standing principles of human behavior and social interaction.

In this thesis, we focus on the algorithmic and computational issues involved in the study of online social interactions and processes, such as, *behavior*, *influence* and *diffusion*. We first present an analysis of user-generated content published online by non-technical users. Our analysis reveals cascading patterns of online communication and establishes evidence of a cascading effect occurring in online communities. Then, we design an analytical method for detecting social influence, the cause of cascading effects, in social systems. Our method suggests a strategy for studying the causality between changes of individual behavior and the social influence that individuals exert in their network. The ability of a network to carry on social processes is a characteristic that depends, to a great extent, on its topology. Therefore, slight modifications in the network topology, might have a dramatic effect on how efficiently social processes evolve. We approach network modification as a graph augmentation problem and come up with methods that can optimize connectivity properties of the network that can boost social processes. Finally, we present efficient methods that utilize information available in a user's social network in order to improve the quality of results they get back when they search the web. The main goal and premise is to re-order the list of retrieved results in a way that favors items that are more relevant to a user's interest towards a more personalized search experience.

Overall, formal understanding of principles and models of social interaction and processes that are persistent in online social media provides an opportunity to design and develop methods, algorithms and applications that can positively impact the way we interact with, search and access information online.

Dedication

To the endless scientific quest of the fundamental principles of knowledge, existence and conduct

Acknowledgements

Completion of a PhD thesis requires patience and perseverance - and a number of loyal friends and individuals. First, I would like to express my gratitude to my supervisor Nick Koudas for providing the opportunity to pursue this research, for invaluable guidance on technical issues of the thesis and for always setting high standards by example, which in turn challenges, motivates and inspires students. Moreover, I would like to thank Mark Chignell, Khai Truong, Yashar Ganjali, Angela Demke-Brown of University of Toronto and Panagiotis G. Ipeirotis of New York University for agreeing to serve as members of my final examination committee. With their experience, sincere assessment of my work, fruitful discussions and valuable suggestions helped to shape this research in a meaningful way.

One's expertise and understanding of a subject is further enhanced when applied within a social context. Throughout this journey I had the chance and pleasure to meet, interact and collaborate with a number of colleagues. I will always recall these years and come to realize the importance of spending long hours in a shared lab. There is a large number of social interactions that take place when you physically share a working or reading environment with colleagues. Sharing information, trying to prove yourself or each other wrong (or right), challenging yourself or being exposed to fallacies and pitfalls are all aspects of a social learning process where we are seeking acceptance from our social group and we are learning through influential models. Being exposed and engaging in such social interactions, on a daily basis, for a number of years, was critical to my understanding and appreciation of computer science concepts and is describing a learning environment that is difficult to replicate or experience elsewhere. With this opportunity I would like to thank my labmates: Nikos, Dimitris, Michalis, Albert, Michael, Amit, Chaitanya, Nilesh, Priyank, Alex, Mohammad and Oktie, as well as, any visitor in our lab. I would also like to thank Gautam Das of University of Texas, Austin for our collaboration and vital discussions that helped to shape part of this research. Moreover, I would like to thank Tamer El-Diraby of University of Toronto and his students for engaging me in research outside the scope of this thesis and an interesting collaboration.

During my PhD, I had the opportunity to obtain valuable industrial experience working twice, as a research intern at Yahoo! Labs, Barcelona. I would like to express my gratitude and appreciation to its director, Ricardo Baeza-Yates and research staff and collaborators Vanessa Murdock, Roelof van Zwol, Francesco Bonchi and Aristides Gionis. Part of this research stems from our interaction and invaluable discussions. My experience at Yahoo! Labs was tremendous, as it provided of an environment that requires thinking big of both problems and engineering solutions that can have impact in the lives of millions of users.

The Department of Computer Science at the University of Toronto offers a great academic environment that welcomes creativity, fosters the interchange of ideas and conversations, and encourages collaboration. Over the years, I have had the opportunity to interact with fellow graduate students, but also to teach in several different capacities (as a sessional instructor and teaching assistant). As a result, I have acquired extensive teaching experience in developing courses and educational materials, as well as, in mentoring and interacting with students. Transferring knowledge to others and experiencing the joy of making a clear difference in the lives of students, as they gain new insights and become more interested and knowledgeable in a subject was very rewarding. With this opportunity, I would like to thank the outstanding student body and fellow instructors for all their support and encouragement that helped me grow as an educator.

Accomplishing any significant aim or purpose requires finding the right work-life balance. I am glad that, over the years, I had the chance to meet many wonderful people outside the work environment. I wish I could list here all of them, as well as, a representative story of each one that proves how amazing they were, but have to limit myself to a few good friends that have stood the test of time and my awkwardness level: first and foremost my bosom friend Yannis, but also, Dimitris, Giorgos, Apostolis, Igor, Thomas, Tassos, Matei, Andrei; they have loaded this experience with unforgettable moments. Last but not least, I would like to thank a truly special and wonderful person, Angela, for her care and support, for sharing with me some of the most beautiful moments in life, and for bringing a smile to my face every day.

Looking back, I cannot but be grateful to my mentors and educators at the University of Crete, Greece and in particular Dimitris Plexousakis and Martin Doerr of FORTH-ICS, for their generosity and encouragement to take the next steps in research.

Finally, I would like to thank my close friends in Greece and around the World, my brother Sakis and my beloved parents Eleni and Ioannis, for their constant support and encouragement in all aspects of my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the last years, the web witnesses a prolific growth largely due to the changing trends in the use of web technology that aims to enhance interconnectivity, self-expression and information sharing. These trends have led to the development and evolution of virtual communities and services, such as social networking sites, photo and video sharing sites, blogs, wikis, and folksonomies, but also to the creation of voluminous user-generated data in form of text, images, videos and more. Even as these new online social media change the way we communicate, the social processes occurring remain governed by long-standing principles of human behavior and social interaction.

To date, analysis of human behavior and interactions has been limited to a small number of self-reporting individuals, due to practical issues. As such, models and algorithms have been typically based on simple network structures and limited data. On the other hand, social processes that take place in online spaces, can be monitored at unprecedented levels of scale and resolution, producing massive datasets. The capacity to collect and analyze the actions of millions of individuals at minute-by-minute time granularity, offers new perspectives on collective human behavior and forms a new research direction; one that builds theories of large social systems by examining their reflections in voluminous datasets [83].

The focus of our research is on the algorithmic and computational issues involved in the study of online social interactions and processes, such as, *behavior*, *influence* and *diffusion*. On the one hand is related to the formal understanding of principles and models of social interaction and processes that are persistent in online social media, and on the other hand is related to the design and development of algorithms and methods that aim to support social interactions and processes on the web. The nature of these problems requires an interdisciplinary research approach with questions and ideas flowing from both social and computing sciences.

Social sciences make available theories of social interactions and processes, while analysis of massive data sets informed by computational algorithms and models offer ample opportunities for research allowing to make progress on fundamental social science questions.

1.1 Abstract Setting

Sociologist Niklas Luhmann described society as a set of functionally differentiated sub-systems and suggested using systems analysis to disclose the structure and processes which characterize the most important of all systems (which includes all others) - the *social system* [116]. We consider a social system to be a *complex system* [23] defined through *social interactions* that occur among individuals (or groups of individuals). Social interactions that occur again and again in such systems lead to the formation of *social processes.* The recurrent form in which individuals express behaviors, establish relationships and communicate with each other are examples of social interactions and processes that now take place in *online social network services.* A social networking service is a web-based platform that enables the creation of social networks among people who share interests, activities, backgrounds or real-life connections and it typically consists of a representation of each individual (user) and her social connections (friends).

Central to our research is the notion of the *social graph.* In the context of graph theory, a *graph* is a mathematical structure used to model pairwise relations between objects. A graph $G(V, E)$ consists of a set of vertices (or nodes) V and a set of edges (or links) E. The graph can be *directed* meaning that the edges have a direction associated with them (from a source vertex to a target vertex), or *undirected* where there is no distinction between the two vertices associated with each edge. Social networks are examples of systems composed by a large number of interconnected individuals. Theregore, a common approach to capture the properties of such social systems is to model them as graphs whose nodes represent the *individuals* or *users* of the system, and whose edges stand for the *social connections* or *relationships* between them. We typically refer to the graph that models a social network as a *social graph* (Figure 1.1).

The concept of *social influence* is also essential in our research. Social influence refers to the change in behavior that one person causes in another, intentionally or unintentionally, as a result of the way the changed person perceives themselves in relationship to the influencer, other people and society in general. In the presence of social influence, an idea, norm of behavior, or a product *diffuses* through the social network like an epidemic. We assume that social influence occurs in an online social system when an individual's thoughts or actions are affected by one of his friends in the network.

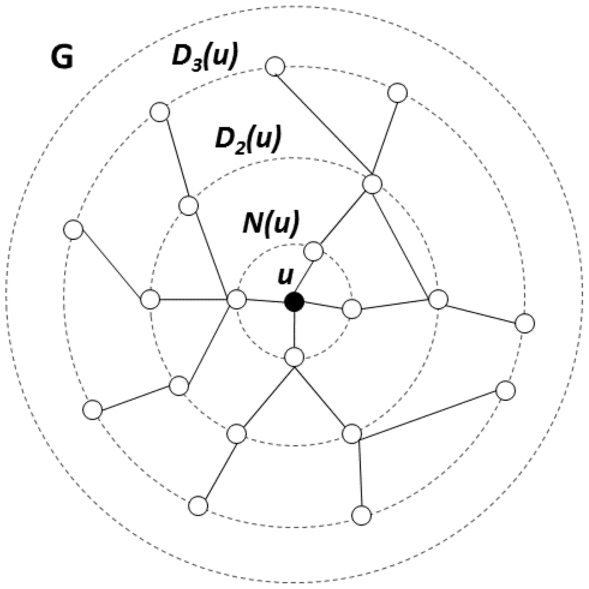


Figure 1.1: Example of a social graph $G(V, E)$. Vertices V represent individuals of a social network and edges E represent social connections between them. The notion of a neighborhood $N(v)$ and the vicinity $D_d(v)$ of a node v at depth d is also represented.

Meanwhile, large social networks serve as conduits for communication and the flow of information. Recent research work has primarily studied communication of an event-driven nature, looking at communication within a social system triggered by a particular event or activity that propagates through many nodes over short time-scales, forming *information cascades.* Examples include cascades of chain e-mails, cascades of references among bloggers, and more. These types of event-driven communication, however, take place against a much broader setting of natural communication rhythms, a kind of systemic communication that circulates information continuously through the network. Pairs of individuals communicate over time at different rates, for an enormous range of different reasons. Viewed cumulatively, structural patterns of the *topology of the network* form *information pathways* [85] that enable information to spread through the network efficiently. This type of systemic communication has remained essentially invisible in analyses of social systems. How these patterns affect the ability of a social network to carry on such social processes, how these patterns change over time and how to control the evolving social processes constitute interesting research questions.

Note that while there is a common abstract setting on which we operate, each of the following chapters deals with a significantly different problem and minimal notation is common across all chapters.

1.2 Outline

We provide an overview of each chapter below. We begin by providing definitions, terminology and the abstract setting upon which our research is conducted and the description of our technical contributions is based (**Chapter 1**).

Our initial research in this area was motivated by a broad class of research questions related to *how information spreads in blogs*. At a particular point in time, some users become aware of and post new information online and they have the potential to link it on to their friends and colleagues, causing the information to cascade through the blogosphere[1]. Our first objective was to establish evidence of communication occurring in the blogosphere. To this extent, we had to model information cascades happening in blogs and monitor them over a large period of time looking for interesting patterns and trends, such as characteristics of posts that are able to trigger cascades, the size and height of formed cascades, assessment of the engagement level and reaction times of users that participate in the online discussions. Our study analyzed one of the largest data sets available at the time (including 30M blogs and over 400M posts) and was distilled into a comprehensive analysis of how information cascades among blogs under various parameters and constraints, and contributed to a better understanding of the linking activity in the blogosphere (**Chapter 2**).

Analysis of massive data sets at an *aggregate level* raises interesting observations of information cascades within social systems, but it has its limitations as it typically does not allow for interpretations at the *individual level*. More importantly, it does not allow to argue about *the causality of the observed cascades*. Our next research was therefore driven by more refined research questions related to information cascades. *Why cascades happen? How to determine whether cascades will occur in a social system? How one can influence a cascade process?* Social scientists agree that in the presence of *social influence*, an idea or behavior diffuses through a social network like an epidemic. In our research, we were interested to develop methods that given a history of individual actions and social relationships of individuals can qualitatively and quantitatively detect social influence effects in a social system. Further, we developed a basic framework that allows to study the causality between individual actions of users and the social influence that they exert to their social network. As a case study for our research we employed a popular online social graph, Flickr[2] (including 525K nodes, 47M edges), and our analysis of social influence was based on the cascade of the geotagging behavior. To the best of our knowledge, our work was one of the first that tried to bridge the gap between aggregate level cascade processes and individual behavior using large data sets (**Chapter 3**).

[1]The blogosphere is made up of all blogs and their interconnections.
[2]www.flickr.com

Another broad class of computational questions in this area is related to the problem of *maximizing the information spread in a social network*, usually motivated by applications of viral marketing. A *critical problem* of the information propagation process is that while information spreads effectively among members of the same community (intra-cluster), it can eventually stall when it tries to break into other communities (inter-cluster). A similar problem occurs frequently when we try to sample nodes in a graph by performing *random walks over the graph.* In particular, a random walk might require to take a very large number of steps in order to retrieve a single sample. In our research we first made the observation that the ability of a network to carry on such social processes is a characteristic that depends, to a great extent, on the *topology of the network.* Therefore, slight modifications in the network topology, might have a dramatic effect on its connectivity and thus to its capacity to carry on such social processes. Then, we approached network modification as a *graph augmentation problem*, where we ask to find a small set of non-existing edges to add to the network that can optimize a connectivity property and give rise to more efficient social processes. In the augmented network, information propagates easier and faster and random walks are shorter, as they have to travel from a node to another by traversing fewer edges of a more connected network (**Chapter 4**).

Next, we turned our focus on designing and developing efficient methods that utilize information available in a social information network to better satisfy the *search needs of a user.* In particular, we described a *social search engine paradigm* which can be built on top of a classic search engine (such as Google, Bing, etc.) and an online social network (such as Google+, Facebook, etc.). Our objective was to design algorithms and develop methods to efficiently sample information available in a underlying social network and utilize it to re-order a list of URLs retrieved by querying a traditional search engine. The main goal and premise is to re-order the list of retrieved URLs in a way that favors URLs that are more relevant to a user's interest towards a *personalized search engine.* Search results obtained using this method can be superior compared to the results of the traditional search engines (**Chapter 5**).

Finally, we conclude the thesis and provide thoughts on possible future research directions (**Chapter 6**).

Each of the thesis chapters is self-contained and can be read in isolation (not in a particular order). It introduces a specific problem of interest, references related work, explains the proposed methods and algorithms, evaluates their performance theoretically and experimentally, and concludes with the important findings.

1.3 Research Questions

While we understand how information spreads offline, it is not always clear how information spreads online. A natural first research question is how we can model information spreading online and how to monitor and characterize it. Moreover, what the effect of heterogeneity in the information spread is and in particular how information cascades differentiate by topic or individual demographics. Once evidence of communication occurring online is established, further important questions remain about the causality of cascades: why information cascades happen and how to detect them in a social network setting? Note that research in this topic occurs at two operational levels: at the network-level and at the individual-level. An interesting research question is whether we can find points of convergence between these two operational levels that will allow to interpret interesting changes that occur on the network-level by observing changes of behaviors at the individual-level (and the other way around)? While understanding social processes is important, the main research questions of the thesis are focusing around the idea of improving online social interactions and processes. For example, how can we boost information propagation processes in social networks? Or, how can we facilitate a sampling process on a social graph? In addition, since social graphs are big and change dynamically there are important research questions related to efficient collection of information in networks. In particular, how can we exploit information that lies in social networks in order to improve on information search needs of the end user? These are all interesting and challenging research questions that motivate our research. The interested reader is encouraged to follow the motivation and discussion about the implications of our research found at the beginning of each chapter for a more comprehensive coverage.

1.4 Key Contributions

As the thesis is structured in a number of self-contained chapters that each deal with a significantly different problem, comprehensive conclusions of our research findings can be found at the end of each chapter. In this paragraph we highlight the main research contributions of the thesis.

- We conducted a large-scale analysis of the linking structure of blogs and established evidence of a cascading effect of online communication occurring online. Our study was the first to incorporate heterogeneity and revealed notable variations in the structural properties of cascades and in the ability of blogs to trigger cascades that mainly depend on the blogger's profile and the topic of a post.

- We designed a method that can both *detect* and *quantify* the occurrence of social influence in an online social system. The method is based on a randomization test that tests for the likelihood of a given type of behavior to appear in the data set, versus the null hypothesis, which states that the observed behavior has appeared purely by chance in a random set of observations. We demonstrated the use of the method to a popular online social network (Flickr[3]).

- We presented a method that allows to characterize the credibility of a user's expressed behavior in an online social network. In particular, a method that characterizes the *credibility* with which users place photos on an interactive map (geotagging) provided by Flickr. The method is using the collective wisdom of a large number of users expressing similar behaviors, and as such, relies heavily upon the wisdom of the crowd in assessing credibility scores that can be characterized by constant change.

- We presented an analytical and experimental framework that allows to investigate the *causality* between *changes of individual behavior* and *social influence* that individuals exert in their network. The framework can be utilized to find points of convergence between network-level analysis and individual-level analysis in an online social system.

- We modeled the information propagation problem in graphs as a graph augmentation problem and presented methods that can decide which new edges to add in the graph a multitude of times faster (x252 to x1078) than sensible baselines, without duly affecting the accuracy (less than 10%). In the augmented graph (i) information spread can reach 10% to 80% more nodes, and (ii) random walks can converge a multitude times faster (x4-x10), giving rise to faster graph simulations.

- Social graphs are big, dynamic, and embed content. We introduced sampling methods that use a variation of rejection sampling techniques to efficiently collect information from a social network. Our sampling methods are a multitude of times faster than crawling a friend network without duly affecting accuracy (less than 15%) and are much more accurate than naive sampling methods while introducing only slightly larger sampling cost.

Overall, the methods and algorithms we have presented are simple to understand and implement, but accurate, very fast and general, so they can probably be easily adopted in a variety of strategies. As such, we expect the thesis to be beneficial to diverse settings, disciplines, and applications ranging from social to technological networks.

[3]www.flickr.com

Chapter 2

Case Study: Analysis of Information Cascades in Blogs

2.1 Introduction

The process of ideas and practices spreading through a population, contagiously, with the dynamics of an epidemic, has long been of interest in social sciences. Its systematic study developed in the middle of the 20th century, into an area of sociology known as the *diffusion of innovations.* The theory of diffusion of innovations examines the effect of word of mouth and explores the role of social networks in influencing the spread of new ideas and behaviors. At a particular point in time, some nodes in the network, become aware of new ideas, technologies, fads, rumors, or gossip and they have the potential to pass them on to their friends and colleagues, causing the resulting behavior to cascade through the network. The initial research on this topic was empirical, but in the 1970s economists and mathematical sociologists began formulating basic mathematical models for the mechanisms by which ideas and behaviors diffuse through a population [134]. Two fundamental models for the process by which nodes adopt new ideas have been considered, the *threshold model* [60] and the *independent cascade model* [58]. Models of diffusion of innovations in a social network have since been considered in many disciplines including mathematical epidemiology, viral marketing, game theory (see [82] and the references therein).

More recently, the changing trends in the use of web technology that aims to enhance interconnectivity, self-expression and information sharing on the web has led to the development of online, many-to-many communication services such as blogs. A *blog* is a website with regular entries of commentary called *blog posts* or simply *posts*. The collective community of all blogs and their interconnections is known as the *blogosphere*. Information diffusion in technological networks raises interesting connections to the theoretical models [83]. As news and stories become available in the real-world, they spread in the blogosphere forming information cascades. The capacity to collect and analyze information cascades can be useful in various domains, such as providing insight on public opinion on a variety of topics [61] or developing better predictive models of the spread of ideas and behaviors [96]. It can also be useful in applications that make use of the diffusion process, such as in the problem of maximizing the spread of influence in a social network [75, 131] and the problem of early detecting outbreaks in a network [93].

The cascade process may be modeled by inferring links through textual similarity, as in [62] where authors focus on the diffusion of topics in blogs, or in [91] where authors track the news cycle through short, distinctive phrases that travel intact through online text. Cascades may as well be modeled through links among blog posts, such as in [1], where linking patterns of political blogs are explored, and in [86], where community-level behavior is analyzed in an online social network inferred from blog-roll links (links to affiliated blogs) among blogs. Our work is mostly related to work in [96], where authors study the temporal and topological patterns of cascades in large blog graphs based on link analysis. In that work, authors assumed spreading of information regardless of the *context* of the posts, and highlighted the need for further analysis that would be useful to the development of more accurate patterns of information propagation.

In this chapter, we present trends of the degree of engagement and reaction of bloggers in stories that become available in blogs under various parameters and constraints. To this end, we analyze cascades that are attributed to different population groups constrained by factors of *gender*, *age*, and *continent*. Then, we analyze how cascades differentiate in five subjects: *technology*, *politics*, *financial*, *sports*, and *entertainment*. In each case, we report on the structural properties of cascades and try to identify and quantify any variability in the cascading behavior. In our study we collected and analyzed information of cascading behavior that is available in *Blogscope* [17]. BlogScope is an analysis and visualization tool for the blogosphere and is currently tracking almost 36 million blogs with over 800 million posts making it one of the largest available datasets for our analysis.

2.2 Methodology

This section describes the methodology we follow to compute cascades. First, we introduce terminology and notation. Then, we present the datasets employed in the analysis and how they are collected. Finally, we present the observation measures on which cascades are compared.

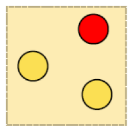
2.2.1 Preliminaries

Let $U(B, P)$ represent the blogosphere, where B is the set of all blogs and P the set of all posts. Each blog $b \in B$ consists of a set of posts $P^b \subseteq P$ (Figure 2.1(a)). Also assume that each post $p \in P$ is associated with a unique timestamp t_p that corresponds to the time of its submission, allowing for a total temporal ordering of the posts in P. Further, let $\ell_{p_y \to p_x}$ represent a link from a post p_y in the future to a post p_x in the past. For each link $\ell_{p_y \to p_x}$ we also define $\Delta_{p_x}^{p_y}$ to be the difference in the submission times of post p_y and p_x. Note that $\Delta_{p_x}^{p_y} = t_{p_y} - t_{p_x} > 0$.

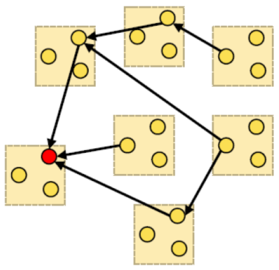
Now, let $G(P, L)$ be a graph where P is the set of all posts and L is the set of all links between posts. We call this graph the *post graph* (Figure 2.1(b)). Let $\hat{L}$ represent the set of all links in L but with reversed direction. Let also the graph $\hat{G}(P, \hat{L})$. A cascade $C(P^C, L^C)$ is an induced graph of the graph $\hat{G}$ where $P^C \subseteq P$ and $L^C \subseteq \hat{L}$ (Figure 2.1(c)).

A cascade C can be thought of as a directed graph, where nodes represent posts and edges represent information flow between posts. Note that the direction of an edge in C follows the information propagation (the actual permalink follows the opposite direction). We denote the in-degree of a node $\nu \in C$ as $deg^-(\nu)$ and its out-degree as $deg^+(\nu)$. A node $\nu \in C$ with $deg^-(\nu) = 0$ is called a *source* and a node $\nu \in C$ with $deg^+(\nu) = 0$ is called a *sink*. A cascade C has only one source, which represents the *initiator post* p_i. Any node $\nu \in C$ with $deg^-(\nu) > 1$ is called a *connector node* and represents a post that has permalinks to at least two posts in the past, and therefore connects branches of different cascades. Throughout the study we assume that a connector node participates (i.e., it is re-evaluated) in all the cascades that it connects. For each post p in the cascade C we define its reaction time R_p as the difference in the submission times of p and the initiator post p_i (i.e., $R_p = \Delta_{p_i}^{p}$). We define the following properties for a cascade C:

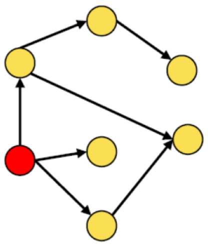
- *cascade size* (C_s): The number of nodes in C, excluding the initiator post p_i.
- *cascade height* (C_h): The height of the spanning tree obtained by traversing the cascade graph C using a *depth-first search* (DFS) algorithm. The algorithm starts at the source and at each step visits adjacent



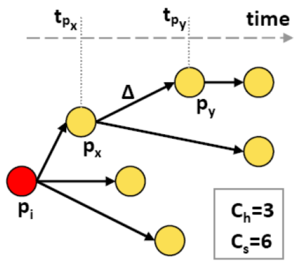
(a) *blog with posts*



(b) *post graph*



(c) *cascade*



(d) *cascade properties*

Figure 2.1: Example Cascade

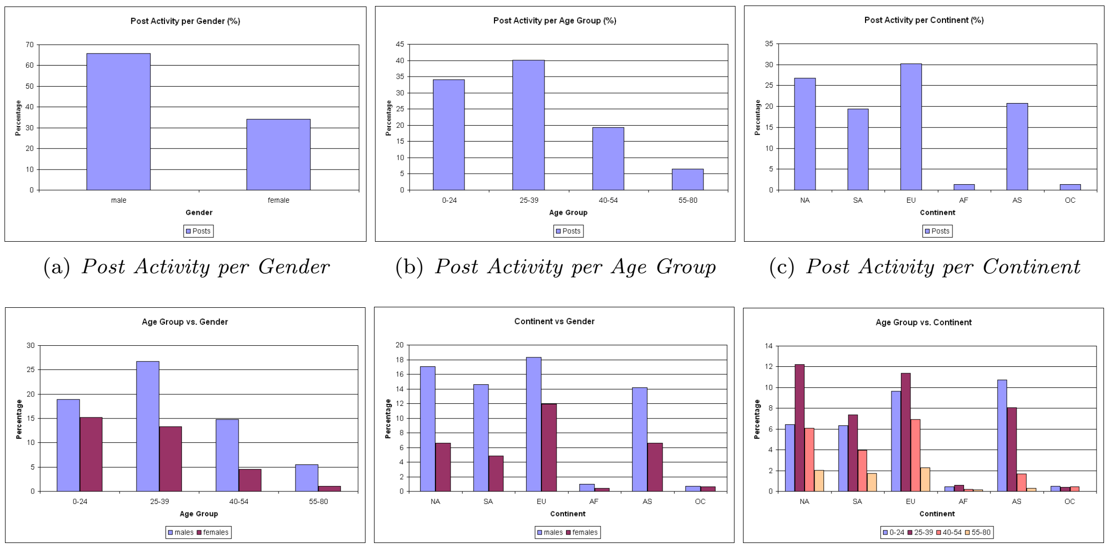
nodes giving priority to the node whose the post has smaller timestamp. The algorithm remembers previously visited nodes and will not revisit them (Figure 2.1(d)).

- *minimum reaction time* (R_{min}): The minimum reaction time of all posts in the cascade (excluding p_i).
- *mean reaction time* (R_{mean}): The mean reaction time of all posts in the cascade (excluding p_i).
- *maximum reaction time* (R_{max}): The maximum reaction time of all posts in the cascade (excluding p_i).

We consider a cascade C with $C_s = 0$ and $C_h = 0$ to be a *trivial cascade* (i.e., a single post). A *non-trivial cascade* C has $C_s \geq 1$, $C_h \geq 1$ and all links obey time order (i.e., $\Delta > 0$).

2.2.2 Dataset Description

In order to analyze the cascading behavior of posts under various parameters and constraints we abide by the following method. First, we formulate a sample dataset consisting of a set of posts satisfying the required



(a) *Post Activity per Gender* (b) *Post Activity per Age Group* (c) *Post Activity per Continent*

(d) *Age Group vs. Gender* (e) *Continent vs. Gender* (f) *Age Group vs. Continent*

Figure 2.2: Statistics by Population Factors

technology blogs
http://gizmodo.com
http://www.techcrunch.com
http://mashable.com
http://kotaku.com
http://www.readwriteweb.com
http://valleywag.com
http://slashgear.com
http://www.coolest-gadgets.com
http://searchengineland.com
http://i4u.com

politics blogs
http://www.dailykos.com
http://thinkprogress.org
http://corner.nationalreview.com
http://rightwingnews.com
http://instapundit.com
http://powerlineblog.com
http://www.redstate.com
http://firedoglake.com
http://digbysblog.blogspot.com
http://littlegreenfootballs.com/weblog

financial blogs
http://seekingalpha.com
http://www.bloggingstocks.com
http://www.247wallst.com
http://bespokeinvest.typepad.com
http://www.traderfeed.blogspot.com
http://globaleconomicanalysis.blogspot.com
http://calculatedrisk.blogspot.com
http://paul.kedrosky.com
http://billcara.com
http://www.getrichslowly.org/blog

sports blogs
http://outsidethebeltway.com
http://awfulannouncing.blogspot.com
http://draysbay.com
http://www.velonews.com
http://www.armchairgm.com
http://minorleagueball.com
http://mirtle.blogspot.com
http://bleedcubbieblue.com
http://homerderby.com
http://sportsmediawatch.blogspot.com

entertainment blogs
http://tmz.com
http://www.bestweekever.tv
http://justjared.buzznet.com
http://x17online.com
http://celebitchy.com
http://seriouslyomg.com
http://popbytes.com
http://theblemish.com
http://www.celebritysmackblog.com
http://www.dailystab.com

post distribution per subject	
subject	# Posts
technology	1916
politics	1649
financial	1104
sports	495
entertainment	1014

Table 2.1: Blogs per Subject

specifications. These posts serve as the initiator posts for the analysis. Then, we retrieve the cascades triggered by these posts by monitoring the blogosphere (i.e., approx. 36M blogs with 800M posts) for a specific time frame. The monitoring refers to the process of searching and retrieving all posts that have back links to the initiator post in the specified time frame. For each of the retrieved posts, we continue monitoring the blogosphere looking again for backlinks. This describes a recursive process that eventually computes a cascade (trivial or not-trivial) for each of the initiator posts. The recursion stops when there are not any backlinks to any of the intermediately retrieved posts. Note that all posts are allocated the same time frame to evolve, therefore there is no truncation occurring for posts later in the cascade. For the scope of our analysis we employed two sample datasets; one for analysis of cascades attributed to variable blog profiles and one for analysis of cascades of different subjects. Following we present details on each of the sample datasets.

Sample of Posts With Complete Profile

The study considers 160k posts submitted by 5000 blogs in the period starting on 01-Jun-2008 and ending on 10-Jun-2008. These blogs have *complete profile* information, meaning that the gender, age, and country of the blogger is defined (blogs in Blogscope are associated with a *blog profile*). For each post we monitor the blogosphere looking for cascades in a 30-days time frame starting at the post's submission day. We had a bias in favor of blogs with large number of posts. The final set consisted of blogs that had at least 17 posts in these 10 days with the most active ones having hundreds of posts. The average number of posts per blog in the dataset was 32. Figure 2.2 shows the posting activity of the blogs analyzed by factors of gender, age, and continent.

Sample of Posts in Different Subjects

The study considers around 6000 posts distributed in five subjects: *technology*, *politics*, *financial*, *sports*, and *entertainment*. For each subject we manually collect a number of representative blogs. Then, for each blog we obtain a set of posts submitted between 01-Jun-2008 and 10-Jun-2008. For each post we monitor the blogosphere looking for cascades in a 30-days time frame starting at the post's submission day. We had a bias in favor of more authoritative blogs taking into account the number of inlinks to a blog in the last year. The higher the number, the more authoritative (trusted) a blog is. A similar notion of authority has been used by other popular blog services. Table 2.1 provides further information on the blogs analyzed.

2.2.3 Observation Measures

We present measures that characterize the cascading behavior of a set of posts S. Let $S_C \subseteq S$ represent the set of posts in S that were able to trigger non-trivial cascades. Formally, we define the *cascade triggering ability* A_S of a set of posts S to be the ratio of $|S_C|$ over $|S|$:

$$A_S = \frac{|S_C|}{|S|} \tag{2.1}$$

Each post in S_C corresponds to an initiator post and forms a non-trivial cascade with properties of size, height, as well as minimum, mean and maximum reaction time (C_s, C_h, R_{min}, R_{mean}, R_{max} respectively). Since these properties typically have very skewed distributions, we report in our analysis the *medians* of their distributions in order to characterize the behavior of a set of cascades. (i.e., the median of the cascade sizes, the median of the cascade heights, the median of the minimum, the mean, and the maximum reaction times.). Throughout our analysis, we analyse cascading behavior based on these observation measures.

2.3 Analysis

The raw data obtained from the cascades requires a quality check in order to reject potential wrong cascades and avoid the existence of possible deviations in results. In particular, the quality check takes the following into account:

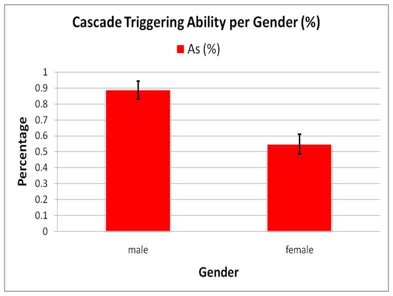
- *Closed-world Assumption*: We make a closed-world assumption that out-links to posts are valid only if they are out-links to posts in the dataset.

- *Total-ordering of Posts Assumption*: We assume that there is a total-ordering of the posts in the database based on the timestamps of the posts. This assumption allows to ignore links from a post in the past to a post in the future. Such links are infrequent but resident in the dataset and are mostly due to post updates or differences in time-zones.

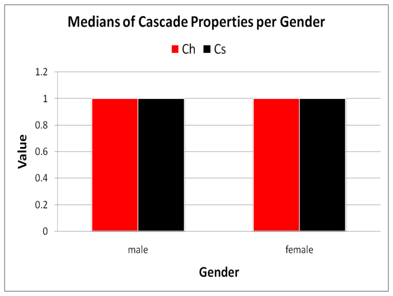
- *Self-links Removal*: We are interested in information cascades in a blog level. Links between posts of the same blog are considered self-links and are eliminated from the study.

- *Spam and Duplicate Post Detection and Elimination*: We make use of the Blogscope spam analyzer to filter out posts that are likely to be spam. We also detect and eliminate any duplicate posts from the datasets.

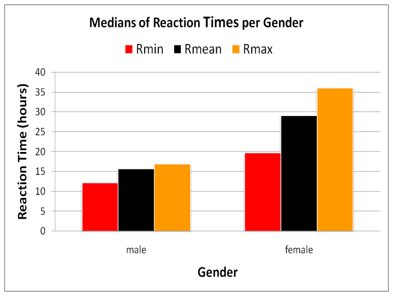
Once the wrong cascades are rejected, the evaluation measures are computed on the clean dataset. When we report on the cascading behavior ability A_s of a sample, the confidence intervals at the 95% confidence level are also stated (error bars on graphs). As aforementioned, medians are reported for the rest of the observation measures.



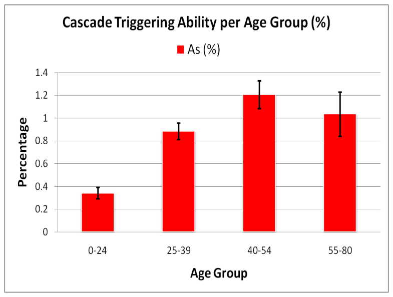
(a) *Cascade Triggering Ability*



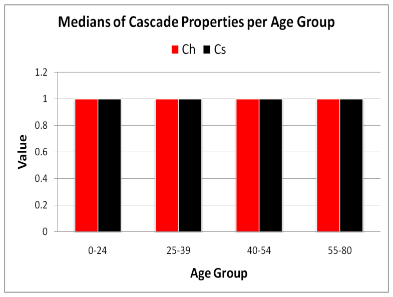
(b) *Cascade Properties*



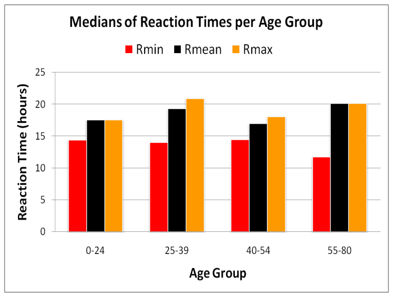
(c) *Reaction Times*



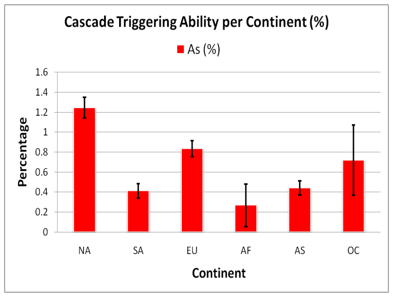
(d) *Cascade Triggering Ability*



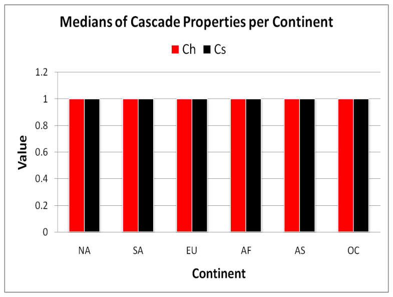
(e) *Cascade Properties*



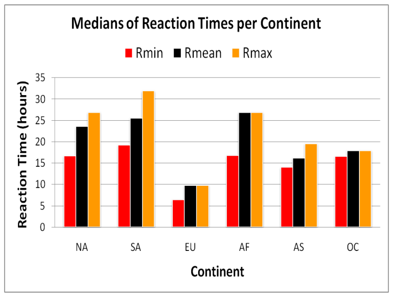
(f) *Reaction Times*



(g) *Cascade Triggering Ability*



(h) *Cascade Properties*



(i) *Reaction Times*

Figure 2.3: Cascades per Gender (a, b, c), Age Group (d, e, f), and Continent (g, h, i)

2.3.1 Analysis of Cascades by Population Group

The cascades obtained allow the comparison of cascading behavior among different population groups. The analysis first compares cascades of posts submitted by bloggers of different gender (male, female). It is then extended to compare the situation in age groups (0-24, 25-39, 40-54, 55-80) and continents (North America (NA), South America (SA), Europe (EU), Africa (AF), Asia (AS), Oceania (OC)).

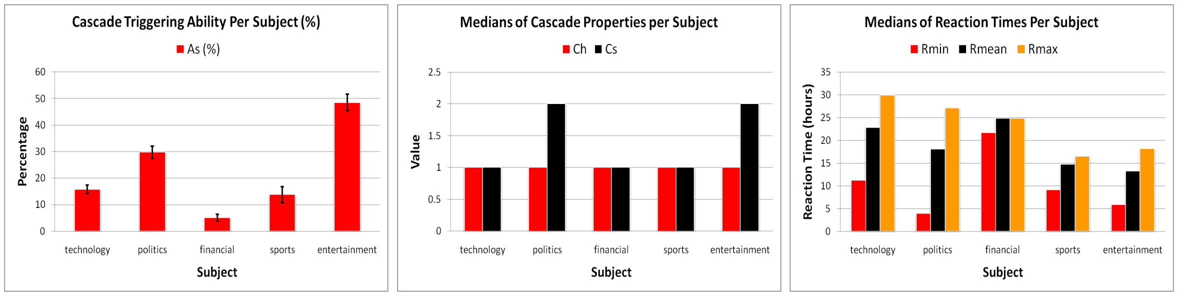
Cascades per Gender: Figure 2.3(a) highlights that there are notable variations on the cascade triggering ability (A_S) between male posts (0.9%) and female posts (0.5%). These cascades are typically small and short ($C_s = 1$, $C_h = 1$) in both male and female posts (Figure 2.3(b)). However, male posts exhibit shorter reaction times in the blogosphere, but their discussions appear to be more *ephemeral* (i.e., finish soon after they start) compared to the female posts that last more (Figure 2.3(c)).

Cascades per Age Group: Figure 2.3(d) highlights that there are notable variations on the cascade triggering ability (A_S) among people of different age groups. The blogosphere seems to mostly engage in cascades triggered by posts of people in the age group of 40-54 (1.2%) and 55-80 (1.1%) and less in posts of people in age groups of 0-24 (0.3%) and 25-39 (0.9%). However, cascades in all age groups are typically small and short ($C_s = 1$, $C_h = 1$) (Figure 2.3(e)). Finally, the reaction time of the blogosphere to senior posts is shorter than to posts from younger and discussions generally last more (Figure 2.3(f)). It is also evident that discussions on posts coming from people in the 0-24 age group are late to start and ephemeral.

Cascades per Continent: Figure 2.3(g) highlights that there are notable variations on the cascade triggering ability (A_S) of posts among continents. People appear to engage more in posts coming from NA, followed by EU, OC, AS, SA, and AF with values 1.2%, 0.8%, 0.7%, 0.4%, 0.4%, and 0.3% respectively. However, cascades attributed to all continents are typically small and short ($C_s = 1$, $C_h = 1$) (Figure 2.3(h)). Finally, the reaction times vary slightly among continents with the exception of EU posts that exhibit reactions that are immediate, but very ephemeral (Figure 2.3(i)). On the other hand reaction times in NA, SA and AF are not that timely but discussions last longer.

2.3.2 Analysis of Cascades by Subject

The cascades obtained allow the comparison of cascading behavior of posts in relation to their subject that varies among technology, entertainment, sports, financial, and politics. Figure 2.4(a) highlights that there are notable variations on the cascade triggering ability (A_S) of posts depending on their subject. Entertainment posts are much more likely to trigger cascades (50%). Politics posts have also a high probability (30%). On the other hand, financial posts rarely trigger cascades (only 5%). Technology and sports posts fall somewhere in the middle with 15%, and 13% respectively. Note that this trend appears to be independent to the number of posts in each subject. For example, even if there are almost as many financial posts as entertainment posts in our sample, the latter are 10 times more likely to launch cascades. In all subjects cascades are typically small and short ($C_s = 1$, $C_h = 1$) (Figure 2.4(b)). However, there are variations in the reaction times of the cascades (Figure 2.4(c)). The blogosphere seems to be more reactive to politics, sports and entertainment posts as indicated by the corresponding R_{min} values. However, politics posts have much larger R_{max} values which indicates that they continue to occupy the blogosphere for a longer period. This



(a) *Cascade Triggering Ability* (b) *Cascade Properties* (c) *Reaction Times*

Figure 2.4: Cascades per Subject

is also true for the technology posts. On the other hand, sports, financial and entertainment posts appear to be more ephemeral.

2.4 Validation

Thoughout our study we based our observations on sample populations. But, how dependent our observations on these samples are? Statistical inference allows to assess evidence in favor of or against some claim about the population from which the sample has been drawn. The methods of inference we use to support or reject our claims based on sample data are known as *tests of significance*.

2.4.1 Tests of Significance

The first step in testing for statistical significance is to postulate a *null hypothesis* H_0 and a *research hypothesis* H_a. For the case of k-samples, a null hypothesis usually states that samples are not significantly different (i.e., they come from the same population). To test whether the null hypothesis should be rejected or accepted, we perform, for a selected probability of error level (alpha level), a *statistical significance test* that compares the k-samples. If the null hypothesis is rejected, indicating that at least two samples are significantly different, we then need to perform a *post-hoc analysis* to determine which of the k-samples are different. Due to the nature of the quantities that we are reporting in our study we employ two types of significance tests.

Tests for k Proportions

The cascade triggering ability A_s is a proportion that measures, on a [0, 1] scale, how many of the posts in a sample set are able to trigger non-trivial cascades (see Equation 2.1). The statistical test we use to compare k proportions is known as a Chi-square test. Since the Chi-square test is an asymptotical test (i.e., vulnerable to errors with small values), we actually run the simulations based test (Monte Carlo test) with 5000 simulations. For the various cases in our study, we formally define the following hypotheses:

H_0^A: *The proportions are not significantly different*

H_a^A: *At least one proportion is significantly different* from another

In the case that the null hypothesis is rejected by the test, we perform post-hoc analysis using the Marascuilo procedure to identify which of the series differ. The Marascuilo procedure simultaneously tests the differences of all pairs of proportions for the several samples under investigation.

Non-parametric Tests for k Independent Samples

The observation measures C_s, C_h, R_{min}, R_{mean}, R_{max} are all real numbers that describe a cascade. The values of these properties typically have skewed distributions. Thus, to compare k-samples of one of these properties we employ a method that does not assume a normal population, such as the Kruskal-Wallis test. This test is a non-parametric method for testing equality of population medians among groups. Intuitively, it is identical to a one-way analysis of variance with the data replaced by their ranks. For the various cases in our study, we formally define the following hypotheses:

H_0^B: *The samples are not significantly different*

H_a^B: *The samples do not come from the same population*

In the case that the null hypothesis is rejected by the Kruskal-Wallis test, we perform post-hoc analysis to identify which of the series differ. We use the bonferroni adjustments post-hoc test to identify these pairs.

2.4.2 Results

Cascades by Population Group

We performed statistical significance tests for each of the various segmentations of the population in our study (i.e., Gender, Age Group, Continent). For each segmentation, the computed p-value of the Chi-square test was lower than the significance level $alpha = 0.05$, so one should reject the null hypothesis H_0^A and accept the alternative H_a^A. Therefore, we can conclude that a blogger's profile (each of gender, age, and continent) can significantly differentiate the cascade triggering ability A_s of a post. Moreover, for each population segmentation, the post-hoc analysis showed that all pairs are significantly different except for the pairs (25-39, 55-80) and (40-54, 55-80) regarding the age group segmentation and the pairs (NA-OC), (SA-AF), (SA-AS), (SA-OC), (EU-OC), (AF-AS), (AF-OC), (AS-OC) regarding the continent segmentation.

For the case of C_s, C_h, R_{min}, R_{mean}, R_{max}, the Kruskal-Wallis test showed that the samples do not come from the same population, thus rejecting the null hypothesis H_0^B. The only exception was in the reaction times of the age group segmentation where the null hypothesis was accepted, indicating no significant difference in the reaction times of posts coming from diverse age groups. The post-hoc analysis revealed the set of different pairs in each case. We omit details on pairs due to space limitations.

Cascades by Subject

We performed statistical significance tests for the cascades of different subjects in our study. The computed p-value of the Chi-square test was lower than the significance level $alpha = 0.05$, so one should reject the null hypothesis H_0^A and accept the alternative H_a^A. Therefore, we can conclude that the subject of a post can significantly differentiate its cascade triggering ability A_s. Moreover, the post-hoc analysis showed that only the pair (technology, sports) is not significantly different, while all other pairs are.

For the case of C_s, C_h, R_{min}, R_{mean}, R_{max}, the Kruskal-Wallis test showed that the samples do not come from the same population, thus rejecting the null hypothesis H_0^B. The post-hoc analysis revealed the set of different pairs in each case. We omit details on pairs due to space limitations.

2.5 Discussion

Analysis of the cascading behavior of blogs is a challenging research topic. We focused on the effect of incorporating heterogeneity into the analysis of information cascades in the blogosphere. Our analysis revealed notable variations of the cascading behavior depending on (a) the blogger's profile and (b) the subject of a post. More specifically:

- **cascade triggering ability**: Posts are more likely to trigger cascades if they are coming from males than from females, if they are submitted by middle-agers or seniors than younger, if they are related to entertainment or politics as opposed to sports, technology, or finance.

- **cascade size and height**: Structural properties of cascades, such as size and height, follow power law distributions with only a few cascades being large and deep. A typical cascade is small and shallow (i.e., $C_h = 1$, $C_s = 1$). Practically, this is a sign of limited discussion taking place in the blogosphere.
- **reaction times**: Reaction times are shorter for posts submitted by males, where the author is middled-aged or senior, or the post relates to politics or entertainment.

The issue that different bloggers present different potential to trigger cascades provides useful insights for the development of better prediction models. In particular, microscopic analysis of blogger behaviors could serve as the basis for analyzing macroscopic properties of networks, such as prediction models of the spread of ideas and methods to measure influence among blogs. It might also be useful in the development of tools that make use of the diffusion process (e.g., in fields of epidemiology and viral marketing).

Regarding the subject of a post, the issue that different topics present different background requirements could explain some of the variability of the cascading behavior we observe. For example, one would expect that entertainment posts are more likely to trigger cascades than financial posts, since it is much easier to discuss entertainment than finance topics. It may also be relevant that different user groups are associated with different topics. The latent semantics of these values and how they correlate is intricate. However, it is important that our work not only clearly identifies this variability, but also tries to quantify it.

Chapter 3

Individual Behavior and Social Influence in Online Social Networks

3.1 Introduction

Over the last several years the Web witnessed a prolific growth largely due to the changing trends in the use of Web 2.0 technology that aims to enhance interconnectivity, self-expression and information sharing. These trends have led to the development and evolution of virtual communities and services, such as social networking sites, photo and video sharing services, blogs, wikis, and folksonomies, but also to the creation of voluminous user-generated data in the form of text, images, videos and more. Even though these new social media change the way we communicate, the underlying social processes remain governed by long-standing principles of human behavior and social interaction.

To date, analysis of human behavior and interaction has been limited to a small number of self-reporting individuals, due to practical issues. As such, models and algorithms typically have been based on a small number of temporary snapshots of the network structure and data. On the other hand, social processes that take place in online spaces can be monitored at unprecedented levels of scale and resolution, producing massive datasets. The capacity to collect and analyze the actions of millions of individuals at a minute-by-minute time granularity offers new perspectives on collective human behavior.

Macroscopic analyses of massive datasets raise interesting observations of temporal patterns of communication within social systems. The way we form connections, imitate actions, follow suggestions, influence others in forming opinions or making decisions, as well as the spread of behavior and the information flow in groups, describe dynamics and phenomena of everyday life, now expressed in an online setting. However, working at a large scale has its own limitations. Monitoring social activity in an aggregate fashion typically does not allow for interpretations on a microscopic level. Questions of how influential an individual is and what makes her more (or less) influential among her neighbors to a large extent remain disputed and open. Leveraging the interplay between macroscopic and microscopic worlds of online social processes and being able to find points of convergence between them is a challenge [83] and the main focus of this research.

In this work, we examine how different types of individual behavior affect the decisions of friends in a network. We begin with the problem of detecting social influence in a social system. In the presence of social influence, an idea, behavior norm, or product can diffuse through the social network like an epidemic. Thus being able to identify situations where social influence prevails in a social system is important. Our analysis of social influence is based on the diffusion of a technological innovation. Then, we focus on two types of behavior expression, one that characterizes the *quantity* property (i.e., how often) and one that characterizes the *quality* property (i.e., how well) of the expressed behavior.

Detecting social influence derives from macroscopic analysis of aggregated data, while characterization of user behavior derives from microscopic analysis of user-specific actions. This setting allows to investigate causality between individual behavior and social influence in a principal way. In particular, we observe the diffusion of an innovation among social peers and try to identify what is the effect of particular behaviors in the social influence that individuals exert to their network. This raises several research questions: Are more active users more influential? Are more accurate users more influential? How do we obtain and evaluate this information?

Our analysis of social influence is based on the adoption of the *geotagging innovation* in Flickr[1]. *Geotagging* refers to the process of adding geographical identification metadata to uploaded photographs, usually consisting of latitude and longitude coordinates, by placing them on a map. In our analysis, we need to look closer to individual actions and monitor how systematically and how accurately they use the innovation. We carefully design the experiments around Flickr and its geotagging innovation as it provides information on user actions in an adequate degree of detail for carrying out such research. Given this environment we make the following contributions:

- We present a method that given a social network and a log of user actions over a time period, detects and quantifies the occurrence of social influence among peers. We apply this method to show that

[1]www.flickr.com visited February 2010

adopting and using the geotagging innovation in Flickr does not happen randomly, but can to a large extent be attributed to social influence among users in the network (Section 3.3).

- We present a method that allows to characterize the *quality* property of a user's behavior by evaluating the accuracy with which they use the innovation - a measure of *user credibility.* (Section 3.4).
- We hypothesize there is a relation between a user's social influence and specific expressions of individual behavior and design experiments to test the hypotheses (Section 3.5). Our findings reveal an essential gap on the effect that these types of behavior have on influencing other people in their network and suggest design opportunities for online social systems.

The basic issue which we try to deal with in this research is the way in which an individual's choices depend on what other people do. Our work suggests an experimental framework of investigating causality of individual behavior and social influence in a network, while the methods we describe make a relatively small number of assumptions about the data, are general and can probably be easily adapted to other analysis of social systems.

3.2 Related Work

The process of ideas and practices spreading through a population contagiously, with the dynamics of an epidemic, has long been of interest in the social sciences [148]. Its systematic study developed in the middle of the 20th century into an area of sociology known as the *diffusion of innovations* [132]. The theory of diffusion of innovations examines the effect of word-of-mouth information sharing and explores the role of social networks in influencing the spread of new ideas and behaviors. At a particular point in time, some nodes in the network become aware of new ideas, technologies, fads, rumors, or gossip, and they have the potential to pass them on to their friends and colleagues, causing the resulting behavior to cascade through the network. Models of diffusion of innovations in a social network have since been considered in many disciplines including mathematical sociology [58, 60], mathematical epidemiology, viral marketing, and game theory (see Kleinberg [43, 82] and the references therein). Diffusion of innovations is usually related to the problem of finding influential nodes in a network [76, 93, 131].

More recently, information diffusion in technological networks raises interesting connections to theoretical models [83]. As news and stories become available in the real-world, they spread online forming information cascades. The capacity to collect and analyze information cascades can be useful in various domains, such as providing insight into public opinion on a variety of topics [1, 61, 124] or developing better predictive models of the spread of ideas and behaviors [2, 62, 91, 96].

Other researchers have investigated forms of cascading effects in the Flickr social graph. Singla et Weber [137] monitor the phenomenon of brand congruence in Flickr for hundreds of thousands of users and over a period of two years. Among other observations, their study suggests that two friends have a higher probability of being using the same brand, compared to two random users, suggesting the existence of social influence effects. Che et al. [28], study the distribution of photo popularity in the Flickr social graph by monitoring the favoring mechanism (users can flag photos as favorite). Their study suggests that information spreading is limited to individuals who are within close proximity of the uploaders, suggesting that social influence happens on the level of direct friends.

Characterizing the relationship between user behavior and their social environment is also the focus of other research work recently. Christakis and Fowler [66, 115] investigate cascading effects of individual behavior in human social networks. Online, Singla and Richardson [138] focus on Instant Messenger interactions and try to characterize the relationship between a person's social group and its personal behavior. They apply data mining techniques to quantify how similarity is altered based on various attributes, such as communication time with another user and more, while in [35], the authors try to quantify how social interactions affect personal interests and vice versa. A similar approach has been taken in the work of Leskovec et al. [92] where the focus was on node arrival and edge creation actions that collectively lead to macroscopic properties of networks and in [90], where authors study the diffusion of recommendations in a network by controlling a single threshold value that determines whether a user will forward a recommendation. Characterizing individual behavior in social systems has also been the topic of interest in [133], but authors do not investigate the relation of behavior and social influence. Our research complements these works in that we are interested in investigating the causality of social influence in social systems due to individual behavior, thus exploring the interplay between macroscopic and microscopic properties in the diffusion process.

3.3 Detecting Social Influence

Identifying situations where social influence is present in a social system is important. In the presence of social influence, an idea, norm of behavior, or a product diffuses through the social network like an epidemic. We assume that social influence occurs in a social system when an individual's thoughts or actions are affected by one of his friends in the network.

Formally, assume a directed graph $G(V, E)$ where nodes V represent users and edges E represent friendships among users. Suppose user a adopts an innovation at time t_1. We say that user a influences user b if and only if at time t_2 when user b adopts the innovation, user a has already adopted it at an earlier time t_1, at which time a and b were already friends. We therefore assume that social influence occurs when the information of a friend adopting the innovation has the *potential* to flow to neighboring nodes in the social network.

Note that we narrow the definition of social influence to the special case where a user has at least one friend in the network that has previously adopted the innovation. We show that we can make this assumption without any loss of generality and can be generalized to more strict definitions of social influence. Our definition of social influence does not aim to be universal, but rather to help distinguish the existence of some sort of social influence from randomness in the process of adopting an innovation in a social system.

3.3.1 Methodology

We would like to be able to reason about whether - and if yes, in what extent - social influence occurs during the adoption of an innovation in a social system. Depending on whether they have adopted the innovation or not, nodes in the social graph $G(V, E)$ are distinguished between *active* or *inactive* respectively. We monitor the adoption of the innovation during a period $[0, T]$ and assume that activation is a binary decision, thus a node in the network that adopts the innovation remains active for the rest of the period, otherwise it remains inactive. At time $t = 0$ all nodes are inactive, while at the end of the monitoring period, $t = T$, a set A of k nodes in the network have been activated.

We now describe a method for determining if the activation of the nodes can be attributed to social influence. The idea is to test whether the activation of nodes over the monitoring period $[0, T]$ happens randomly or is affected by earlier activation of neighboring nodes in the network. If the latter is true, then we can assume that some social influence takes place, and therefore nodes that have already active neighbors have a higher probability of being activated. Note that the time of activations is important in determining social influence. Let $A = \{v_1, v_2, ..., v_k\}$ be the set of k users that are activated during the period $[0, T]$ and assume that user v_i is activated at time t_i. Since activations happen in discrete time, a total ordering of all k activation times is possible. To determine whether there is a social influence, we run the following test, called the *shuffle test*, which consists of two steps.

- **Step 1 (Original)**: We observe the adoption of the innovation in the social graph in the time period

$[0, T]$. For each node activation we determine whether it can be attributed to social influence according to our definition of social influence (i.e., at least one friend is already active). Let $A_{SI} \subseteq A$ denote the set of activations that can be attributed to social influence. Then the total effect of social influence $SI_{original}$ in adopting the innovation is given by:

$$SI_{original} = \frac{A_{SI}}{A}$$

- **Step 2 (Shuffled)**: Next we create a second problem instance with the same graph G and the same set A of active nodes, by picking a random permutation π of $\{1, 2, ..., k\}$ and setting the time of activation of node v_i to $t'_i := t_{\pi(i)}$. Again we observe the adoption of the innovation in the period $[0, T]$ as in Step 1, and determine the new set $A'_{SI} \subseteq A$ of activations that can be attributed to social influence and the total effect of social influence $SI_{shuffled}$ in adopting the innovation is given by:

$$SI_{shuffled} = \frac{A'_{SI}}{A}$$

If $SI_{original} > SI_{shuffled}$ then we can conclude that some sort of social influence has been detected in the adoption of the innovation. Note that we can monitor the effect of social influence at any specific time $t \in [0, T]$ (or when a specific number of activations has occurred) by comparing the number of activations that are due to social influence as opposed to random activations by the following formula:

$$SI^t_{t \in [0,T]} = SI^t_{original} - SI^t_{shuffled}$$

The test is based on the idea that if social influence plays an important role in adopting an innovation, then the timing of activations is not independent but rather is affected by the number of already activated neighboring nodes in the network. The idea of using timestamp permutations to distinguish influence from correlation in a social network hinges on the shuffle test suggested in Anagnostopoulos and al. [9]. However, our method does not make any assumption about a priori knowledge of the distribution of each node's influence over its neighbors (this information cannot be assumed to be known in a real system) and does not need to simulate a theoretical cascade model to determine which nodes are eventually activated. A similar randomization technique has been used by La Fond and Neville [87] to measure the gain that is due to influence and/or homophily in a social network. Their method assumes that users have attributes (e.g.,

Table 3.1: Contacts Per User

	min	max	avg	stddev	median
indegree	1	27404	89.2	270	15
outdegree	1	19542	99.9	309	19

Table 3.2: Contacts Per Geotagger

	min	max	avg	stddev	median
indegree	1	14664	104	277.4	28
outdegree	1	14432	104	299.2	30

Table 3.3: Geotagged Photos Per Geotagger

min	max	avg	stddev	median
1	92782	144	580	21

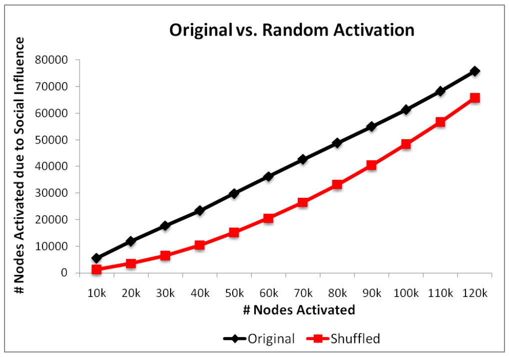
age or gender) that are assumed to be known. Das et al. [36] describe sampling based methods to efficiently collect such information from a social graph. Again, our method does not make any assumption about node attributes, and as such, it is simpler, more practical and designed to efficiently be applied to other online social systems where we would like to both *detect* and *quantify* the existence of social influence.

3.3.2 Adoption of the Geotagging Innovation in Flickr

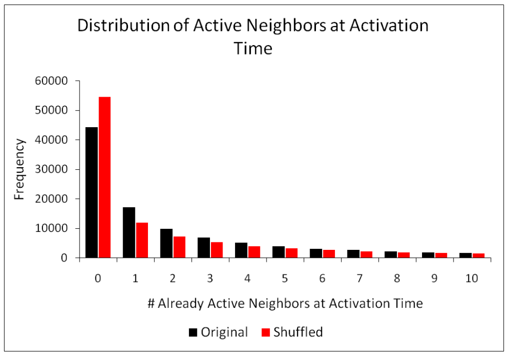
We apply our method of *detecting social influence* in Flickr. Flickr is a popular online social system centered around photo sharing, where users upload and share photos, comment on and tag their own photos and the photos of others, establish "friend" links and join groups of other users. Our analysis of social influence is based on the adoption of the *geotagging innovation* in Flickr.

Data

We crawled a large social network of the Flickr graph. To guarantee that our data includes a large number of users that have adopted the geotagging innovation, we started with a seed set of the 100 most active geotaggers and collected their contacts, and their contacts' contacts. Our final social graph consists of 525,000 nodes and 47 million directed edges. (Note that friendships are not reciprocal in Flickr.) From these users, 120,000 users have used the geotagging activity at least once (geotaggers). Table 3.1 gives descriptive information about the network structure for all users, and Table 3.2 for geotaggers only. Moreover, Table 3.3 provides information about the distribution of photos per geotagger. There are approximately 13 million geotagged photos in our data.



(a) *At least 1 active neighbor*



(b) *Activation Histogram*

Figure 3.1: Original vs. Shuffled Activations

We run the shuffle test on the Flickr social graph and observe the diffusion of the geotagging innovation (i.e., node activations) for the original and the shuffled timestamps. Figure 3.1(a) shows that for a fixed number of activations, the number of nodes activated at a given timestamp is larger in the original timestamps than the shuffled ones. Therefore we attribute adoption of the geotagging innovation to the social influence of users. This effect is more obvious in the beginning of the diffusion process where the chance of having a random active node as a neighbor is smaller. As the process unfolds the network is filled with more and more active nodes. This increases the chance of having at least one active neighbor node and eventually causes the process to saturate. Saturation occurs when all subsequent activations can be attributed to social influence in both the original and the shuffled scenarios (i.e., upper right part of Figure 3.1(a)).

So far we have constrained our definition of social influence to having at least one active neighbor at the time of activation. We examine how this is affected by restricting the definition of social influence to having more active neighbors at the time of activation. To generalize on this observation, we show in Figure 3.1(b) the distribution of the number of already active nodes at the time that a node is activated for both the original and the shuffled timestamps. Note that in the case of the shuffled timestamps the number of nodes that have no other active neighbor at the time of activation (the initiators) is larger than that of the original timestamps. This alone means that initiators are distributed more uniformly in the graph in the case of the shuffled timestamps than in the original timestamps. This is a first indication that activations are not random. Moreover, note that for all the subsequent cases (i.e., at-least-1, at-least-2, at-least-3, ..., at-least-10 active nodes at the time of activation) the number of active neighbors is always larger in the case of the original than in the shuffled scenario. The percentage of the additional active neighbors ranges between 44.7% in the at-least-1 case, to approximately 10% in the at-least-10 case. Therefore, even for more restricted definitions of social influence, there always exist an essential part of activations that may be attributed to social influence as opposed to random activations.

From this analysis we conclude that the diffusion of the geotagging innovation in Flickr (i.e., users start geotagging photos) is not random but can to a large extent be attributed to social influence of neighboring users in the Flickr social network. This observation provides evidence of a cascading process that takes place in the social graph; a process of users passing to their neighbors a signal of access to the geotagging innovation.

3.4 User Geotagging Credibility

We have seen evidence that users are influenced by their friends to start geotagging photos. In this Section, we develop a methodology that allows to characterize the *quality* of the geotagging behavior of users. As a surrogate of quality we define a credibility metric, based on the accuracy with which users make use of the geotagging innovation. Note that our data contains two types of geotags; either coming from GPS-enabled devices or manually assigned by a user. GPS-enabled devices give very accurate coordinates. But, to develop a user credibility metric we are more interested in the second type of geotags, which are determined by the system when a user manually places a photo on the Earth map.

The next paragraphs provide details of our methodology. In brief, it involves steps that allow to determine with high probability the location that was intented by the user (i.e., the target location) and then to measure the geodesic distance between the actual location (where the photo was placed on the earth map) and the target location. More specifically, we try to associate each user v in the network with a credibility value c_v that expresses the accuracy with which she geotags photos. Eventually, the method allows to compare a user's geotagging accuracy to the accuracy of other users and to determine who is more or less credible. As such, the method provides a primitive way to assess and argue about the *quality* of a user's expressed behavior.

Part of this work is performed by using freely available services of Yahoo! Geo Technologies [150], such as Yahoo! PlaceMaker and Yahoo! GeoPlanet.

3.4.1 Distinguishing Location Tags from Free-text Tags

Each photo in the dataset is associated with a set of free-text tags or tags, which are non-hierarchical keywords assigned informally to a photo by its owner. This metadata provides clues to both the context and the content of the photo and allows it to be found by browsing or searching. Tags often refer to geographical places. Throughout the chapter, we refer to these tags as *location tags*.

To identify location tags, we utilize Yahoo! Placemaker, a geoparsing Web service. Provided with text, the service returns the probability that the text is a placename (e.g, the probability of the word “altitude” or “urban area” to be a placename is 0.077 and 0.055 respectively, while the probability of the word “lake echo” or “quebec city” to be a placename is 0.71 and 0.89 respectively). The Yahoo! Placemaker service takes care of the data cleansing process and considers variants of placenames in the identification process. Furthermore, the probability model (text is recognized as place with a certain probability) provides flexibility in which tags to accept as location tags, by filtering out those that are below a threshold θ.

For each photo we compute the set of its location tags by scanning the set of its tags and omitting those that are unlikely to refer to geographical places. For our needs, a threshold of 0.7 is appropriate in identifying location tags with high precision. More specifically, from an initial set of 6,756,605 unique tags, distributed over approx. 13.3M photos, we were able to identify 241,072 unique location tags. Note that this is the number of distinct location tags, so even if a location tag is very popular (e.g., “paris”) it only appears once in the set. (See examples in Figure 3.2.)

3.4.2 Mapping Location Tags to Places

Location tags may be defined at various levels of granularity, for example, Country, Province, Town, etc. Yahoo! GeoPlanet uses a hierarchical model for places that provides both vertical consistency and horizontal consistency of place geography. Spatial entities in Geoplanet are identified by a unique 32-bit identifier: the *Where On Earth ID* or *WOEID*. Every named place represented by a WOEID can be mapped to a *location type* in the hierarchical model (e.g., Country, Town, Point Of Interest-POI, etc.). We would like to find a one-to-one mapping of the photo to a place, given the set of its location tags.

Normally a geographic name alone is not sufficient to identify a geographic place unambiguously because it may have been assigned to more than one place [40]. This happens particularly with descriptive names (for example, Newcastle, Takayama (Japanese = high mountain) and Matsushima (Japanese = pine island)), names of saints, or emigrated communities (for example, Athens). However, it is reasonable to assume that people deliberately use unambiguous placenames within local areas. Under this assumption it is possible to eliminate the problem of duplication of geographical place using two heuristics:

- **Heuristic 1: Focus the Search Near a Reference Point**: If we can focus the search of the

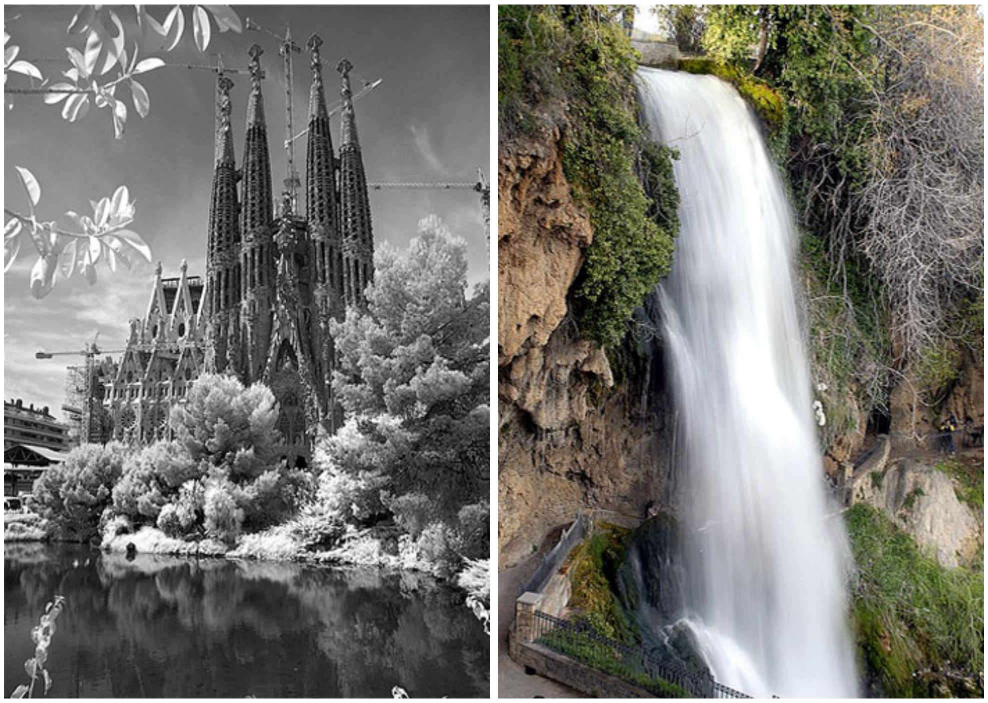
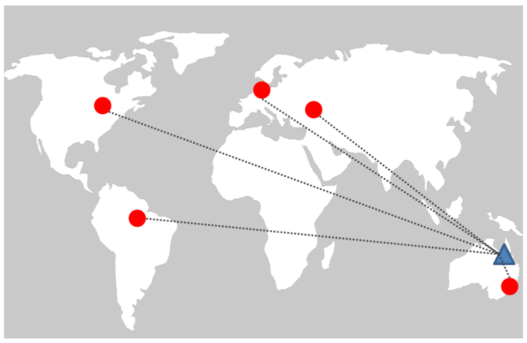

(a) *Gaudi's Sagrada Familia in Barcelona, Spain* (b) *Waterfalls in Edessa, Greece*

Figure 3.2: Distinguishing Location Tags from Free-text Tags. (a) *Tags* = {sagrada familia, gaudi, barcelona, modernist, cathedral, construction, cranes, infrared}, *Location tags* = {sagrada familia, barcelona}. (b) *Tags* = {waterfall, edessa, greece, green, pond, wall, tree, fall, balcony}, *Location Tags* = {edessa, greece}

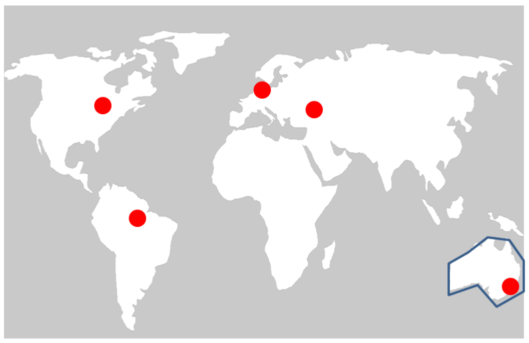
geographical place around or close to another reference point (i.e., another place), then a location tag has a higher chance of uniquely identifying a place under this restriction (See Figure 3.3(a)).

- **Heuristic 2: Knowledge of Higher Levels in a Hierarchy of Places**: If the wider area in which the geographical place resides (a higher level in a hierarchy of places, such as Country, etc.) can be identified, then a location tag has a higher chance of uniquely identifying a place under this restriction (See Figure 3.3(b)).

The first heuristic is always available. Each time a photo is placed on a map, latitude and longitude coordinates are assigned. These coordinates can be used to define a reference point place that guides the search and helps to disambiguate the location mention from other geographical places with the same name. The second heuristic offers further support in the disambiguation process when a photo has been assigned more than one location tag. In this case, it is expected that these can be mapped to different levels in a hierarchy of places. For example "Toronto, Canada" is mapped to (Town, Country) or "Sagrada Familia,

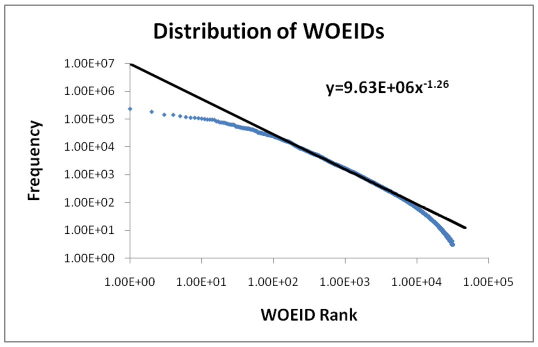


(a) *Heuristic 1*

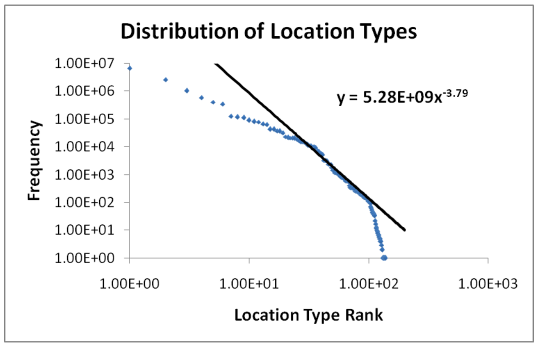


(b) *Heuristic 2*

Figure 3.3: Disambiguation Heuristics: Assume that red dots represent *places* on Earth with the same *location tag*. Our method efficiently disambiguates places by searching for a place that satisfies the location tag and in addition (a) is closer to a reference point (i.e., closer to the triangle on the map) or/and (b) resides in the specified boundaries of a known higher level in the hierarchy of places (i.e., Country is known)



(a) *WOEIDs*



(b) *Location Types*

Figure 3.4: WOEID and Location Type Frequency Distributions

Barcelona, Spain" is mapped to (Point of Interest, Town, Country).

Using the heuristics described above and Yahoo! GeoPlanet, we are able to efficiently disambiguate geographical names, to derive the *target location*. Each query submitted to Geoplanet, returns a list of places (WOEIDs) ordered from the most likely to the least likely. Due to the well-defined semantics of the above heuristics, in almost all cases we are able to identify the most likely WOEID with a probability larger than 0.9. Thus, we are able to almost always unambiguously identify the target location by its intended WOEID.

3.4.3 Defining and Assessing User Credibility

Manually placing photos on a map is not an easy task and can be inaccurate at times. In our research, we focus on manually geotagged photos and try to determine the reliability of individuals by assessing the accuracy with which they place photos on the map; the closer to the target, the more reliable a user is. We measure the proximity of a user's geotagging to the target location by computing their geodesic distance d (see next paragraph). The coordinates of the target location are assumed to be known through its WOEID (all WOEIDs in our dataset are assigned a number of attributes by third-party authoritative sources, such as longitude and latitude, which we consider as the ground truth in our research). Recall that given a photo and its set of location tags we know how to unambiguously map it to its WOEID with high probability. For locations with a surface area (such as a city, as opposed to a building or statue), a wide range of latitude/longitude coordinates will fall within its boundaries. The coordinates associated with a WOEID indicate the centroid of the bounding box that encompasses the location. It is possible that the centroid of the bounding box is not the same location as the human-centric logical center of the location. Therefore, rather than comparing the user's accuracy to the coordinates of the WOEID directly, we compare the user's accuracy, to the average accuracy of all users. We assume that in the aggregate, in large numbers, users are accurate, and can capture place boundaries with a sophistication that is not represented by a centroid-based coordinate system. An example of this phenomenon is the work of Tom Taylor.[2]

Geodesic Distance d

Geodesic distance is the distance of two points measured along the surface of the Earth. Calculating geodesic distance is typically based on some level of abstraction, which ignores changes in elevation and other irregularities in Earth's surface. Common abstractions assume a flat, spherical, or ellipsoidal Earth. We employ an ellipsoidal approximation of the surface and compute the *ellipsoidal distance* between two points. The Ellipsoidal distance is the shortest distance between two points along the surface of an ellipsoid. It is more accurate than methods such as great-circle distance which assume a spherical Earth. We use the inverse method of Vincenty's formula[145]. This method has been widely used in geodesy because it is accurate to within 0.5 mm on the Earth ellipsoid. In our experiments we compute geodesic distances using the $WGS84$ standard Earth reference ellipsoid, used by the Global Positioning System.

[2]http://tomtaylor.co.uk/projects/boundaries/ visited February 2010.

The Wisdom of the Crowd

The geodesic distance provides a measure of proximity of the user's geotagging to the target location, but presents little information on how reliable a user is in comparison to others. Moreover, geotagging happens at different levels of location granularity. For example one user geotags at the city level and another at the country level. Thus, it is not valid to compare user credibility based on different sets of photos and different sets of intended target locations. To compensate for this limitation, we rely on a wisdom of the crowd practice. The wisdom of the crowd refers to the process of taking into account the collective opinion of a group of individuals rather than a single expert to answer a question. In our case, we would like to compare the performance of a user to the collective behavior of all users that have tried to geotag a photo that corresponds to the same target location.

Let $W = \{w_1, w_2, ..., w_k\}$ be the set of all distinct WOEIDs in our data after mapping each photo to a WOEID. Figure 3.4(a) plots on a log-log scale the distribution of the WOEIDs (the number of times a given WOEID has been a target location). Let a random variable D represent geodesic distances in the users' geotagging activity. For a specific WOEID w_i the random variable D takes on N real values $d_1^{w_i}, d_2^{w_i}, ..., d_N^{w_i}$, with arithmetic mean $\overline{d^{w_i}}$ and standard deviation σ^{w_i}. We compute these values ($\overline{d^{w_i}}$ and σ^{w_i}) for all WOEIDs in our data and use them to characterize the quality of the geotagging behavior of users, through hits and misses.

Hits and Misses

We consider geotagging to be a binary decision: a *hit* or a *miss*. We define a user's geotagging to be a hit when $d_v^{w_i}$ (the geodesic distance of user's v actual geotagging to the target place w_i) is smaller than the average distance $\overline{d^{w_i}}$ of all users geotaggings for this target place plus λ times the standard deviation σ^{w_i} of these distances. By definition, if it is not a hit, then it is considered a miss. More specifically:

$$hit : d_v^{w_i} < \overline{d^{w_i}} + \lambda \cdot \sigma^{w_i}$$

$$miss : d_v^{w_i} \geq \overline{d^{w_i}} + \lambda \cdot \sigma^{w_i}$$

where v is a user, and λ is a parameter that controls the selectivity of hits and misses in the geotagging process. A larger λ indicates that a hit is easier (i.e., larger geodesic distance from the target is allowed) and vice versa (See Figure 3.5).

The λ parameter is effectively controlling a threshold that must be exceeded for a certain result to be considered a hit (or miss). Note that what consists a hit or miss is specific to each application or domain and depends in the sensitivity of the user task itself. So, while our method is generic, setting λ is an application-specific procedure that can help to fine-tune the credibility score of users. The rule of thumb in setting λ is to tune it in a value that effectively defines a variability of the credibility values among users (i.e., different credibility values are spread in users). If this parameter is not set properly, then the risk is to end up with the same credibility value for all users, which eventually renders the credibility property futile.

Recall that our initial objective was to associate each user v with a single credibility value c_v. However, individuals may tag photos at different levels in the place hierarchy (different *location types*) and a WOEID can be mapped to any location type. Figure 3.4(b) plots in a log-log scale the distribution of the location types in our data (that is, the number of times a target place was of that location type). Determining the semantics of a user credibility in a single value that aggregates information from all location types (Country, Town, POI, etc.) is challenging (as semantics of the hierarchy of places could be violated). Instead, we prefer to maintain *a vector representation* of a user's credibility, let $C_v = \{c_v^{l_1}, c_v^{l_2}, ..., c_v^{l_n}\}$, where each vector dimension represents the user credibility at a specific location type. We define the credibility value of a user v at a specific location type l_i (e.g., Town) to be the number of hits $h_v^{l_i}$ for a given location type, divided by the total number of geotagged photos by this user of that location type $t_v^{l_i}$. Note here, that we are allowed to aggregate averages over all WOEIDs of the same location type, despite the fact that some places are more popular than other. This is because our definition of hits and misses incorporates these semantics (i.e., despite how popular a place is, in order to get a hit you must be at least as accurate as many other people that tried to geotag photos for the same place). Our methodology aims to provide common ground in assessing the quality of the geotagging behavior.

Note that our methodology, while developed with a specific application in mind, can be easily generalized and employed in other domains. In principle, our method has two components: one component is generic and one specific to the application. The application-specific component is based on defining what consists a hit and what a miss in trying to accomplish a task in a domain area (for example, in Flickr, this was the geotagging process). The generic part is consisting of how to define user overall credibility scores among a number of tasks based on the aforementioned definition of hits and misses. It also consists of the "wisdom of the crowd" component that allows to compare the performance of a user to other users in a dynamic way, probably including tasks that require different expertise to be accomplished. Further, it describes cases where a user credibility score cannot be represented as an atomic value, but needs to be represented as a

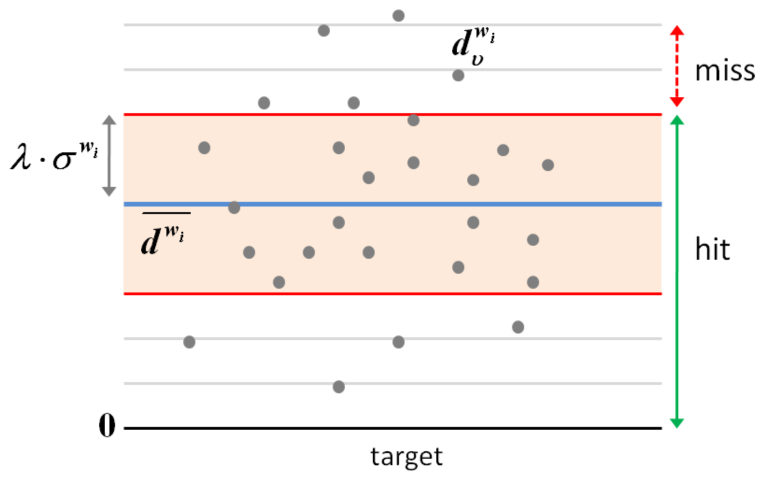


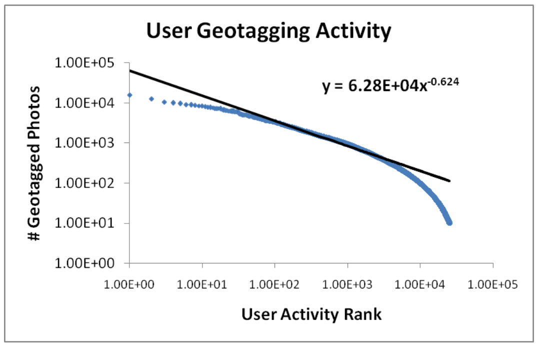
Figure 3.5: Illustrative example of the way a *Hit* (or *Miss*) is defined for a specific target location w_i following a wisdom of the crowd practice and provided the distances of all users' geotaggings to the target location

vector of values; this models cases where a task can be accomplished at different levels of abstraction (that require different level of expertise). Designing measures of behavior quality in a generic way is challenging as it largely depends on the application of interest and, as such, is out of the scope of this research.

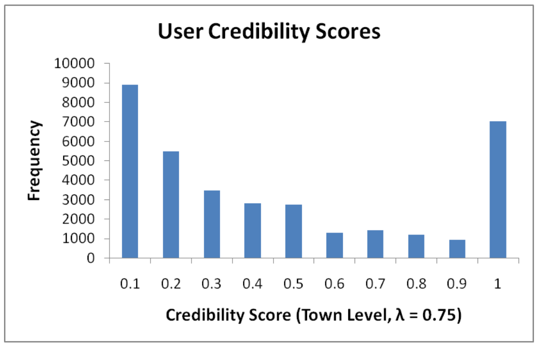
3.5 Influence vs. Behavior

Understanding the way in which an individual's choices depend on what other individuals do requires analysis that can be performed at two conceptually different levels of network resolution. A global one, in which we observe network effects in aggregate, and a local one, in which we observe how individuals are influenced by their network neighbors [44]. In Section 3.3, we detected the occurrence of social influence in the adoption of the geotagging innovation in Flickr by monitoring aggregate information of user actions on a macroscopic scale. In Section 3.4, we focused on individuals and monitored user behavior on a microscopic scale. In this section we try to find points where these two different levels of resolution converge and investigate the causality of social influence.

Many of our interactions with the rest of the world happen at a local, rather than a global, level - we often don't care as much about the full population's decisions as about the decisions made by friends and colleagues. To this end, we bring the analysis closer to the detailed network level and look at how individuals are influenced by their particular network neighbors. A particular individual behavior, such as



(a) *User Geotagging Activity*



(b) *Credibility Scores Histogram*

Figure 3.6: User Geotagging Behavior

geotagging, can be communicated to the network in different ways. In our research we focus on (a) intensity and on (b) credibility of geotagging. These properties are distinct as intensity of geotagging characterizes the *quantity* of the expressed behavior (i.e., how many times a user expressed the behavior), while credibility of geotagging characterizes the *quality* of the expressed behavior (i.e., how well a user expressed the behavior). Our research examines which of the two primitive types of the expressed behavior (related to quantity vs. related to quality) has more effect in the influence that a user exerts in her social network. Associating aggregate observations with individual behaviors improves our understanding of how individual's choices depend on what others do and eventually how ideas, behaviors and innovations diffuse in a population.

3.5.1 Data

As mentioned earlier, different users may be more or less accurate for different location types. We consider a user's credibility for each location type independently. In the following experiments we focus on the credibility of users at the *town* level. Limiting our analysis to a single location type provides more clear semantics when comparing users. We pick *towns* because they are the most popular location type in our data. In addition, there is a very large variation in user credibility scores at the town level.

To produce a more reliable data set, we reject users who only occasionally used the geotagging technology or who were extremely inaccurate in identifying the places associated with their photos. Thus we eliminated users who had fewer than ten photos geotagged as well as users that had a credibility score of zero. The final set consisted of approximately 25,000 users that have geotagged more than 5 million photos. Figure 3.6(a) plots in a log-log scale the distribution of user geotagging activity. The distribution can be modeled quite accurately, and the probability $f(x)$ of a user to have geotagged x number of photos is proportional to

$x^{-0.624}$. The x-axis represents the rank of each user, taking values from 1 to approximately 25,000, ordered by activity from the most to the least active.

Recall that λ is a parameter that controls the selectivity of hits and misses in the geotagging process, and setting it correctly is dependent on the particular application. Since we are investigating in a general setting, with no specific application in mind, we seek a setting that provides variability in the credibility scores. We experimented with different settings of λ, and found that making λ large (i.e., 1.0 and over) causes most users to be highly credible, while making λ small (i.e., 0.5 and below) is too strict and no users are credible. For values between 0.5 and 1.0, most users are assigned intermediate credibility scores. For our experiments we fix lambda to 0.75 as it provides sufficient variability in the credibility scores of users. Figure 3.6(b) shows a histogram of the credibility scores for $\lambda = 0.75$. Still, the majority of users have low credibility scores. The peak at credibility of one may be because all of the user's photos were from one location, or because the location was sufficiently large that the user could drop it in the appropriate place on the map. It may also be the case of users that tend to geotag photos of their hometown, thus being very accurate, or the images were taken with GPS-enabled cameras.

3.5.2 Experiment Sketch

We design an experiment that measures how influential is a user u to its social network, in adopting the geotagging innovation. The semantics hinge on the social influence experiment discussed in Section 3.3, but differ in focus. Here the focus is not on determining if a user's activation was due to a neighbor's influence, but rather the opposite, to evaluate how influential a user is in her network, i.e. her *social influence effect*. Formally, we say that user a has the potential to influence user b, if and only if user a adopted the geotagging technology at time t_1, and user b adopted it at time t_2, and a and b became friends at time t_3, where $t_1 < t_2$ *and* $t_3 < t_2$. That is, a user a has the potential to influence a user b, if a was activated before b and they became friends at least before b was activated. Therefore a user is said to influence a neighbor if he has the *potential* to pass to his neighbor a signal of access to the geotagging innovation.

Given these semantics, it is possible to algorithmically assess the potential social influence of a user (or a set of users V) to the network over a time period. Algorithm 1 gives the details of a recursive computational procedure for evaluating social influence in a finite number of steps. It takes as input a set of users V, a social graph G and the activation times of all users A in the monitoring period $[0, T]$, and returns the number of users that have potentially been influenced by the set of users V.

In our experiments we do not assume that users have the same amount of time to influence others.

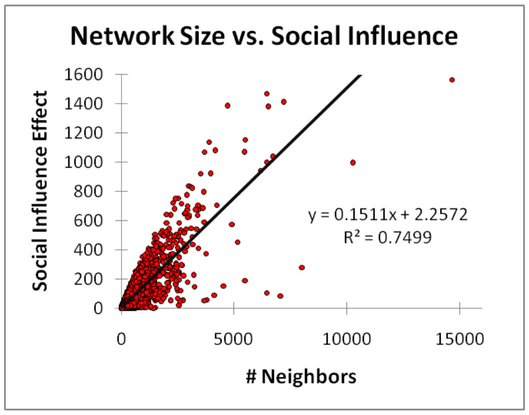
ALGORITHM 1: Social Influence Effect Estimation

```
Input: A Social Graph $G$, a set of users $V$ and the log of user activation times $A$
Output: The number of users that have been influenced

$G$: Social Graph ($from$, $to$, $friendship\_create\_time$);
$V$: Set of users;
$A$: Activation Times ($user$, $activation\_time$);
$I$: Set of Influenced Users;

$I \leftarrow \emptyset$;
forall the $v \in V$ do
    $t_1 = getActivationTime(v, A)$;
    $N^v = getFriends(v, G)$;
    forall the $n \in N^v$ do
        $t_2 = getActivationTime(n, A)$;
        $t_3 = getFriendshipCreateTime(v, n, G)$;
        if $t_1 < t_2$ && $t_3 < t_2$ then
            $I \leftarrow I \cup n$;
        end
    end
end
return $|I|$;
```

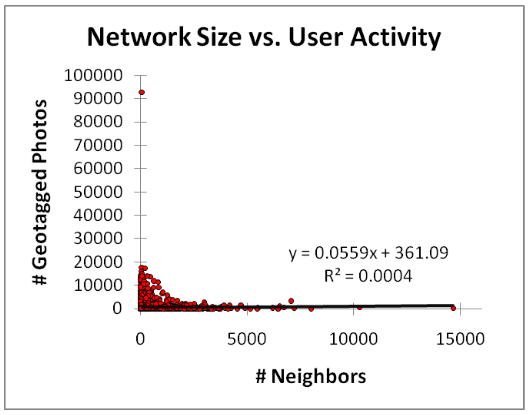
In fact, earlier adopters have larger amount of time. It is important to note though that we monitor the adoption of a real technology from its launch (time 0 in our experiments), and therefore provide to all users "equal" opportunity to be early adopters (through a form of external influence; reading about it, testing the innovation, etc.). Could late adopters complain that they didn't have the chance or the time to influence others? Yes, but this is always the case in a real setting (e.g., political influence, fads, diffusion of innovation, etc.) and we would like to be able to capture this effect. Algorithm 1 is the basic tool to assess and compare the *social influence effect* of varying sets of users in our experiments.

Effect of Neighborhood Size: Before investigating the effect of individual behaviors to the social influence, it is important to examine whether the number of neighbors that an individual has, plays an important role in determining her social influence. To determine this effect we explore the correlation between the network size of an individual (i.e., number of neighbors in our context) and her social influence effect.

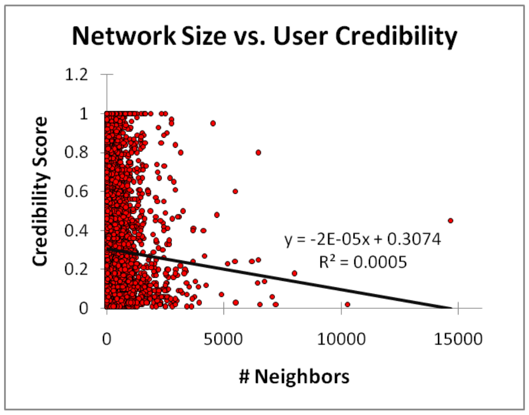
Figure 3.7 shows scatter plots and the regression lines for the correlation tests we run (Pearson's correlation was used as a measure of correlation). Our analysis shows that there is direct and high correlation between the neighborhood size of an individual and the social influence she exerts to her social network (Figure 3.7(a)). This is in alignment with [16], where authors report that adoption rates quicken as the number of friends adopting increases and this effect varies with the connectivity of a particular user. Due to the large effect that the neighborhood size has to an individual's social infuence, for the rest of the ex-



(a) *Network Size vs. Social Influence*



(b) *Network Size vs. User Activity*



(c) *Network Size vs. User Credibility*

Figure 3.7: Network Size Effect

periments we had to normalize on the neighborhood size parameter. The normalization allows to focus on the individual behavior observations. Otherwise, these observations could simply be a function of larger or smaller neighborhood. From now on, any time we refer to the social influence effect of a user, we refer to a proportion that represents the number of influenced neighbors over the total number of neighbors.

It's also important to note that as shown in Figure 3.7(b) and Figure 3.7(c), there is very weak to negligible correlation between an individual's neighborhood size and any of the two behaviors under investigatation. In other words, the number of the neighbors has no effect in the expressed behaviors of an individual.

Social Influence vs. User Activity

We hypothesize that users who are more active are more influential. This stands to reason because users who are more active upload more photos, and provide geotags for more photos, they are "experts" of a sort, in the social system of user-generated content. To test the hypothesis we run experiments that measure the relationship between user activity (as determined by the number of photos a user has geotagged) and influence in the Flickr social network.

From the set of 25,000 users, we first sort the users based on their activity (number of geotagged photos), from the most active to the least active. Then, we define 5 user groups, each consisting of 5,000 users, based on their activity level. The first group has the most active users and the last one the least active users (Figure 3.8(a)). For each activity level we compute the potential social influence in the network using Algorithm 1. We report the median value of the normalized social influence effect, as the distribution of this property is very skewed.

Assuming the increased visibility they have in their network such as through notification mechanisms in Flickr, active users, because they generate more content, are viewed by more people. In fact, we see in Figure 3.8(b) that more active users can be as much as 23% more influential in the social network, in terms of encouraging people to adopt the innovation of geotagging.

Social Influence vs. User Credibility

The measure of credibility based on geodesic distance is intuitive: a person is more accurate as a geotagger if she consistently places her photos very close to the true intended location. However credibility may relate not only to how accurate a person is, but how accurate they are perceived to be by others. Thus there is an interplay between influence and accuracy. We hypothesize that more credible users are more influential. To test the hypothesis we run experiments that measure the relationship between accuracy and influence in the Flickr social network.

From the set of 25,000 users, we first sort the users based on their credibility score, from the most credible to the least credible. Then, we define 5 user groups, each consisting of 5,000 users, based on their credibility level. The first group has the most credible users and the last one the least credible users (Figure 3.9(a)). For each user group we compute the potential social influence in the network using Algorithm 1. We report the median value of the normalized social influence effect, as the distribution of this property is very skewed.

One would expect that users with higher credibility scores - that is, users who are more accurate - would be more influential, assuming that people adopt a technology because they have seen it done well. But in fact, as we show in Figure 3.9(b), this is not the case. It appears that the reasons people become good geotaggers do not relate to social influence. That is, you may adopt the technology when you see another person try it, but whether they are good at it or not does not factor into your decision to adopt the technology. We further discuss the consequences of this result in the next paragraph.

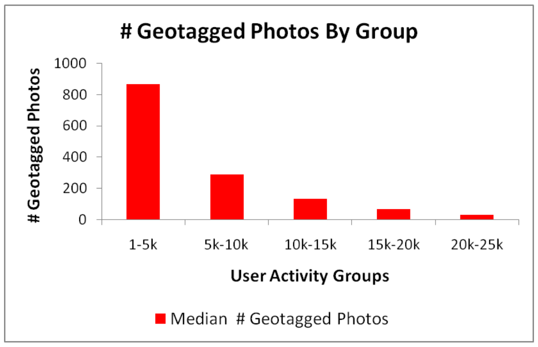
Hidden Social Influence: This is an unexpected finding, as credibility of a user consists an essential individual quality that appears to be currently hidden in the social network. This is an important observation, since it suggests that current design of online social systems may refrain users from exploiting the full potential of their social influence. In practice, it lessens or cancels the *social influence* of a user in her environment. This is not happening intentionally, but it's rather a consequence of stale design. Interfaces to electronic systems have traditionally been designed as single-user systems. The existence of other users and their activities have been implicitly assumed to be an attribute of the system that should be hidden from end-users. Similar design approach has been adopted for accessing information on the web.

This observation raises a more general concern. While *quantity* of behavior is usually clearly communicated in social systems, *quality* of behavior is systematically not (e.g., due to lack of feedback mechanisms related to quality). As users interact in online systems they expose or develop different qualities of individual behavior. This becomes more evident in modern online social systems, where behaviors may or (may not) be revealed by the system to their social peers. For example, the quality of user-generated content in Twitter[3] or in Facebook[4] varies drastically from being of high quality to being abuse or spam. Quality of user behavior consists an important part of a user's social value, which might translate to power of influencing her environment or decision making processes, but largely remains hidden. As such, our research suggests that users should have the option to reveal certain qualities of their behavior to their social network. Acknowledging the social value of users offers a design opportunity; in designing social systems, it is not only necessary to see other users, but to also clearly communicate what behaviors are disclosed and how they are formed - a design that entails a balance of visibility, awareness of others and their qualities.

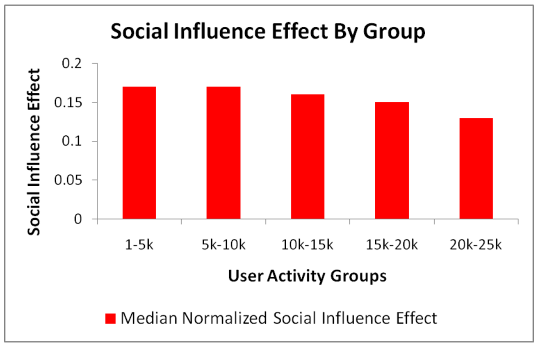
3.5.3 Tests of Significance

Thoughout our study we based our observations on sample populations. But, how dependent our observations on these samples are? We performed statistical significance tests for the two behaviors in our study (i.e., user activity and user credibility) to assess evidence in favor of or against our claims. The observations in the samples (i.e., groups of users) are all real numbers that describe the median value of the normalized social influence in each group of individuals. The values of the property we observe typically has skewed distributions. We actually run normality tests, based on the Shapiro-Wilk method, to show that the samples

[3]www.twitter.com
[4]www.facebook.com



(a) *User Activity By Group*



(b) *Social Influence vs. User Activity*

Figure 3.8: Social Influence vs. User Activity

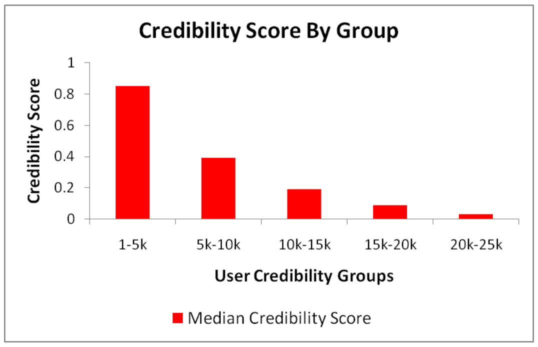
in each group do not follow a normal distribution. Thus, to compare k-samples (i.e., the 5 different groups in each behavior) we employ a method that does not assume a normal population of samples, such as the Kruskal-Wallis test, for a selected significance level $alpha = 0.05$. This test is a non-parametric method for testing equality of population medians among groups. Intuitively, it is identical to a one-way analysis of variance with the data replaced by their ranks. In the case that the null hypothesis is rejected by the Kruskal-Wallis test, we perform post-hoc analysis to identify which of the groups differ. We use the bonferroni adjustments post-hoc test to identify these pairs. For the various cases in our study, we formally define the following hypotheses:

H_0: *The samples are not significantly different*

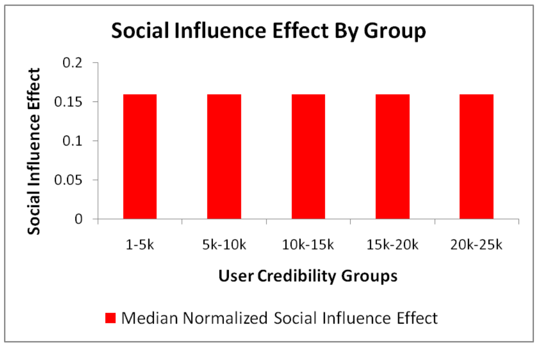
H_a: *The samples do not come from the same population*

For the case of Figure 3.8(b) (Social Influence vs. User Activity), the Kruskal-Wallis test showed that the samples do not come from the same population, thus rejecting the null hypothesis H_0. The post-hoc analysis revealed that all groups were significantly different with each other. This supported our claim that more active users are more influential in the social network.

For the case of Figure 3.9(b) (Social Influence vs. User Credibility), the Kruskal-Wallis test showed that the samples come from the same population, thus accepting the null hypothesis H_0. This supported our claim that being more credible does not necessarily make a user more influential in the social network.



(a) *User Credibility By Group*



(b) *Social Influence vs. User Credibility*

Figure 3.9: Social Influence vs. User Credibility

3.6 Conclusions

We developed a method that detects social influence in a social system. The method is similar to the method proposed in [9], as they both hinge on the idea of shuffling the timestamps of social actions in a network. Despite its technical soundness and completeness, their method makes assumptions of prior knowledge of the distribution of a node's influence over its neighbors and of a theoretical cascade model that when is simulated determines which nodes are eventually activated. Our method is simpler and is designed to both detect and quantify a user-defined social influence in the context of a real social system. We applied our method in Flickr to demonstrate that adoption of the geotagging innovation can to a large extent be attributed to social influence between users. This observation provides evidence of a cascading process taking place in the social graph; a process of users passing to their neighbors a signal of access to the geotagging innovation. The proposed method is generic and can be useful for providing decision support for designing viral marketing campaigns in social systems.

We performed a series of experimental and observational studies on rich data coming from a large social system in order to investigate causality, and in particular to draw a conclusion on the effect of changes of user behavior in determining how influential they are in their network. In particular, we studied the geotagging behavior of individuals that collectively leads to macroscopic properties of social influence and contagion. Our main objective was to investigate potential points of convergence between the macroscopic and microscopic view of the social system. This is significant because it allows us to argue about higher levels of granularity by observing individual behavior.

We investigated the role of the neighborhood size in an individual's behavior. Our findings suggest that individuals that have many neighbors have more potential to influence their friends in their social network. Then, we investigated two types of user behavior, frequency of using the geotagging innovation and accuracy in placing images on a map (based on a measure of credibility), and tried to assess the effect of the assumed behavior on its social influence in the network. Note that these two behaviors are interesting, as the former characterizes the quantity of the expressed behavior, while the latter characterizes the quality of the expressed behavior.

Our findings suggest that the first is a potential *reinforcing activity* such that when it occurs, the probability of a friend in the network adopting the behavior increases. This is an important observation that relates to the concept of *reinforcement* in behavior sciences [22, 139]. On the other hand, we found that the credibility of a user has only a small effect in how influential she is in her network. This raises concerns for the design of social systems, since user credibility, by construction, is an important indicator of a user's quality.

To the best of our knowledge, our work is the first that experimentally unveils that there are qualities of individual behavior that remain hidden, a practice that lowers the social influence of individuals in their network. As an outcome of our research, we propose the conception of mechanisms that inform users about their friends' activities and qualities, hoping that such a social feedback would eventually help users to regain their hidden social value and encourage other people in the network to expose or develop high quality behaviors. Similar systems are seen in [121, 122] where users become aware of similar people's surfing behavior and in Yahoo! Answers[5] where users are encouraged to answer accurately within a reward system based on a token economy.

There are many open issues left for future research. For instance, in our experiments we assume that social influence occurs only through internal stimuli, realized in Flickr through social feedback mechanisms. As such we do not take into consideration any external stimuli or source of communication. However, social systems co-exist (for example Flickr, Blogosphere, our family, our classmates and more) and communication between social systems is possible [117]. Formal verification of social influence occurrence is tricky since it is susceptible to the assumptions made about the social system. Being able to argue about causality of social influence on the whole is a challenging task that requires the design of controlled user experiments.

[5]http://answers.yahoo.com visited February 2010

Chapter 4

A Graph Augmentation Approach to Facilitate Online Social Processes

4.1 Introduction

Group formation and segregation, the rise and fall of fashion fads, the adoption or rejection of an innovation are all manifestations of fundamental *social processes*, such as homophily, influence and contagion, that now take place in online social networks and can be monitored and examined through traces of online social interactions. The ability of a social network to carry on such social processes is a characteristic that depends, to a great extent, on the *topology of the network*. Consider the process of information propagation; as individuals become aware of new or interesting pieces of information they have the chance to pass them forward to their friends, and so on and so forth. The number of nodes in the network that can be reached during the propagation depends on *contextual properties* of the propagation itself (e.g., the content of the information being transmitted, how susceptible nodes are in receiving the information, how willing they are to further transmit it, and so on), but there is also a genuine connection between the propagation process and the topology of the network. It becomes obvious that slight modifications in the network topology, might have a dramatic effect on its connectivity and thus to its capacity to carry on such social processes.

In this research, we approach network modification as a *graph augmentation problem*, where we ask to find a small set of non-existing edges (from now on we refer to them as *shortcut edges* or simply *shortcuts*) to add to the network that will optimize a connectivity property. While in traditional graph theory, network connectivity typically refers to a well-defined property, such as biconnectivity, in the context of a social graph, connectivity is usually manifested by the ability of a network to bring users closer to each other and drive up user engagement. Having plenty of choices for measuring the level of connectivity in the social graph, we focus on a structural feature and try to minimize the *characteristic path length* of the network. This property determines the average distance between pairs of vertices and essentially controls the evolution of social processes in the network; for example, it controls how efficiently information can propagate through a network. In the augmented network, information propagates easier and faster, as it has to travel from a node to another by traversing fewer edges of a more connected network. It is also interesting to note that while in traditional graph augmentation problems the addition of edges can occur anywhere in the network, in the context of a social graph, augmentation might need to respect constraints and *norms of social activity.* For example, it might make sense to connect two social peers (nodes) if only they share a contact (an adjacent node); we further elaborate on this issue when we formally define the problem.

There are many possible application scenarios of this work. For example, suggesting friends in a social network with the global objective of enabling efficient information dissemination or performing faster network simulations by reducing the convergence time of random walk processes or boosting collaboration in scientific networks by connecting the right peers. In the next paragraph, we provide more details about a few of these applications to further motivate our research.

4.1.1 Motivating Applications

We describe two applications, where a slightly modified network topology is preferred, to motivate the problem of interest.

Boosting Information Propagation Processes

A longstanding interesting topic of research in network analysis has been to identify community-like structure in complex networks (such as social, biological or information networks). Community structure (typically referred to as a module or cluster) refers to the occurrence of groups of nodes in a network that are more densely connected internally than with the rest of the network. Identifying communities is interesting because communities are often interpreted as organizational units in social networks. However, there are cases where evidence of tightly-knit communities in a network topology might constrain the evolution of processes that take place or unfold in it. This is the case of the process by which information flows in a social network. A *critical problem* of this social process' evolution is that while information spreads effectively among members of the same community (intra-cluster), it can eventually stall when it tries to break into other communities (inter-cluster). In fact, as Easley and Kleinberg [43] state:

> "Clusters and cascades (the outcome of information spread) can be seen as natural opposites, in the sense that clusters block the spread of cascades, and whenever a cascade comes to a stop, there is a cluster that can be used to explain why."

In this research, we are interested in exploring how small changes in a network's topology, manifested by the addition of a few shortcuts, might alleviate its community structure, giving rise to more efficient information propagation processes.

Boosting Random Walks on Graphs

Graph sampling has been a common strategy to efficiently compute interesting metrics in a large graph (e.g., distance measures, centrality measures, connectivity measures and so on). The objective of sampling is to randomly select elements (nodes or edges) according to a pre-determined probability distribution, and then to generate aggregate estimations based on the retrieved samples. Many of the graph sampling techniques are built upon the idea of performing random walks over the graph. One of the most popular schemes is the simple random walk (SRW) [98], where a walk starts from an arbitrary node and in each step picks the next node to visit uniformly at random from the adjacent nodes. The last node to be visited by the walk forms a node of the sample. If the random walk is sufficiently long, then the probability of each node to be in the sample tends to reach a stationary (probability) distribution proportional to the degree of each node (the number of connections it has to other nodes). Based on the retrieved samples and knowledge of such a stationary distribution one can generate unbiased estimations of aggregates over all nodes in the network.

A *critical problem* of that sampling method is that a random walk might require to take a very large number of steps in order to retrieve a single sample. The minimal length of a random walk in order to reach the stationary distribution usually appears in the literature as the *mixing time* or *mixing rate* of a random walk or Markov chain (i.e., how fast the random walk converges to its stationary distribution). Studies on graph theory suggest that the mixing time is tightly related to the connectivity of the graph. In particular, strongly-connected graphs are known to have small mixing time (fast convergence), while weakly-connected graphs are known to have large mixing time (slow convergence). How "well-knit" a graph is, is usually determined by *the conductance of a graph.* It is known that graphs with large conductance have a small mixing time. Therefore, the smaller the conductance of a graph is, the longer the random walk of the sampling method will be.

In this research, we are interested in exploring how small changes in a network's topology, manifested by the addition of a few shortcuts, might increase the conductance of the graph and dramatically improve the performance of graph sampling-based simulations, driven by faster random walks.

4.1.2 Contributions

This chapter makes the following contributions:

- We formalize the problem of finding a set of shortcuts to add in a social graph that will *shrink* the network as much as possible and formally characterize its hardness.
- We propose a novel method (*path screening*) for efficiently evaluating the utility of adding shortcuts in a network.
- We present a *sampling-based variant* of the proposed method that improves its performance and extents its applicability to larger graphs.
- We formally validate the claims of our methods.
- We evaluate the accuracy and efficiency of our methods using real and synthetic data and show that they (i) outperform sensible baselines, (ii) ease the spread of information in networks, and (iii) speed up the convergence of random walk based simulations in graphs, by *a multitude of times* and for *a varying range of conditions.*

4.1.3 Chapter Organization

The rest of the chapter is organized as follows. Section 4.2 introduces notation and presents background on graph theory that is required to understand the chapter. Section 4.3 formally defines the problem of interest and characterizes its hardness. Some theoretical results of the problem are presented in Section 4.4. In particular, we show that in a graph, the sum of the lengths of all-pairs shortest paths is equal to the sum of the edge betweenness of all its edges. We also show that the function of adding shortcuts in a graph in order to minimize its characteristic path length, while being monotonic, it is not submodular; therefore efficient techniques that rely on the submodularity assumption cannot be employed to solve the optimization problem. Section 4.5 presents efficient heuristic methods for fast assessment of the importance of a shortcut in a network. We experimentally evaluate our methods in Section 4.6. In Section 4.7 we review related work and conclude in Section 4.8.

4.2 Preliminaries

4.2.1 Network Model

Let $G(V, E)$ be a connected undirected simple graph with V the set of vertices and E the set of edges, where $|V| = n$ and $|E| = m$. Let $d(u, v)$ be the length of the shortest path between the vertices u and v. The sum of all-pairs shortest path lengths is $L = \sum_{(u,v)\in V\times V} d(u, v)$. Then, the *characteristic path length*, $\bar{L}$, is defined as the average shortest path length over all pairs of vertices in the graph ($\bar{L} = L/\binom{n}{2}$). $\bar{L}$ is a measure of the efficiency of information transport on a network. It is also a measure of how close users are to each other which encourages social interaction and drives up user engagement. The *range* of an edge $R(u, v)$ is the length of a shortest path between u and v in the absence of that edge. An edge with $R(u, v) \geq 2$ is called a *shortcut*.

4.2.2 Simple Random Walk on a Graph

We recall a few definitions of random walks on graphs. For more detailed exposition, see [136]. We define the degree of a node $v \in V$ as the number of nodes in V adjacent to v and denote it by $deg(v)$. We also define the neighborhood of a node v to be $N(v) = \{u : d(u, v) = 1\}$ (so $v \notin N(v)$). A simple random walk on a graph G is a Markov Chain Monte Carlo method which takes successive random steps according to a transition probability matrix $P = [p_{vu}]$ of size $n \times n$ where the $(v, u)^{th}$ entry in P is the probability of moving from node v to node u defined as:

$$p_{vu} = \begin{cases} \frac{1}{deg(v)} & \text{, if } u \in N(v) \\ 0 & \text{, otherwise} \end{cases}$$

At any given iteration t of the random walk, let us denote with $\pi(t)$ the probability distribution of the random walk state at that iteration ($\pi(t)$ is a vector of n entries). The state distribution after t iterations is given by $\pi(t) = \pi(0) \cdot P^t$, where $\pi(0)$ is the initial state distribution. The probability of a t steps random walk starting from v and ending at u is given by P^t_{vu}. For irreducible and aperiodic graphs (which is the case of undirected connected social graphs), the corresponding Markov chain is said to be ergodic. In that case, the stationary distribution π of the Markov chain (the distribution after a random walk of length μ, as $\mu \to \infty$.) is a probability distribution that is invariant to the transition matrix P (i.e., satisfies $\pi = \pi \cdot P$). For an undirected unweighted connected graph G, the stationary distribution of the Markov chain over G is the probability vector $\pi = [\pi_v]$, where $\pi_v = deg(v)/2|E|$.

Mixing Time

The mixing time of a Markov process is a measure of the minimal length of the random walk in order to reach the stationary distribution. Let $\Delta(t)$ be the relative point-wise distance between the current sampling distribution and the stationary distribution, after t steps of simple random walk:

$$\Delta(t) = \max_{v,u \in V, u \in N(v)} \left\{ \frac{|P^t_{vu} - \pi_u|}{\pi_u} \right\}$$

where P^t_{vu} is the element of P^t with indices v and u.

Definition 1. ***(Mixing time of a graph)*** *The* mixing time *of a graph is the minimum number of steps t required by a random walk to converge; i.e., $\Delta(t) \leq \epsilon$, where ϵ is a threshold.*

Note that social graphs have been shown to have relatively larger mixing time than what it was usually anticipated [107].

4.2.3 Community Structure and Conductance

A common characteristic in the study of complex networks is *community structure.* The quality of a community is commonly defined using modularity or flow-based measures. We adopt a more intuitive community quality concept, *conductance.*

Definition 2. ***(Conductance of a set of nodes)*** *Let $G = (V, E)$ be an undirected graph and $C \subset V$ be a set of nodes (C can be thought of as a cluster or community). Then the conductance $\Phi(C, \bar{C})$ (or simply $\Phi(C)$) is given by:*

$$\Phi(C) = \Phi(C, \bar{C}) = \frac{|E_{C,\bar{C}}|}{Vol(C)}$$

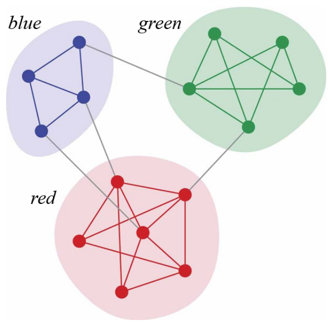


Figure 4.1: Community structure in complex networks. For example, $\Phi(blue) = 3/13$, $\Phi(green) = 2/18$, $\Phi(red) = 3/25$. The set of *green* nodes (that has the lower conductance) forms a more community-like set of nodes than the set of *blue* or *red* nodes. The conductance of the graph is $\Phi_G = 2/18$, which represents the minimum conductance of any set of nodes C, where $|C| \leq |V|/2$.

where

$$\bar{C} = V - C$$

$$E_{C,\bar{C}} = \{(v,u) \in E | v \in C, u \in \bar{C}\}$$

$$Vol(C) = \sum_{v \in C} deg(v)$$

The notion of conductance $\Phi(C, \bar{C})$ of a set of nodes C may be thought of as the ratio between the number of connections pointing outside the community C (to nodes in $\bar{C}$) and the number of connections inside C. It is widely-used to capture quantitatively the notion of a good network community as a set of nodes that has better internal-connectivity than external-connectivity [95]. Note that more community-like sets of nodes have lower conductance (Figure 4.1).

Using conductance as a measure of network community quality, one can define the *conductance of a graph.*

Definition 3. ***(Conductance of a graph)*** *Let $G = (V, E)$ be an undirected graph. The conductance Φ_G of G is given by:*

$$\Phi_G = \min_{|C| \leq |V|/2} \Phi(C, \bar{C})$$

Graph Partitioning

The conductance Φ_G of a graph is intractable to compute exactly (*NP-hard*), since we need to search over all possible cuts of a graph (of any community size $|C|$) and find the one with minimum conductance. Therefore, we have to resort to efficient approximation methods for computing partitions of graphs with minimum conductance over all the possible cuts. This is the *graph partitioning* problem, where the goal is to find groups of nodes (clusters) in a graph G, with roughly equal size, such that the number of edges between the groups is minimized and a merit function is optimized. In our analysis, the popular graph partitioning package Metis [71] has been used to obtain sets of nodes that have very good conductance scores. Metis is a multi-level graph partitioning algorithm that can give both fast execution times and high quality results.

4.2.4 Relation between Conductance and Mixing Time

Conductance is a measure of the tendency of a random walk to move out of a subset of nodes. The relationship between the graph conductance and the mixing time of a random walk is explained by the following inequalities [6, 67]:

$$(1-2\Phi(G))^t \leq \Delta(t) \leq \frac{2|E|}{min_{v\in V}\cdot deg(v)}\left(1-\frac{\Phi(G)^2}{2}\right)^t$$

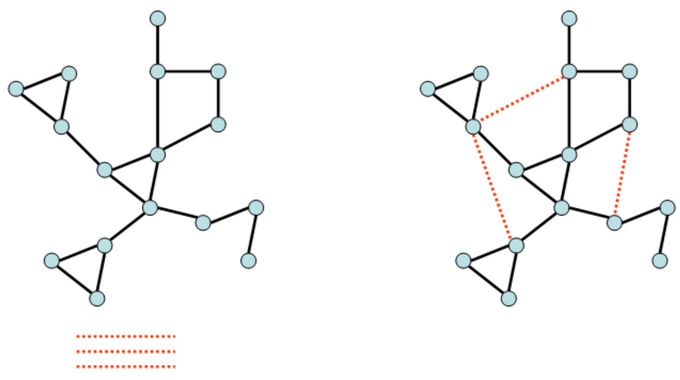
The above inequalities effectively provide a characterization of a simple random walk's mixing time in terms of the graph conductance $\Phi(G)$. One can see that the larger the graph conductance $\Phi(G)$ is, the smaller the mixing time t will be. Intuitively, high conductance would imply that the graph has a relatively low mixing time (see [79] for details and discussion). Note, as well, that because of the logscale relationship between $\Phi(G)$ and the mixing time t, a small change on $\Phi(G)$ will lead to a significant change of t. Now, if ϵ is a threshold, let

$$\frac{2|E|}{min_{v\in V}\cdot deg(v)}\left(1-\Phi(G)^2/2\right)^t \leq \epsilon$$

By applying the natural logarithm to both sides, we get

$$t \geq \frac{1}{log(1-\Phi(G)^2/2)} log\left(\frac{\epsilon}{\frac{2|E|}{min_{v\in V}\cdot deg(v)}}\right) \tag{4.1}$$

Based on the minimum value of t in (4.1) we can define a theoretical mixing time of the fastest random walk in a graph. We employ this measure in the experimental evaluation.



(a) $G = (V, E)$, $|S| \leq k$, $\mathcal{P}$ (b) $G_{aug} = (V, E \cup S)$

Figure 4.2: Graph augmentation of G to G_{aug}. (a) the input consists of G, an integer k and an augmentation policy $\mathcal{P}$, (b) the augmented graph G_{aug} is a possible output. We ask to determine the set of shortcuts S that when used to augment G, $u(S)$ is maximized.

4.3 The Problem

Let $G(V, E)$ be a connected undirected simple graph. Now, assume that we are allowed to augment G with a small number of edges, with the objective of optimizing a connectivity property. We denote G_{aug} an augmentation of G (see Figure 4.2). While having plenty of choices for measuring the level of connectivity in G_{aug}, we focus on a structural feature, namely the *characteristic path length*, $\bar{L}$. Note that, adding edges in a network results in altering some of the st-shortest paths in the graph, where $s, t \in V$. It is worth noting immediately that while adding long-haul edges (shortcuts with large range R) may bring a large gain in terms of connectivity for some (s, t) pairs (the new length of a shortest path connecting them will be much shorter), long-haul edges are likely to affect the shortest path length of only a few (s, t) pairs. On the other hand, suggesting short-haul, intra-community edges (shortcuts with small range R) is likely to affect the shortest path length of many (s, t) pairs, but the gain in connectivity for each of them might be limited (i.e., the new length of a shortest path connecting them will not be much shorter). Therefore in trying to suggest edges to be added to the graph we are facing an interesting trade-off. Let CS represent the set of all candidate shortcuts for addition. This set represents all shortcut edges (i, j) between any non-adjacent nodes $i, j \in G$ (i.e., $R(i, j) \geq 2$). Note that in a sparse network of n nodes (most real networks are sparse), there exist (almost) $n(n-1)/2$ candidate shortcuts. Now, assume the existence of a *utility set function* $u : 2^{CS} \rightarrow \mathbb{R}$ that maps a given subset of shortcut edges $S \subseteq CS$ to a non-negative real number that represents the difference of the sum of all-pairs shortest path length ($L(.)$) in $G = (V, E)$ and $G_{aug} = (V, E \cup S)$. Formally,

$$u(S) = L(G) - L(G_{aug}) \tag{4.2}$$

Another critical issue is related to the semantics of the set of shortcut edges S. From a network optimization perspective one might be interested to add shortcuts anywhere in the network. However, in some settings (e.g., suggesting friends in a social network) it might be required to suggest a few shortcuts to each node in the network, in a balanced way. In both scenarios we are interested in a more efficient network, but we are required to respect an *augmentation policy* $\mathcal{P}$ of adding shortcuts in the network. We consider two particular cases that we see to be of more interest in practice and define the following *augmentation policies*:

- *Network-centric Policy*: Graph augmentations can be applied anywhere in the network. In this case, an integer k that is provided as input represents the total number of shortcuts to be added, not necessarily equally distributed over nodes. Therefore, $|S| \leq k$.
- *Node-centric Policy*: Graph augmentations should be applied equally over all nodes in the network. In this case, an integer k that is provided as input represents the total number of shortcuts to be attached to *each node*. Therefore, $|S| \leq k|N|$.

The two *augmentation policies* control the level of flexibility in suggesting specific shortcuts to add in a graph. For example, following the *network-centric policy* it is allowed to select k shortcuts and attach them to a single node, in an attempt to resemble the topology of a star network. The intuition behind this is that stars are network topologies with very small average shortest path distance [104]. On the other hand, the *node-centric policy* restricts the number of shortcuts that can be attached to a single node and, in addition, requires that the same number of shortcuts is attached to any node in the network (e.g, one shortcut to each node). It becomes clear that the *augmentation policy* can have important implications on the way that algorithms that try to solve the problem operate. In one case, we seek to find the best k shortcuts globally in the network, therefore early stopping conditions can be considered, as soon as, the required number of shortcuts has been reached. In the other case, we seek to find the best k shortcuts for *each node*, therefore we might need to evaluate a possible larger set of candidate shortcuts before meeting the requirement. We further elaborate on this issue in Section 4.6.

We are now in position to formally define the problem of interest in this research.

Problem 1. *Let $G = (V, E)$ be a connected, undirected, simple graph, and let $CS = \{(i, j) : R(i, j) \geq 2\}$ be* *a set of candidate shortcuts for augmentation. If $G_{aug} = (V, E \cup S)$ is the augmented graph of G, then the objective is, given G, an integer k and an augmentation policy $\mathcal{P}$, to find a set of shortcut edges $S \subseteq CS$ that maximizes $u(S)$.*

It is easy to see that maximizing $u(S)$ is equivalent to maximizing $u'(S) = \bar{L}(G) - \bar{L}(G_{aug})$, since we only add new edges in G and do not change its number of nodes.

4.3.1 Problem Hardness

We establish the hardness of our problem by exploiting its equivalence with the weighted version of the *Maximum Coverage Problem* (WMCP). Not surprisingly, it turns out that our problem is NP-hard.

Theorem 1. *Problem 1 is NP-Hard.*

Proof. We prove the claim by reducing our problem to the WMCP. In WMCP the input is a universe $\mathcal{U}$ of n elements $\mathcal{U} = \{e_1, e_2, ..., e_n\}$, a collection $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ of m subsets of $\mathcal{U}$, such that $\bigcup_i C_i = \mathcal{U}$, a weight function $w : \mathcal{C} \to \mathbb{R}$ that assigns non-negative weights to subsets and an integer $k \leq m$. The goal is to select a subset $\mathcal{S} \subseteq \mathcal{C}$, such that $|\mathcal{S}| \leq k$ and the combined weight in the union of the subsets to be maximized. It is easy to see that our problem can be reduced to WMCP as follows: let a universe $\mathcal{U}$ of n elements, where each element represents an (s, t) pair of the graph G. Let, as well, a collection of candidate shortcuts $\mathcal{CS} = \{CS_1, CS_2, ..., CS_m\}$, where each candidate shortcut CS_i represents a set of (s, t) pairs; this is the set of (s, t) pairs whose shortest path lengths are affected (shortened) by the addition of the corresponding shortcut in G. Moreover, every candidate shortcut is assigned a weight w that represents the utility u of adding the corresponding shortcut in G. Then, given an integer k, a collection of candidate shortcuts and their utility scores, the goal is to select a subset of shortcuts $\mathcal{S} \subseteq \mathcal{CS}$, such that $|\mathcal{S}| \leq k$ that when added in G the combined utility is maximized. Problem 1 is NP-hard by reduction to the WMCP, one of the classical NP-hard combinatorial optimization problems. This concludes our proof. □

4.3.2 Negative Results

The maximum coverage problem is NP-hard, and cannot be approximated within $1 - \frac{1}{e} + o(1) \approx 0.632$ under standard assumptions. This result matches the approximation ratio achieved by a generic greedy algorithm used for maximization of submodular functions with a cardinality constraint [112]. The generic greedy algorithm for maximum coverage chooses sets according to one rule: at each stage, choose a set which maximizes the number of elements covered by the new set but not by any of the preceding ones.

Moreover, due to Feige, we know that the greedy algorithm is essentially the best-possible polynomial time approximation algorithm for maximum coverage [47]. That inapproximability result applies to the weighted version of the maximum coverage problem, as well, since it holds the maximum coverage problem as a special case.

These results, suggest that the greedy heuristic which adds the current best shortcut to a graph G, until k shortcuts are added would be the best possible heuristic. However, we show that the function $u(.)$ of adding shortcuts in a graph in order to minimize the average shortest path length, while being strictly monotonic (see Theorem 3), it is not submodular (see Theorem 4). Submodular functions have some nice parametric and post-optimality properties and related optimization problems can be solved efficiently if the submodularity property is valid [48, 70, 111, 146]. Since $u(.)$ does not possess the submodularity property, techniques that utilize it for approximation cannot be used. Thus, strictly speaking, we cannot expect to have an efficient algorithm with a provable approximation guarantee for our problem. Nevertheless, given the hardness of the problem, a greedy algorithm of successively adding shortcuts by repeatedly picking the best shortcut could still be a good heuristic (see Section 4.5). But, since the search space is massive, the greedy algorithm would be extremely slow. So, we focus our attention on designing an *efficient heuristic algorithm*, by avoiding unnecessary coverage evaluations. We explain the details of our algorithm in Section 4.5.

4.4 Some Theoretical Results

In this Section, some theoretical results are discussed.

4.4.1 Relationship between Characteristic Path Length and Edge Betweenness in G

We have structured the problem statement around minimizing the characteristic path length of a graph, $\bar{L}$. We discuss here the relationship between $\bar{L}$ and another popular network structural property, namely *edge betweenness.*

Definition 4. ***Edge Betweenness*** *[54]: The betweenness centrality of an edge or simply edge betweenness is the number of shortest paths between pairs of nodes that run along it. If there is more than one shortest path between a pair of nodes, each path is assigned equal weight such that the total weight of all of the paths* *is equal to unity. Formally, let $\sigma_{st} = \sigma_{ts}$ denote the number of shortest paths from $s \in V$ to $t \in V$ and let $\sigma_{st}(e)$ denote the number of shortest paths from s to t that pass through edge e. Then the edge betweenness of e is given by*

$$b_e = \sum_{s,t \in V, s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

Definition 5. ***Total Betweenness****: Let B be the sum of the edge betweenness of all edges in G, given by*

$$B(G) = \sum_{e \in E} b_e$$

Interestingly, we show that in a graph G, the sum of all-pairs shortest paths, $L(G)$, is equal to the sum of the edge betweenness of all edges, $B(G)$.

Theorem 2. *Let $G(V, E)$ be a connected undirected simple graph with V the set of vertices and E the set of edges. If L is the length of all-pairs shortest paths and B is the sum of the edge betweenness of all edges in G, then $B(G) = L(G)$.*

Proof. Without loss of generality we fix a pair of nodes $s, t \in V$. Assume that there are p^{st} such shortest paths between s, t, all of which have the same length $d(s,t)$. Note that each path $p_i \in p^{st}$ consists of $|d(s,t)|$ edges and contributes to the edge betweenness of each edge along the path by a factor of $\frac{1}{p^{st}}$. So, each single shortest path p_i contributes to the total edge betweenness by a factor of $\frac{d(s,t)}{p^{st}}$, and all p^{st} shortest paths of the pair of nodes $s, t \in V$ contribute by $d(s,t)$ $(= p^{st} \cdot \frac{d(s,t)}{p^{st}})$. Now, considering and summing up the total edge betweennesses of all s, t pairs in G we get $B = \sum_{s \in V} \sum_{t \in V} d(s,t)$ or $B = L$. $\square$

Due to Theorem 2, minimizing $u(S)$ in equation (4.2) is equivalent to minimizing:

$$u''(S) = B(G) - B(G_{aug}) \tag{4.3}$$

4.4.2 Properties of the Utility Function $u(.)$

We show some properties of the utility function $u(.)$.

Lemma 1. *Let $G(V, E)$ be an undirected graph and let $G_{aug} = (V, E \cup \{e\})$ be an augmentation of G after the addition of an edge $e \in CS$. Then, $L(G_{aug}) < L(G)$.*

Proof. Without loss of generality we fix a pair of nodes $s, t \in V$ and let $d(s,t)$, $d'(s,t)$ be the length of a shortest path connecting them in G and in G_{aug}, respectively. For any shortest path between s and t in

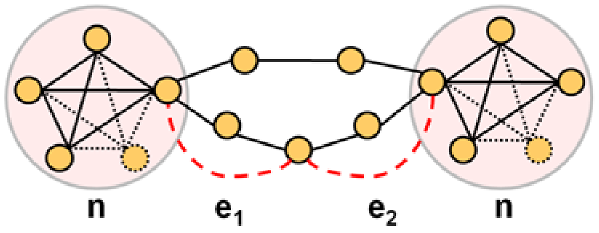


Figure 4.3: Submodularity Counterexample. Assume $G(V, E)$ and let the sets $X = \{\}$, $Y = \{e_1\}$ and $Z = \{e_1, e_2\}$.

G_{aug} there are two options:

- it does not traverse e: then $d'(s,t) = d(s,t)$.
- it traverses e: then $d'(s,t) < d(s,t)$.

Therefore, for any pair of nodes $s, t \in V$ it holds that $d'(s,t) \leq d(s,t)$. Summing up the shortest path lenghts of all s, t pairs in G_{aug} and G we get $L(G_{aug}) \leq L(G)$ (*monotonically decreasing*). Now, let the new edge e be incident to two nodes, say x and y. It's easy to see that the (x, y)-shortest path in G_{aug} is always shorter than the (x, y)-shortest path in G. Therefore, after the addition of the edge e in G, there is at least one pair of vertices where $d'(x,y) < d(x,y)$. It follows that $L(G_{aug}) < L(G)$. □

Theorem 3. *$u(.)$ is strictly monotonic.*

Proof. From Lemma 1 it follows that adding an edge $e \in CS$ to a set $S \subset CS$ will always cause $u(.)$ to increase, so $u(S \cup \{e\}) > u(S)$ for any edge e and set S (strictly increasing). □

Theorem 4. *$u(.)$ is not submodular.*

Proof. Let Z be a finite set. A function $f : 2^Z \to R$ is submodular if for any $X \subset Y \subset Z$ and $e \in Z \setminus Y$, it holds that:

$$f(X \cup \{e\}) - f(X) \geq f(Y \cup \{e\}) - f(Y)$$

Intuitively, a submodular function over the subsets demonstrates "diminishing returns". We provide a counterexample that proves that the utility function $u(.)$ does not possess the submodularity property.

Counterexample: Let $G(V, E)$ be an undirected graph and let the sets $X = \{\}$, $Y = \{e_1\}$ and $Z = \{e_1, e_2\}$ (see Figure 4.3). It is easy to see that $X \subset Y \subset Z$. Note that the edges in G form two very large fully connected subgraphs, each of size n (i.e, the number of nodes in each subgraph). Then, for $e = e_2 \in Z \setminus Y$ holds that:

$$
\begin{aligned}
u(X \cup \{e\}) - u(X) &= \big(L(G) - L(G(V, E \cup X \cup \{e\}))\big) \\
&\qquad - \big(L(G) - L(G(V, E \cup X))\big) \\
&= L(G(V, E \cup X)) - L(G(V, E \cup X \cup \{e\})) \\
&\simeq 5\frac{n(n-1)}{2} - 5\frac{n(n-1)}{2} = 0
\end{aligned}
$$

and

$$
\begin{aligned}
u(Y \cup \{e\}) - u(Y) &= \big(L(G) - L(G(V, E \cup Y \cup \{e\}))\big) \\
&\qquad - \big(L(G) - L(G(V, E \cup Y))\big) \\
&= L(G(V, E \cup Y)) - L(G(V, E \cup Y \cup \{e\})) \\
&\simeq 5\frac{n(n-1)}{2} - 4\frac{n(n-1)}{2} = \frac{n(n-1)}{2}
\end{aligned}
$$

The coefficients 4 and 5 above represent the typical shortest path lengths for (almost) all pairs of nodes in the network instances.

Asymptotically it holds that:

$$u(X \cup \{e\}) - u(X) < u(Y \cup \{e\}) - u(Y)$$

Therefore, we obtain a contradiction and we can conclude that the submodularity property does not hold. $\square$

4.5 Shortcut Utility Computation

Let u_{xy} represent a measure of the utility of adding a single shortcut edge $(x, y) \in CS$ in G. According to (4.2), it is:

$$u_{xy} = L(G) - L(G_{aug})$$

To evaluate u_{xy}, we need to recompute the shortest paths between all pairs of vertices $s,\ t \in V$ in G_{aug}. This is because a previously found shortest path connecting s, t may now need to traverse the newly added

ALGORITHM 2: Greedy Method

Input: A Social Graph $G(V, E)$ and a set of Candidate Shortcuts $CS = \{(x, y) : R(x, y) \geq 2\}$
Output: A shortcut edge e_{max} with maximum utility

```
$L(G) = computeL(G)$;
$u_{max} = 0$;
forall the $(x, y) \in CS$ do
    $G_{aug} \leftarrow G(V, E \cup \{(x, y)\})$ ;                    // add $(x, y)$ in $G$
    $L(G_{aug}) = computeL(G_{aug})$;
    $u_{xy} = L(G) - L(G_{aug})$;
    if $u_{xy} > u_{max}$ then
        $u_{max} = u_{xy}$;
        $e_{max} \leftarrow (x, y)$;
    end
end
return $e_{max}$;
```

edge (x, y). Let $d'(s, t)$ represent the length of the shortest path between s and t in G_{aug}. Then:

$$u_{xy} = \sum_{(s,t) \in V \times V \setminus E} (d(s, t) - d'(s, t))$$

In the following paragraphs we describe methods for computing the utility of all candidate shortcuts for augmentation. We first describe a greedy method. Then, we explain why this approach is not scalable and thus not suitable for larger graphs, and discuss methods that can scale up its performance.

4.5.1 Greedy Method

We describe a general greedy method (see Algorithm 2) where all candidate shortcuts are considered and a shortcut is selected in each iteration that lowers the average shortest path length of the graph as much as possible (locally optimal choice). The algorithm operates in k rounds. In each round i, the algorithm determines the shortcut with the highest utility and adds it to the set of the top-k edges to be suggested for augmentation. Equivalently, this means that the shortcut selected in round i is the one that minimizes the average shortest path length in this round and for this graph. In more detail, the method (see Algorithm 2) iterates over all candidate shortcuts CS and for each shortcut $(x, y) \in CS$ estimates its associated utility u_{xy} by adding it to the graph and computing the difference between the sums of all-pairs shortest path lengths in the original graph G and the augmented graph G_{aug}. Then, the candidate shortcut (x, y) is removed from the graph and the process is repeated for the rest ones. In order to determine the best shortcut for augmentation, the method needs to compute the utility of all candidate shortcuts of a graph.

The worst-case time complexity of the greedy method is $O(k \cdot n^2 \cdot n(nlogn + m))$: the utility computation of a single candidate shortcut has time complexity $O(n(nlogn + m))$ (equivalent to running the Johnshon's algorithm for all-pairs shortest paths one time [68], using Fibonacci heaps [50] in the implementation of Dijkstra's algorithm), there are $|CS| = n(n-1)/2 - m = \sim n^2$ candidate shortcuts that need to be evaluated in order to pick the best one, and to find the best k, one would need to repeat the procedure k times.

4.5.2 Relaxation of the Greedy Method

It is possible to drop k if instead of computing the best candidate shortcut, one at a time, we use the already computed shortcut utility scores from the first iteration and determine the best k shortcuts all at once (*batch processing*). While this variant is expected to be faster (time complexity is $O(n^2 \cdot n(nlogn + m))$), it might affect the overall performance of the shortcuts added, as the addition of a shortcut might lower the utility of a subsequent one. We discuss this tradeoff in the experimental evaluation section.

4.5.3 Path Screening Method

We describe a novel method that can speed up the process of computing the utility of all candidate shortcuts for augmentation. The method works by decomposing the computation of the sum of the shortest path lengths into subproblems. It's easy to see, that in the general case the addition of a shortcut $(x, y) \in CS$ to G, alters the shortest path lengths of many (s, t) pairs with $s \in V$ and $t \in V$. We say that these (s, t) pairs *depend* on shortcut (x, y). However, many st-shortest paths do not depend on (x, y), and as such the computation of the shortcut utility doesn't need to consider them. Let $D_{st}(x, y)$ be the set of (s, t) pairs that depend on the shortcut (x, y). For each (s, t) pair it holds that:

$$\begin{cases} (s,t) \in D_{st}(x,y) & \text{if } d'(s,t) < d(s,t) \\ (s,t) \notin D_{st}(x,y) & \text{if } d'(s,t) = d(s,t) \end{cases}$$

Note that for the majority of shortcuts (x, y), the number of (s, t) pairs that actually depend on a shortcut is much smaller than the total number of node pairs in G (i.e., $|D_{st}(x, y)| \ll |V \times V \setminus E|$). This suggests large savings in the computation of the utility u_{xy}, since in fact we only need to sum up the differences in

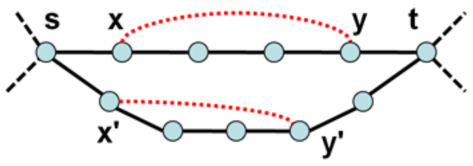


Figure 4.4: The (s,t) pair belongs in both $D_{st}(x,y)$ and $D_{st}(x',y')$, but only to $D_{st}^{SP}(x,y)$. It doesn't belong to $D_{st}^{SP}(x',y')$, as x' or y' are not nodes of the st-shortest path.

the shortest path length of a much smaller set of pairs $(s,t) \in D_{st}(x,y)$:

$$u_{xy} = \sum_{(s,t)\in D_{st}(x,y)} (d(s,t) - d'(s,t)) \tag{4.4}$$

Next, we present a theorem that states that given a shortcut edge of a fixed range δ in G, it is more beneficial to attach this shortcut on nodes of an existing st-shortest path. We use this theorem to further decompose u_{xy} in (4.4).

Theorem 5. *Let $G(V,E)$ be an undirected graph and let p_s be a shortest path and p_a be an alternative (i.e., non-shortest) path between nodes $s \in V$, $t \in V$. Assume an addition of a new shortcut $\{(x,y) : R(x,y) = \delta, \delta \leq d(s,t)\}$ in G. Then it holds that $d'^{p_s}(s,t) < d'^{p_a}(s,t)$, if both $x \in V$, $y \in V$ are nodes traversed by p_s in G, where $d'^{p_s}(s,t)$, $d'^{p_a}(s,t)$ are the lengths of the shortest paths p_s, p_a in G_{aug}, respectively.*

Proof. Let p_s represent a shortest path in G and assume that one (or both) of $x \in V$, $y \in V$ is not traversed by p_s (or any of the shortest paths connecting s, t). Now, let p_a represent a non-shortest path in G that connects $s \in V$, $t \in V$ and traverses both x, y. Then, (since p_a is not a shortest path) it holds that:

$$d^{p_a}(s,t) > d^{p_s}(s,t) \tag{4.5}$$

where $d^{p_a}(s,t)$, $d^{p_s}(s,t)$ are the lengths of the paths p_s and p_a in G. Now assume addition of a shortcut $\{(x,y) : R(x,y) = \delta\}$ in G. Then, it holds that:

$$d^{p_a}(s,t) = d'^{p_a}(s,t) + (\delta - 1) \tag{4.6}$$

where $d'^{p_a}(s,t)$ is the length of the path connecting (s,t) in the augmented graph G_{aug}. This is due to the constraint that $R(x,y) = \delta$. By (4.5) and (4.6), it follows that:

$$d'^{p_a}(s,t) + (\delta - 1) > d^{p_s}(s,t) \tag{4.7}$$

On the contrary, it's easy to see that if both x, y were traversed by p_s, then after addition of an edge $\{(x,y) : R(x,y) = \delta\}$ in G it would hold that:

$$d^{p_s}(s,t) = d'^{p_s}(s,t) + (\delta - 1) \tag{4.8}$$

By (4.7) and (4.8), it follows that:

$$d'^{p_a}(s,t) > d'^{p_s}(s,t) \tag{4.9}$$

This concludes our proof. □

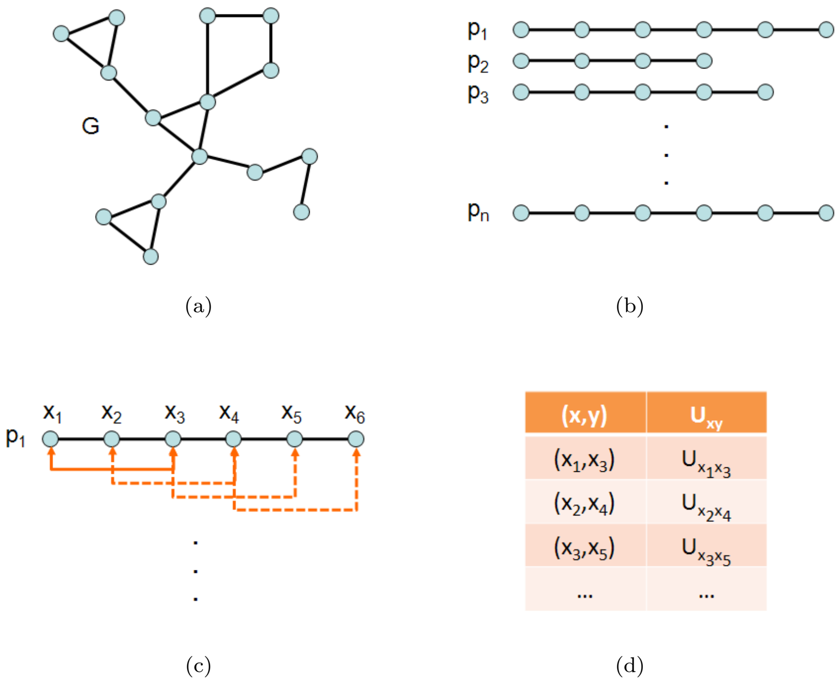
Theorem 5 states that given a shortcut edge of a fixed range δ in G, it is more beneficial to attach this shortcut on nodes of an existing st-shortest path. As such, it constrains the set of (s,t) pairs that depend on a candidate shortcut (x,y) to be (s,t) pairs that their shortest paths in G traverse both nodes x and y. Let this set of (s,t) pairs that depend on a candidate shortcut (x,y) be $D_{st}^{SP}(x,y)$ (see Figure 4.4). Then, we can further decompose u_{xy} in (4.4) by:

$$\begin{aligned} u_{xy} &= \sum_{(s,t)\in D_{st}^{SP}(x,y)} ((d(s,x) + d(x,y) + d(y,t)) \\ &\qquad - (d(s,x) + d'(x,y) + d(y,t))) \\ &= \sum_{(s,t)\in D_{st}^{SP}(x,y)} (d(x,y) - d'(x,y)) \end{aligned}$$

Note that the shortest path length $d(x,y)$ represents the range of (x,y) in G, $R(x,y)$. Moreover, the shortest path length $d'(x,y)$ represents the length of the newly added shortcut in G_{aug}, which is constant ($d'(x,y) = 1$). Therefore,

$$u_{xy} = \sum_{(s,t)\in D_{st}^{SP}(x,y)} (R(x,y) - 1) \tag{4.10}$$

Eventually, the utility u_{xy} depends on the range of (x,y) in G and the size of $D_{st}^{SP}(x,y)$, in accordance with our initial intuition.



(x,y)	U_{xy}
(x_1,x_3)	$U_{x_1x_3}$
(x_2,x_4)	$U_{x_2x_4}$
(x_3,x_5)	$U_{x_3x_5}$
...	...

Figure 4.5: Path Screening Method. (a) The input consists of a graph G, (b) All-pairs shortest paths are extracted from G, (c) A δ-size window (for varying δ) is slided over each shortest path, (d) Occurences of any candidate shortcut (x, y) found contribute to its utility u_{xy}.

Fast Assessment of Shortcut Utility Scores

We look to efficiently compute the utility score u_{xy} of any shortcut $(x, y) \in CS$ in G according to (4.10). Algorithm 3 describes the steps to solve the problem. It uses a modification of Johnson's Algorithm [68] that does not only provide the lengths of the all-pairs shortest paths, but also reconstructs and stores them in P. Then, for each shortest path $p \in P$ it determines what are the (x, y) shortcuts that could shorten the current path p by sliding a variable-size window of size δ over the path's nodes (see Figure 4.5). Note that δ takes values that range from $\delta = 2$ (shortcuts between non-adjacent nodes) to $\delta = l$, where l is the length of the current path p. It is easy to see that the maximum δ to be considered will be equal to the *diameter* D of G, where $D = max_{i,j} d(i, j)$ (the longest shortest path between any two nodes in G), so the range of values of δ is $2 \leq \delta \leq D$. As we slide the window over a path, the window's endpoints determine the x and y nodes of a candidate shortcut (x, y). Each time a shortcut (x, y) is encountered in a path p, an increment equal to $(\delta - 1)$ is contributed to its utility. The procedure continues over all paths $p \in P$ and at the end, for each (x, y) its utility u_{xy} is computed in accordance to (4.10). The time complexity of our path screening

ALGORITHM 3: Path Screening Method

```
Input: A Social Graph $G(V, E)$
Output: A Map between Candidate Shortcuts and their utility scores

$U \leftarrow \emptyset$;
$P = getAllPairsShortestPaths(G)$;
forall the $p \in P$ do
    $l = length(p)$;
    // Variable size of window
    for $\delta = 2$ to $l$ do
        // Slide a $\delta$-size window over nodes of $p$
        for $i = 0$ to $l - \delta$ do
            $x = p[i]$ ;                                        // $i$-th node of path $p$
            $y = p[i + \delta]$ ;                               // $(i + \delta)$-th node of path $p$
            if $(x, y) \notin U$ then
                $u_{xy} = (\delta - 1)$;
                add $< (x, y), u_{xy} >$ in $U$;
            else
                update $< (x, y), u_{xy} + (\delta - 1) >$ in $U$;
            end
        end
    end
end
return $U$
```

method is $O(n(logn + m)) + O(n^2\bar{L}(G)\bar{L}(G))$: it requires $O(n(logn + m))$ time to find the shortest paths in G (equivalent to running the Johnson's algorithm ones). Then, for each path ($O(n^2)$) needs to vary the δ-size of the window ($O(\bar{L}(G))$) and slide the window over the path p ($O(\bar{L}(G))$).

4.5.4 Path Screening Sampling-based Method

Algorithm 3 describes an efficient way to accurately compute the utility of each candidate shortcut u_{xy}. However, it is still not very practical for large graphs as it requires that all-pairs shortest paths can be efficiently computed ($O(n(logn + m))$) and then screened ($O(n^2\bar{L}(G)\bar{L}(G))$). In this paragraph, we describe a sampling-based variant of the path screening method for efficiently computing the utilities of the candidate shortcuts. Ideally, we would like to sample uniformly at random a few shortest paths from the set of all available ones. Then, we would feed them to our Algorithm 3 and compute estimates of the utility of each candidate shortcut based on that sample. However, uniform shortest path sampling is rarely feasible [69]. Instead, we sample uniformly at random a set of Q nodes $Q = \{q_1, ..., q_q\}$. Then, starting from a node q_i, $i = 1, ..., q$, where $q \ll n$, we execute the Dijkstra algorithm [39] and compute the single source shortest path tree from q_i to all nodes x in the graph. Now, instead of computing the utility of a candidate shortcut by operating over all-pairs shortest paths, we operate only on the set Q of shortest paths returned by the

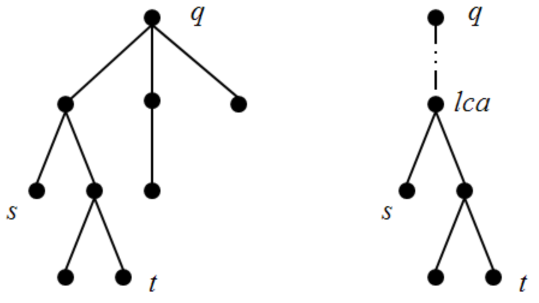


Figure 4.6: Example shortest path trees (SPT). (a) SPT rooted at $q \in Q$, (b) Lowest common ancestor (lca) in a SPT.

sample:

$$u_{xy}^{Q} = \sum_{s \in Q, t \in V} d(s,t) - d'(s,t)$$

The way we perform the sampling reduces the search space where candidates can be found (as not all shortest paths become available) and is likely to introduce bias. The main reason for the bias is that some paths are more likely to exist in the shortest path trees of sampled sources [69]. However, in our setting, we are interested in finding only a few candidate shortcuts that have very high utility. Note that edges with high betweenness centrality [113] in G are more likely to be in the sample as by definition they are edges that are traversed by many st-shortest paths. And, since important candidate shortcuts are shortcuts (x, y) that their endpoints x, y are found in many st-shortest paths, it is expected that candidate shortcuts with high utility will be sufficiently represented in the single-source shortest path trees of the sampled nodes. Therefore, our path screening method will be able to detect them. We assess the effect of the bias in the experimental evaluation. The rest of the algorithm remains the same. The point is that the above computation is relatively efficient, since it doesn't need to compute all-pairs shortest paths. Instead we compute q times the single-source shortest-path Dijkstra. Thus, the time complexity of our sampling-based path screening method is $O(q(logn + m)) + O(qn\bar{L}(G)\bar{L}(G))$, where $q \ll n$.

Sampling-based Method Estimation

In this paragraph, we formally describe the sampling-based estimation method. First, we employ a commonly used graph embedding technique, called *reference node embedding* [57, 81, 130], to show that the distances between any two nodes in a graph can be estimated if a small set of reference nodes are selected (each serving as the root of a shortest path tree (SPT)) and then the distances between these nodes to all other nodes in a graph are computed.

Definition 6. *Let T be a rooted tree with n nodes. The lowest common ancestor (lca) of two nodes u and v in T is the shared ancestor of u and v that is located farthest from the root.*

Theorem 6. *Given a set of reference nodes $Q = \{q_1, q_2, ...q_q\}$, $\forall s, t \in V$ it holds that:*

$$d(s,t) \geq \max_{q \in Q}\{|d(s,q) - d(q,t)|\} \tag{4.11}$$

$$d(s,t) \leq \min_{q \in Q}\{d(lca^q, s) + d(lca^q, t)\} \tag{4.12}$$

Proof. Given a set of reference nodes $Q = \{q_1, q_2, ...q_q\}$, our sampling-based method considers the shortest paths for the node pairs $\{(q, v) : q \in Q, v \in V\}$ by computing the single-source shortest path trees for every $q \in Q$. The shortest-path distance in graphs is a metric, and therefore it satisfies the triangle inequality. That is, given a pair of nodes (s,t), $\forall q \in Q$ the following inequalities hold (see Figure 4.6(a)):

$$d(s,t) \geq |d(s,q) - d(q,t)| \tag{4.13}$$

$$d(s,t) \leq d(s,q) + d(q,t) \tag{4.14}$$

By considering the lowest common ancestors, tighter bounds are feasible (see Figure 4.6(b)). $\forall q \in Q$ it holds that:

$$d(s,q) + d(q,t) = d(lca, s) + d(lca, t) + 2d(lca, q)$$

As $d(lca, q) \geq 0$ we have:

$$d(s, lca) + d(lca, t) \leq d(s,q) + d(q,t)$$

Consequently,

$$d(s,t) \leq d(s, lca) + d(lca, t)$$

By considering all reference nodes $q \in Q$, we have tighter bounds:

$$d(s,t) \geq \max_{q \in Q}\{|d(s,q) - d(q,t)|\} \tag{4.15}$$

$$d(s,t) \leq \min_{q \in Q}\{d(lca^q, s) + d(lca^q, t)\} \tag{4.16}$$

where lca^q is the least common ancestor (lca) of s, t in a SPT rooted at $q \in Q$. This concludes the proof. □

Then, we show that despite the fact that our sampling-based method picks shortcuts to add based on a small sample of shortest paths, the addition of these shortcuts in G will eventually shrink the shortest path length of *any* (s, t) pair.

Theorem 7. *Let $d^i(u,v)$ denote the distance between $u \in V$, $v \in V$ after i iterations, where in each iteration a single shortcut is added in G. Then, $\forall s,t \in V$ and sufficiently large n it holds that:*

$$d^n(s,t) < d(s,t) \tag{4.17}$$

Proof. Assume addition of a shortcut in G. Then, $\forall u,v \in V$ in G' it holds that:

$$d'(u,v) \leq d(u,v) \tag{4.18}$$

since the new shortcut might shorten the distance between $u \in V$, $v \in V$ or not. Now, let $d^i(u,v)$ denote the distance between $u \in V$, $v \in V$ after i iterations, where in each iteration a single shortcut is added in G. Then, it holds that:

$$\begin{aligned} d^1(s,t) &\leq d^1(s,q) + d^1(q,t) = \alpha_1 \\ d^2(s,t) &\leq d^2(s,q) + d^2(q,t) = \alpha_2 \\ &... \\ d^n(s,t) &\leq d^n(s,q) + d^n(q,t) = \alpha_n \end{aligned}$$

and because of (4.18) it holds that:

$$\alpha_1 \geq \alpha_2 \geq ... > ... \geq ... > ... \geq \alpha_n$$

Note that with a sufficient large n there will eventually be instances where the equality can be dropped as a new edge will shrink one of the distances $d(s,q)$ or $d(q,t)$. Finally, after n iterations, the distance between s and t will fall below a constant α:

$$d^n(s,t) \leq d^n(s,q) + d^n(q,t) = \alpha_n \leq \alpha$$

where $\alpha \ll d(s,t)$. Therefore,

$$d^n(s,t) < d(s,t)$$

This concludes the proof. □

Theorem (6) states that for any st-shortest path not included in the sample, alternative shortest paths exist that can be used to estimate its length and this estimation is bound. Theorem (7) states that by adding shortcuts that aim to shorten the length of specific shortest paths in the sample, eventually we manage to shorten many more st-shortest path lengths in the original graph, and thus successfully shrink the characteristic path length of the graph. The premise is that our sampling-based method is not only fast, but it also succeeds in approximating the solution found by applying the path screening method on all-pairs shortest paths. We experimentally validate our claims and empirically show the trade-off between efficiency and quality degradation in the evaluation section.

4.5.5 Summary of Methods

Table 4.1 provides a summary of the worst-case time complexity of the various methods discussed for easy reference.

Table 4.1: Comparison of the Worst-case Time Complexity of the Various Methods

Method	Worst-case Time Complexity
Greedy	$O(k \cdot n^2 \cdot n(nlogn + m))$
Relaxation of Greedy	$O(n^2 \cdot n(nlogn + m))$
Path Screening	$O(n(logn + m)) + O(n^2\bar{L}(G)\bar{L}(G))$
Path Screening Sampling-based	$O(q(logn + m)) + O(qn\bar{L}(G)\bar{L}(G))$, where $q \ll n$

4.6 Experimental Evaluation

In this Section, we evaluate methods that suggest shortcuts for addition in a graph; they take as input a graph G, an integer k and an augmentation policy $\mathcal{P}$ and return a set S of the best shortcuts for augmenting G. Following, we describe the methods that we employ in the experimental evaluation. These reflect the case of *network-centric policy.* In the case of the *node-centric policy* the procedure is similar, but the method has to be repeated until the best k shortcuts for *each node* in G have been determined. The following methods are considered:

- *Greedy-S*: The greedy method described in paragraph 4.5.1.
- *Greedy-B*: The *batch variant* of *Greedy-S* described in paragraph 4.5.2.

- *PS*: Our path screening method described in paragraph 4.5.3.
- *PSS*: Our sampling-based variant of *PS* described in paragraph 4.5.4.
- *Random*: A baseline method that selects k shortcuts uniformly at random to add in G, inspired by [104].
- *Mutual Friends Intersection (MFI)*: This is another sensible baseline method that selects shortcuts based on the number of adjacent nodes that two nodes share. This method is inspired by the common practice of online social networking services to suggest new friendships (shortcuts) to users based on the number of their mutual friends. If $N(x)$ is the neighborhood of node x and $N(y)$ is the neighborhood of node y, then the utility score of a shortcut (x, y) can be expressed as the size of the intersection of the neighborhoods of x and y.

Network Topologies: For the needs of our experimental evaluation we consider both *real* and *synthetic* network topologies. The real network topologies, represent connected[1], undirected, unweighted networks (*dolphins* [100], *netsc* [114], *yeast* [26]). The synthetic network topologies are of two types. Networks of the first type simulate a preferential attachment network topology based on the Barabási-Albert network model [20]. The number of nodes and edges in these networks have been selected so as to resemble trees; trees are typically sparse (less dense) and the shortest paths between pairs of nodes are unique. We experiment with three of them, a small (*ba-s*), a medium (*ba-m*) and a large (*ba-l*). Networks of the second type represent tightly-knit communities (clusters). These networks are typically more dense and have the characteristic that for a large number of pairs of nodes there are many alternative shortest paths that connect them. We experiment with three of them, a small (*comm-s*), a medium (*comm-m*) and a large (*comm-l*). Table 4.2 provides a summary of the networks we consider[2]. For each network we report the number of candidate shortcuts, the number of candidate shortcuts that represent Friends of Friends (FoFs), the network density ($Density = 2|E| \setminus (|V|(|V| - 1))$) and its characteristic path length $\bar{L}(G)$.

Evaluation Metrics: We assess the performance of the various methods according to *accuracy* and *efficiency* measures. *Accuracy* concerns the degree of usefulness of adding a specific set of shortcuts, in terms of optimizing a network property. To assess the accuracy of the various algorithms we define a *gain* metric,

[1]In cases where a network is not connected, we extract its largest connected component and operate on it.

[2]Regarding the choice of the graph sizes considered, one should bear in mind that in this problem we are interested in the evaluation of the utility of missing edges (shortcuts) of a graph, which in our cases range from a few thousands and hundreds of thousands to millions, as is depicted in Table 4.2.

Table 4.2: Networks

Name	#Nodes	#Edges	**#CS**	#FoFs	Density	$\bar{L}(G)$
dolphins	62	159	**1,732**	448	0.084	3.357
netsc	379	914	**70,717**	2,916	0.012	6.042
yeast	2,224	7,049	**2,464,927**	73,479	0.003	4.376
ba-s	50	49	**1,176**	99	0.040	4.641
ba-m	500	499	**124,251**	1,685	0.004	7.398
ba-l	1,000	999	**498,501**	3,501	0.002	8.132
comm-s	50	127	**1,098**	197	0.103	3.786
comm-m	500	1,469	**123,281**	6,589	0.012	6.241
comm-l	1,000	2,959	**496,541**	14,620	0.006	6.764

which expresses the percentage change on the *characteristic path length*, $\bar{L}$, in the original graph G and in the augmented graph G_{aug}. Formally, the gain is given by:

$$Gain = \frac{|\bar{L}(G) - \bar{L}(G_{aug})|}{\bar{L}(G)} \times 100$$

Efficiency concerns the time performance of a shortcut edge addition method. To assess the time performance of a method we measure the period of its execution time *in milliseconds*.

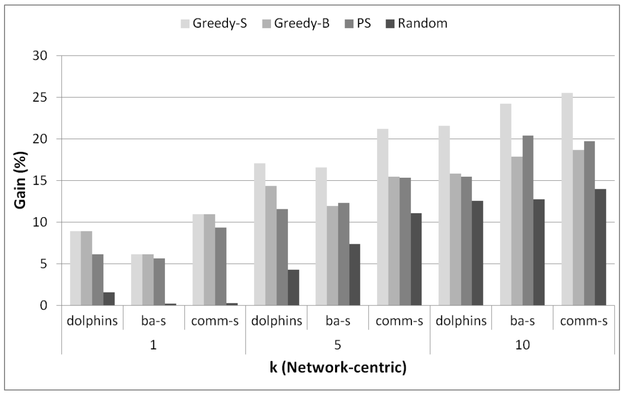
Computing Environment: All our experiments have been conducted on a Dell XPS 8700 desktop running 64-bit Windows 8.1. The desktop is equipped with an Intel Core i7-4770 CPU 3.40GHz with 24.0GB of RAM. The Intel Core i7-4770 processor has four cores and uses hyper-threading, so it appears to have eight logical CPUs. Our algorithms have been implemented using the Java programming language version 8.0 (build 1.8.0_31) and the experiments are executed on a Java HotSpot 64-Bit Server VM (build 25.31-b07, mixed mode).

4.6.1 Evaluation of the Path Screening Method

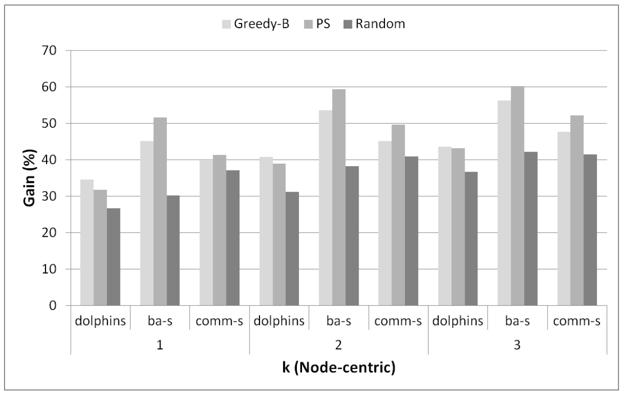
Our experiments are grouped into three sets. For the methods that introduce randomness or are based on computing shortest path trees (i.e., all methods other than *Greedy-S*), single runs are susceptible to introduce bias due to randomness. We alleviate this bias by repeating the experiment many times (x10) and averaging together the results. Then, we report average values of accuracy and efficiency.

PS Performance / Network-centric Policy

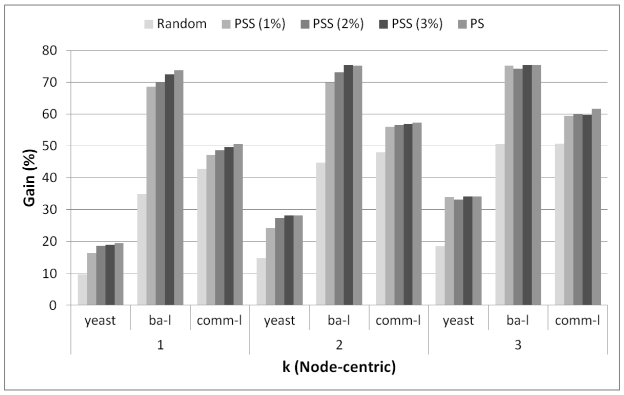
We evaluate the *accuracy* and *efficiency* of our path screening method against the greedy methods, for a varying range of network topologies and number of edges k to be added. More specifically, we experiment



(a) PS Accuracy / Network-centric



(b) PS Accuracy / Node-centric



(c) PSS Accuracy

Figure 4.7: Accuracy of our Path Screening method (PS) and its sampling-based variant (PSS)

with *dolphins*, *ba-s* and *comm-s* and we add 1, 5 and 10 shortcuts. For this set of experiments we had to limit our experiments on very small networks and on *network-policy*, because *Greedy-S* is extremely slow (as we discuss below), so it is not appropriate for larger graphs or cases of *node-centric policy* (because the set of candidate shortcuts that need to be evaluated is very large).

Accuracy: Figure 4.7(a) presents the accuracy results for the various networks. In all instances, *Greedy-S* performs better than any other method. This is because *Greedy-S* evaluates any of the k edges one at a time. Even if this method is still suboptimal (due to making locally optimal choices), it ensures that at each iteration the correct utility for each of the subsequent candidate shortcuts is computed, taking into consideration the previously added ones. As such, it avoids adding shortcuts that have overlapping utility. On the other hand, *Random* performs poorly in all instances, as expected, and forms a baseline for the performance of the other methods. The accuracy performance of the rest two methods, *Greedy-B* and *PS* is comparable. They both perform only slightly worse than *Greedy-S*, but outperform *Random*. Overall, the performance of *PS* is always comparable and sometimes better than that of the *Greedy-B* method. On another note, the difference between the performance of *Greedy-S* and the other methods is becoming more evident as k is increasing. This is to be expected; as all the other methods add shortcuts on a batch way, the larger the number of added shortcuts is, the larger the likelihood that these shortcuts share overlapping *st*-shortest paths will be, rendering the overall gain to be smaller.

Efficiency: Table 4.3 presents the efficiency results for the various networks. Note that, for all methods other than the *Greedy-S* it is sufficient to report only one value for each network, since these methods will evaluate the utility of all shortcuts only once. On the other hand, for the case of *Greedy-S* we report values for all different instantiations of k (i.e., *Greedy-S-1*, *Greedy-S-5*, *Greedy-S-10*). It becomes obvious from Table 4.3 that *Greedy-S* is extremely inefficient, as it adds one shortcut at a time. In fact, *Greedy-S* will be (almost) k times slower than *Greedy-B*. More importantly, our path screening method, *PS*, outperforms *Greedy-B* by *a multitude of times* (i.e., x164, x77 and x93 times faster, respectively). The gain in efficiency is due to the fact that, while *Greedy-B* computes the utility of all shortcuts, *PS* looks for shortcuts that connect nodes that lie on existing shortest paths. As such, our method suggests huge savings in the computation of shortcut utility.

PS Performance / Node-centric Policy

We evaluate the accuracy and efficiency of our path screening method *PS* against *Greedy-B* and *Random*, for a varying range of network topologies. More specifically, we experiment with *dolphins*, *ba-s*, *comm-s* and we ask to add 1, 2 and 3 shortcuts, following the *node-centric policy* (i.e., for a graph of n nodes, we ask to find n, $2n$, $3n$ shortcuts respectively). We had to constrain our tests to smaller networks due to the inefficiency of the *Greedy-B* method.

Table 4.3: PS Efficiency (msec) / Network-centric

Method	dolphins	ba-s	comm-s
Greedy-S-1	7,575	2,919	3,793
Greedy-S-5	33,495	10,830	14,050
Greedy-S-10	65,435	20,295	27,849
Greedy-B	7,729	3,004	3,849
PS	**47**	**39**	**41**
Random	1	1	1

Table 4.4: PS Efficiency (msec) / Node-centric

Method	dolphins	ba-s	comm-s
Greedy-B	7,718	3,033	3,829
PS	**47**	**31**	**32**
Random	1	1	1

Table 4.5: PSS Efficiency (msec) / Node-centric Policy

Method	yeast	ba-l	comm-l
Random	250	187	181
PSS (1%)	**490**	**157**	**172**
PSS (2%)	**848**	**344**	**282**
PSS (3%)	**1,179**	**437**	**422**
PS	16,475	4,365	4,764

Accuracy: Figure 4.7(b) presents the accuracy results for the various networks. In all instances, *PS* and *Greedy-B* perform much better than the *Random* method. It is also easy to see that as k increases the gain increases as well. This is to be expected; the larger the number of shortcuts added in the network, the lower is the average shortest path of the network and therefore the larger the gain would be. Overall, the performance of *PS* is comparable to *Greedy-B* and always better than that of the *Random* algorithms.

Efficiency: Table 4.4 presents the efficiency results for the various networks. Again, our path screening method, *PS*, outperforms *Greedy-B* by a *a multitude of times* (i.e., x164, x97 and x119 times faster, respectively).

PSS Performance

We have demonstrated that *PS* is extremely faster than *Greedy-S* and *Greedy-B*, while maintaining high levels of accuracy. However, the applicability of *PS* can be limited by the need to compute all-pairs shortest paths in the graph. In this set of experiments, we evaluate the accuracy and efficiency of *PSS*, a sampling-based variant of our path screening method.

Accuracy: We assess the performance of *PSS* against the *PS* method that serves as the ground truth and the *Random* that serves as baseline for our sampling method. We experiment in large networks (*yeast*, *ba-l*, *comm-l*) and for varying k, as well as, varying sample size. In particular, for each network, we experiment with sample sizes of 1%, 2% and 3% of the total nodes in the network that define the methods *PSS (1%)*, *PSS (2%)* and *PSS (3%)* respectively, and at each run we consider adding 1, 2 and 3 shortcuts, following the *node-centric policy.* Figure 4.7(c) presents the results for the various instances. In all cases, the sampling accuracy increases with the sample size. This is expected, as more shortest paths are evaluated in larger samples. Moreover, *PSS* has always a better accuracy than *Random* and is only slightly lower than that of *PS*.

Efficiency: Table 4.5 presents the efficiency results for the various methods in varying networks. The execution time of *PS* represents the time of the slowest method. For example, while *PS* requires around 4.76sec for finding shortcuts to add in the *comm-l* network, *PSS (1%)* requires less than 0.2sec (i.e., around x27 times faster). At the same time the accuracy of *PSS (1%)* is comparable to that of *PS* for all k (as depicted in Figure 4.7(c). Overall, PSS can be used to boost the performance of *PS*, without duly affecting its accuracy.

PSS Scalability

We have demonstrated that *PSS* is a multitude of times faster than standard approaches (*Greedy-S* and *Greedy-B*) and the non-sampling version of our algorithm (*PS*), while maintaining high levels of accuracy. In this set of experiments, we evaluate the scalability of *PSS* in much larger networks, following the *node-centric policy*, where $k = 1$ (i.e., we seek to add a new shortcut to each node). For this set of experiments, the rest of the algorithms we have discussed cannot be evaluated, as they are not designed to scale. In particular, we employ a few of the larger network data sets freely available online[3], including an email communication network (*enron*), three co-authoring collaboration networks (*astro-ph*, *cond-mat*, *dblp*), four social networks (*facebook*, *twitter*, *youtube*) and a network of co-purchased products (*amazon*). Details about these networks can be found in Table 4.6 and in [94]. To allow our *PSS* algorithm to scale, we control the *sample size*; for the smaller of these graphs we considered a sample of 1% of the network nodes (*PSS (1%)*) and for the larger of these graphs we considered a sample of 0.01% of the network nodes (*PSS (0.01%)*). Recall that, as we discussed in section 4.5.4, each of the nodes of the sample is used as a *reference node* (or root) on which we

[3]snap.stanford.edu/data/

Table 4.6: Large Networks

Name	#Nodes	#Edges	**#CS**	Density
facebook	4,039	88,234	**8.1E+06**	1.1E-02
astro-ph	14,845	119,652	**1.1E+08**	1.1E-03
enron	33,696	180,811	**5.6E+08**	3.2E-04
cond-mat	36,458	171,736	**6.6E+08**	2.5E-04
twitter	81,306	1,342,310	**3.3E+09**	4.1E-04
dblp	317,072	1,049,779	**5.0E+10**	2.1E-05
amazon	334,860	925,838	**5.6E+10**	1.6E-05
youtube	1,134,791	2,986,629	**6.4E+11**	4.6E-06

Table 4.7: PSS Scalability (msec)

Method	facebook	astro-ph	enron	cond-mat
PSS (1%)	**2,563**	**26,002**	**114,360**	**175,620**

Table 4.8: PSS Scalability (msec)

Method	twitter	dblp	amazon	youtube
PSS (0.01%)	**12,845**	**247,829**	**643,749**	**2,976,239**

execute the Dijkstra algorithm and compute the single source shortest path tree from that node to all other nodes in the graph. Then, these shortest path trees are processed in order to determine the best shortcuts to add in the graph. Table 4.6 also lists the number of all candidate shortcuts in the graph and the density of each of these large networks. It is important to note that the number of candidates shortcuts can be really big; in the case of the *youtube* network this is approximately *640 billion candidate shortcuts*.

Efficiency: Table 4.7 and Table 4.8 present the time efficiency results for the *PSS (1%)* and the *PSS (0.01%)* method, respectively, for a number of large networks. It is clear that our *PSS* method can scale to large networks very well, while standard methods are not even applicable. For example, *PSS (0.01%)* needs less than 50min (i.e., 2,976,239 msec) to find which shortcuts to add in the *youtube* network that consists of ∼1.1 million nodes, ∼3 million edges and a total of ∼640 billion candidate shortcuts. Overall, PSS can be used to boost the performance of *PS*.

Accuracy: Accuracy results for this set of experiments are excluded, as they would require to report on the performance of our algorithms against alternatives methods. But, as discussed earlier, alternative methods do not scale well and therefore it is not possible to apply them on networks of this size.

4.6.2 The Cascade Effect

We have demonstrated that our path screening method outperforms the accuracy of sensible baselines and at the same time is extremely efficient. When we introduced the problem we made a connection between the efficiency of a network to propagate information and its network structure. In this set of experiments we evaluate the impact of our methods in the cascade process. In particular, we evaluate the impact on *the graph conductance*, on *the markov chain mixing time* and on *the information propagation spread.*

Impact on Graph Conductance

We evaluate the impact of our methods in increasing the conductance of a graph. More specifically, we experiment with both real and synthetic networks of small, medium and large size. For each case, we add 1, 5 or 10 shortcuts using the *Random*, *PS* or *Greedy-B* method, following the *network-centric augmentation policy.* Figure 4.8 shows the effect of each method on the original graph's conductance. Note that we had to constrain the computations of the *Greedy-B* method to only small networks due to its inefficiency. The plots indicate that the graph conductance (minimum conductance of any cluster size) is increased more when we augment the graph using our path screening method *PS*, than the *Random* baseline and (almost) always this increase is only slightly smaller than that achieved by *Greedy-B.* Furthermore, conductance is increasing with k; this is depicted by the larger increase in the cases of adding 5 or 10 shortcuts, for all networks. Moreover, this increase is more expressed in the case of the community networks *comm-s, comm-m, comm-l.* This is to be expected, as these networks consist of only a few cross-cutting edges.

Impact on Markov Chain Mixing Time

We evaluate the impact of our methods in decreasing the mixing time of a graph using the same parameter settings as in 4.6.2. We report the theoretical mixing time of the fastest random walk in a network, in terms of the number of steps (i.e., random walk length) before it converges (i.e., reaching a specific relative point-wise threshold ϵ). For the needs of our experiments we set $\epsilon = 0.01$. It is easy to see that a smaller threshold would end up in larger savings and a larger one in smaller savings. Figure 4.9 shows the effect of each method on the original graph's mixing time. The plots indicate that the mixing time is decreasing more when we augment the graph using our path screening method *PS*, than the *Random* baseline and (almost) always this decrease is comparable to the one caused by *Greedy-B.* The trend is consistent for *yeast*, but does not appear properly due to small values. Furthermore, the mixing time is decreasing with the number

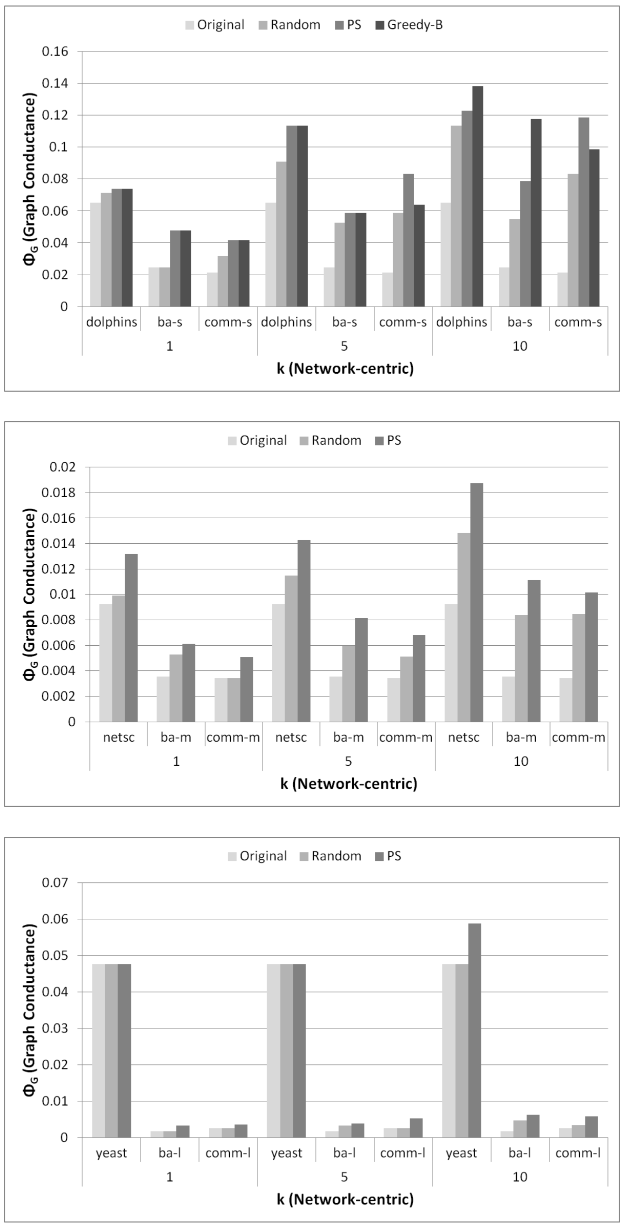
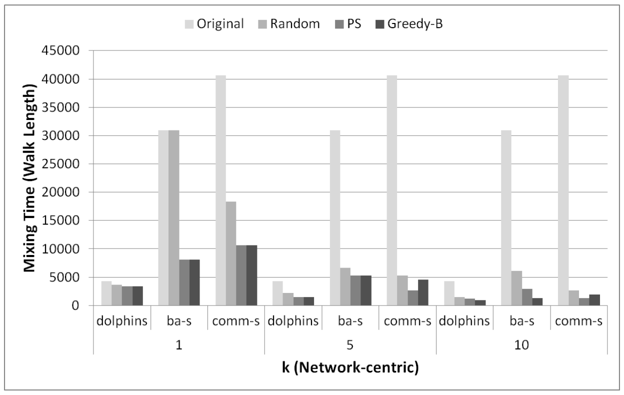
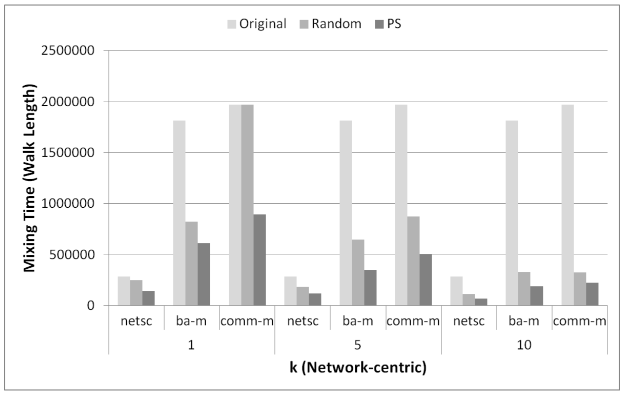


Figure 4.8: Impact on Graph Conductance on small (top), medium (middle) and large (bottom) graphs

of shortcuts added and this decrease is, as with the graph conductance, more expressed in the case of the community-like networks.





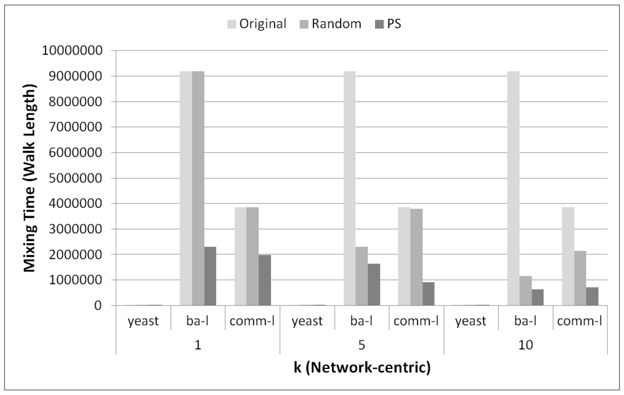


Figure 4.9: Impact on Markov Chain Mixing Time on small (top), medium (middle) and large (bottom) graphs

Impact on Information Propagation Spread

We have demonstrated that our method can dramatically affect properties of the graph that are critical for characterizing its ability to carry on propagation processes, such as the graph conductance and its mixing time. In this set of experiments, we evaluate the impact of our methods on the *cascade spread*; the number of nodes reached by a propagation process. It is important to note that our methods are generic and do not depend on a specific propagation model. Consequently, we first present a *simple generic propagation model* that is used in the evaluation and then we describe the evaluation in detail.

Propagation Model: We employ a basic threshold propagation algorithm based on an individual-level natural model of direct-benefit effects in networks due to Stephen Morris [108]. The underlying consideration of this model is that a node has a certain social network neighbors and the benefits to adopt a new behavior increase as more and more of these neighbors adopt it. Conforming to the model, a set of initiator nodes I in the network has adopted a behavior. At each round, a node should adopt the new behavior once a sufficient proportion of its neighbors have done so. We model the decision making using a simple threshold rule; if a fraction of at least T of a node's neighbors have adopted then it should, too. The semantics of the threshold are intuitive; the smaller the T is the more attractive the cascading behavior is, as you need fewer of your neighbors to engage before you do. Nodes that have already adopted the behavior cannot switch back. The propagation process terminates when no change is detected in a specific round (no new node has adopted). In a sense, the number of nodes that have adopted in the end of the propagation process represents a *sphere of influence* (SOI), of the initiator set I.

Given a graph, first we add new edges based on our path screening method, PS, and then add new edges based on MFI. For the needs of this experiment, we had to constrain the set of candidate shortcuts to represent FoFs ($CS = \{(x,y) : R(x,y) = 2\}$). We experiment with three medium size graphs, *netsc*, *ba-m* and *comm-m*, and ask to add one shortcut to each node (*node-centric policy*, $k = 1$) for variable sizes of the initiator set I and values of the threshold T. For each case, we simulate the propagation model on the augmented network several times (x100), each time computing the number of nodes that have adopted, and averaging over all instances. In the end, we report the average size of the sphere of influence, $SOI(\%)$, as percentage of the total number of nodes in the network for finer representation:

$$SOI(\%) = \frac{|SOI|}{|V|}$$

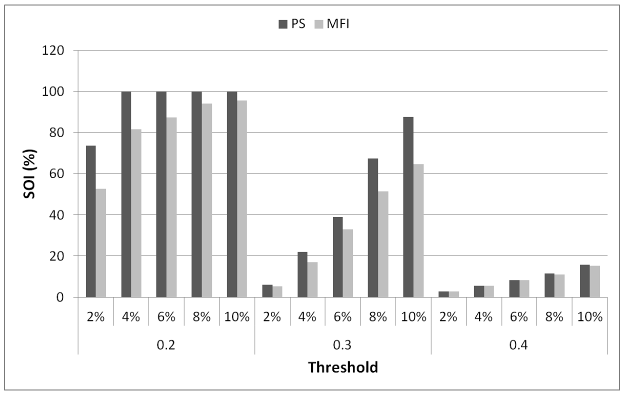
Figure 4.10 presents the results for the various networks. In all instances, PS outpeforms MFI. Note that the spheres of influence of the various initiator sets I (2%, 4%, 6%, 8%, 10% of the total number of nodes) in networks augmented by our PS method, are always larger than or equal to the ones in networks augmented by MFI - an increase that ranges between 0% and ∼80%. This behavior is demonstrated in all three networks, but the impact is larger in *comm-m* and *netsc*; the two networks that have a more profound clustering structure. On another note, as T varies (0.2, 0.3, 0.4) the sizes of the spheres of influence are getting smaller. This is expected as larger threshold indicates a less attractive behavior that is difficult to be adopted by nodes, so irrelevant of the structure of the augmented network, the behavior cannot easily cascade.

4.7 Related Work

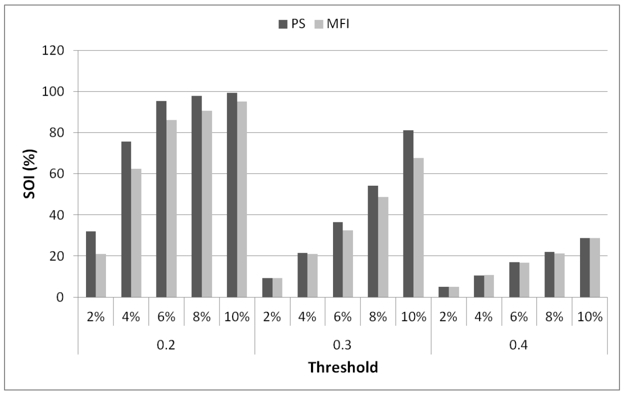
We have tried to provide pointers to work related to our research throughout the chapter. In this Section, we provide a more concrete coverage of related work not mentioned earlier. In particular, literature related to topics of graph augmentation, information propagation and link prediction and recommendation.

4.7.1 Graph Augmentation

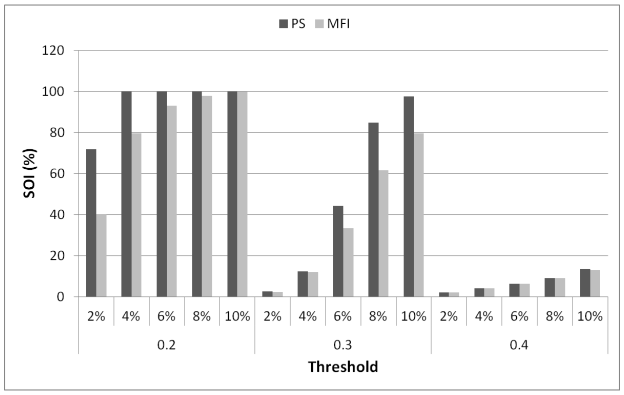
In traditional graph augmentation problems we ask to find a minimum-cost set of edges to add to a graph to satisfy a specified property, such as biconnectivity, bridge-connectivity or strong connectivity. These augmentation problems have been shown to be NP-complete in the restricted case of connected graphs [46], while a number of approximation algorithms with favorable time complexity have been shown to have constant worst-case performance ratios [49, 77, 118]. In one of the first works that considers the problem with a hard limit on the number of edges to be added, Meyerson and Tagiku [104] considered the problem of minimizing the average distance between the nodes. They obtained several constant-factor approximations using the k-median with penalties problem. They also improved the best known approximation ratio for metric k-median with penalties, to obtain better approximation factors for the other problems they considered. If α denotes the best approximation known for metric k-median with penalties, they presented an α-approximation for the single-source average-shortest-path problem, and a 2α-approximation for the general average shortest-path problem. In an another recent work, Demain and Morteza [38] studied the problem of minimizing the diameter of a graph by adding k shortcut edges, for speeding up communication in an existing network design. They developed constant-factor approximation algorithms for different variations of the problem and showed how to improve the approximation ratios using resource augmentation to allow more than k shortcut edges. Both [104] and [38] observe a close relation between the single-source version of the problem, where we want to minimize the largest distance from a given source vertex, and the well-known k-median problem. In the single-source version of the problem, a node may want to construct edges in order to minimize its distance from the other nodes. Adding k shortcuts to a graph to minimize its characteristic



(a) netsc



(b) ba-m



(c) comm-m

Figure 4.10: Impact on Information Propagation Spread

path is also the topic of [125] (self-reference). Laoutaris et al. [88] studied the *bounded budget connection (BBC) game*, where nodes in a network have a budget for purchasing links with the objective to minimize their average distance to the other nodes. A version of the problem for minimizing the maximum distance to the other nodes was considered as well. In a similar context, Zhou et al. [151] develop network engineering algorithms that aim to provide performance gains of random walk simulations over networks by reducing their mixing time.

The problem of interest in this research, is not the same as the above research. In traditional graph theory, connectivity (biconnectivity, triconnectivity, etc.) of a finite undirected graph refers to the minimum number of disjoint paths that can be found between any pair of vertices, as characterized by *Menger's theorem* [103] and generalized by the *max-flow min-cut theorem* [120]. In our research, the focus is on bringing nodes closer to each other, a connectivity property that is better expressed as *the characteristic path length* of a graph. It is also different to the theoretical approach of [104] as the suggested method operates by always attaching shortcuts to a single node, leading to a star network topology; in our problem we need to be able to control where shortcuts are added in the network. In the rest of the approaches, variations of the problem of interest are considered, where either the context of the nodes needs to be known (assumed) or the emphasis is on optimizing network properties different to its characteristic path length.

4.7.2 Maximizing Information Propagation

The idea of making structural changes in a network to enable information to travel easier from a node to another is closely related to the idea of finding influential nodes in a network to maximize information spread [30, 76, 82, 131] or detecting outbreaks in a network [93]. Manipulating edges to control information spread is also the topic in [144]. In the work of Richardson and Domingos [131] the problem of finding the most influential set of k nodes in a network was introduced, motivated by viral marketing. Viral marketing refers to marketing techniques that aim to increase a marketing objective (such as brand awareness) through replicating an epidemic process in social networks, similar to the diffusion of innovations. It is useful in applications that employ the diffusion process, such as maximizing the spread of influence in a social network [76] and early detection of outbreaks in a network [93].

The problem of interest in this research, is not the same as the problem of finding influential nodes in the network, as we focus on importance of edges and not nodes. More importantly, we do not make any assumption about the context of the nodes or factors that can affect an information propagation process or model (e.g., set of initiators or early adopters, how influential is a node to its neighbors, how susceptible is a node to its neighbors' influence and so on). On the contrary, our problem tries to improve on information propagation processes, by operating directly on the network structure.

4.7.3 Link Prediction and Recommendation

Our problem draws connections to the problem of *link prediction* [15, 97, 129], where the objective is to predict which new interactions among members of a social network are likely to occur in the near future. Backstrom and Leskovec [15] study the problem of link prediction and recommendation in social networks by performing supervised random walks on the network. Their method combines network structural information with rich node and edge attribute data to guide random walks. In a similar context, Tian et al. [143] study the *link revival* problem, where the objective is to turn already existing edges with a few interactions to be more active, so that the resulted connection will improve the social network connectivity. The authors develop an algorithm that explores local properties of a an evolving graph and try to recommend links per node at a time. The basic idea of their algorithm is to first consider graph snapshots in distinct time intervals, by monitoring the social interaction graph. Then, predict the probability with which a social interaction will happen in the future and if it has a low probability to happen, then they try to estimate the gain in the connectivity by adding this edge and consider it for recommendation. Finally, they recommend edges that have the larger gain in connectivity.

The problem of interest in this research, is not the same as the link prediction and recommendation problems described above. We are not interested in using historical data of interactions to predict future links or to revive existing but weak links. Instead we want to suggest new edges that minimize global structural properties of the network by operating solely on the network structure.

4.8 Conclusions

We considered the problem of adding a small number of shortcuts to a graph in order to optimize a connectivity property and improve its capacity of carrying on social processes. This is a quite novel, interesting and challenging problem. We proposed a novel method for quickly evaluating the importance of candidate shortcuts in a graph. More specifically, our method is able to approximate the number of shortest paths that would run along a candidate shortcut, if it was already in the network. Intuitively, our method approximates the betweenness centrality of these non-existing edges [113]. Betweenness centrality of an edge or simply edge betweenness is the number of shortest paths between pairs of nodes that run along it and is a measure of the influence of an edge over the flow of information among nodes, especially in cases where information flow over a network primarily follows the shortest available path [54, 113]. We proposed, as well, a sampling-based variant of the method that can be used to scale up the computation for larger graphs. Through experiments on various data sets, we demonstrated that our approach outperforms sensible baselines in both accuracy and efficiency, for a varying range of conditions.

Overall, the algorithms we described are *simple* to understand and implement, accurate, very *fast* and *general*, so they can probably be easily adopted in a variety of strategies. As such, we expect our methods to be beneficial in diverse settings and disciplines, ranging from social to technological networks. Below, we discuss a few ideas related to our research that aim to enlarge or prolong its scope. To some degree they present interesting extensions to our work and offer alternative views and insights.

4.8.1 Data Parallelism

It is important to note that further speedup of our methods can be achieved by technologies that distribute data and computing tasks across different parallel computing nodes. Data parallelism is based on distributing the data and computing tasks across different parallel computing nodes.

The first component of our main algorithm, based on a variation of Johnson's algorithm, is easy to distribute because it runs Dijkstra's algorithm n times, one for each node, which can be done in parallel. For example, all-pairs shortest paths can be computed by employing distributed versions of *breadth-first* or *depth-first* algorithms [12, 13, 29]. New programming models for processing large data sets with a parallel, distributed algorithm on a cluster, such as MapReduce [37] and Hadoop [135], can be used, as well, to compute the all-pairs shortest paths component of our main algorithm: the *Map()* procedure performs multiple parallel breadth-first searches simultaneously assuming source nodes $\{s_0, s_1, ..., s_n\}$ and instead of emitting a single distance, emits an array of distances with respect to each source, as well as, information about the nodes that constitute each path. Then, the *Reduce()* procedure selects a path with minimum length for each element from the array.

The second component of our main algorithm is to process the list of shortest paths (path screening) in parallel in order to compute the utility of each candidate shortcut. Algorithm 4 and Algorithm 5 show how to employ MapReduce in this step, as well: the *Map()* (Algorithm 4) performs the path screening of each shortest path in isolation and computes the utility of any candidate shortcut that occurs in the current path. Then, the *Reduce()* (Algorithm 5) performs a summary operation that computes the utility scores of candidate shortcuts among all shortest paths. We don't further elaborate on this issue due to lack of space and as it is orthogonal to our problem.

ALGORITHM 4: Data Parallelism Using MapReduce − *Map*()

```
Input: An integer $i$ representing a batch of shortest paths
Output: Partial utility of any candidate shortcut found in the $i$-th batch
for each shortest path in the $i$-th batch do
    $(x, y) \longleftarrow$ a candidate shortcut that occurs in a path;
    $\delta \longleftarrow$ the range of the shortcut in $G$ ($\delta = R(x, y)$);
    produce one output record $< (x, y), \delta >$;
end
```

ALGORITHM 5: Data Parallelism Using MapReduce − *Reduce*()

```
Input: Set of candidate shortcuts $(x, y)$
Output: The utility of each candidate shortcut
for each candidate shortcut $< (x, y), \delta >$ do
    accumulate in $U$ the sum of $\delta$;
end
produce one output record $< (x, y), U >$;
```

4.8.2 Disconnected Networks

We have constrained our discussion to connected networks. In the case of disconnected networks, one should first consider finding the connected components of the input network and then connect them with each other. Finding a smallest augmentation that connects an undirected graph is a well known problem, this is the problem of finding a spanning tree, with many efficient algorithms available.

4.8.3 Friend Suggestion Implications

The research also highlighted the fact that the global optimization problem of augmenting a graph by adding new edges can be seen as an alternative for *friend suggestion* in a social network. The premise is that in the augmented network users are more well connected and information can spread more efficiently in the network. In this paragraph we detail a methodology to evaluate such an alternative and relevant implications.

Current state-of-the-art friend suggestion algorithms are mainly based on the idea of suggesting friends with whom an individual shares a large number of mutual friends (similar to the *MFI* algorithm we presented in Section 4.6); that way the social network tries to maximize the probability of a new connection to occur (be realized). However, from a graph perspective, such an algorithm operates only locally by making the local network (cluster) more dense, but does not succeed in optimizing global graph properties (such as better information propagation). An alternative friend suggestion algorithm would try to blend ideas of our graph augmentation algorithms and the current-state-of-the-art by suggesting a *set of mixed friends* including:

- friends that make the local network more dense (coming from current state-of-the-art friend suggest algorithms), and
- friends that try to optimize global network properties (coming from our graph augmentation algorithms)

It becomes clear that many hybrid strategies can be followed to combine and evaluate the above algorithms. The evaluation should seek to optimize on the number of friends suggested by each algorithm and on the best way to rank the suggested friendships before presenting them to the end-user. We do not further elaborate on this issue due to lack of space and as it is orthogonal to our problem.

Another important aspect of the friend suggest problem is that while we suggest friends, we can not be sure that these friendships will be really materialized (i.e., accepted by the users). Therefore we have to reason in probabilistic terms. The probability model is based on the assumption that for each candidate shortcut (x, y) there is an associated probability $p(x, y)$ for it to be realized once suggested. The probability p can be given by any simple prediction method (e.g., it can be based on the number of mutual friends between two users). Then, the objective is to suggest a set of edges that will maximize the *expected utility*, if added in the graph.

Chapter 5

Searching in Online Social Networks

5.1 Introduction

The changing trends in the use of web technology that aims to enhance interconnectivity, self-expression and information sharing on the web have led to the emergence of online social networking services. This is evident by the multitude of activity and social interaction that takes place in web sites like Facebook, Myspace, and Twitter to name a few. At the same time the desire to connect and interact evolves far beyond centralized social networking sites and takes the form of ad hoc social networks formed by instant messaging clients, VoIP software, or mobile geosocial networks. Although interactions with people beyond one's contact list is currently not possible (e.g., via query capabilities), the implicit social networking structure is in place.

Given the large adoption of these networks, there has been increased interest to explore the underlying social structure and information in order to improve on information retrieval tasks of social peers. Such tasks are in the core of many application domains. To further motivate our research, we discuss in more detail the case of *social search.* Social search or a social search engine is a type of search method that tries to determine the relevance of search results by considering interactions or contributions of users. The premise is that by collecting and analyzing information from a user's explicit or implicit social network we can improve the accuracy of search results. The most common social search scenario is the following:

1. A user v in a network submits a query to a search engine.

2. The search engine computes an ordered list L of the most relevant results using a global ranking algorithm.

3. The search engine collects information that lies in the neighborhood of v and relates to the results in L.

4. The search engine utilizes this information to re-order the list L to a new list L' that is presented to v.

The utility of social search has been established via experimental user studies. For example, in [106], authors report improved result accuracy for web search when urls for a query are not ranked based on some global ranking criteria, but based on the number of times people in the same social environment endorsed them. Many ideas have been suggested to realize online social search; from entirely human search engines that utilize humans to filter the search results and assist users in clarifying their search requests to social-influenced algorithms that exploit a user's web history log to influence result rankings, so that pages that she visits more often are ranked higher. In any case, the goal is to provide end users with a limited number of relevant results informed by human judgement, as opposed to traditional search engines that often return a large number of results that may not be relevant. These are all examples of tasks that require to visit and probe a large number of peers in the extended network of an individual for information that lies locally in their logs, and then use this information to improve the quality of search experience.

Despite the fact that many algorithms and tools exist for analysis of networks, in general, these mainly focus on analysis of the properties of the network structure and not on the content of the nodes. They also typically not operate on user specific graphs (i.e, users' neighborhoods), but on the whole graph. Instead, for many modern applications, it would be beneficial to design algorithms that operate on a single node. For example, in the case of social search, it would be beneficial to design algorithms that starting from a specific user in the network, crawl its (extended) neighborhood and collect information that lies on their close peers. Such networks may consist of thousands of users and their structure may not be static, thus a complete crawl of all social peers is infeasible. Therefore, efficient methods are required. Based on these observations we focus on improving the performance of information collection from the neighborhood of a user in a social network and make the following contributions:

- We introduce sampling based algorithms that given a user in a social network quickly obtain a near-uniform random sample of nodes in its neighborhood. We employ these algorithms to quickly approximate the number of users in a user's neighborhood that have endorsed an item.

- We introduce and analyze variants of these basic sampling schemes in which we aim to minimize the total number of nodes in the network visited by exploring correlations across samples.

- We evaluate our sampling based algorithms in terms of accuracy and efficiency using real and synthetic data and demonstrate the utility of our approach. We show that our basic sampling schemes can be utilized for a variety of strategies aiming to rank items in a network, assuming that information for each user in the network is available.

Our research aims to offer performance improvements, via sampling, to the process of (almost) uniformly collecting information from user logs by exploring the underlying graph structure of a social network. Note that our work doesn't make any assumption about the semantics of neighboring nodes, thus we do not assume that users that are closer in the network may exhibit some behavioral similarity, have similar interests or so. On the contrary, in designing our algorithms we assume that any node in the extended neighborhood of a user is by definition equally important, independently of how close or far it lies from the initiator node. How one can utilize this information to increase user satisfaction is orthogonal to our work and out of the scope of this research.

The rest of the chapter is organized as follows. Section 5.2 formally defines the problems of interest in this research and introduces notation. Foundational ideas of our sampling methods are presented in section 5.3, and section 5.4 describes the algorithmic details that implement them. We experimentally evaluate our algorithms in section 5.5. In section 5.6 we review related work and conclude in section 5.7.

5.2 Problem Definition

The social network structure can be modeled as a graph G with individuals representing nodes and relationships among them representing edges. We model environments in which social peers participate in a *centralized* social network (where knowledge of the network structure is assumed) or *distributed* (where network structure is unknown or limited). Centralized graphs are typical in social networking sites in which complete knowledge of users's network is maintained (e.g., del.icio.us, flickr, etc.). Distributed graphs, where a user is aware only of its immediate connections, are more common. Consider for example the case in ad hoc social networks formed by typical instant messaging or VoIP protocols (e.g., MSN, Skype). There are also cases that the model of the graph is between the centralized and the fully distributed allowing limited knowledge of a node's neighborhood typically controlled by the node in terms of privacy settings (e.g., LinkedIn, Facebook). Our methods apply to these models as well, with slight modifications. The *rate of change of the structure* of these networks is also an important factor. The most typical case is for such networks to change rapidly as users join and depart from the graph by forming or destroying social connections. Although one can make a case for relatively static social networks (in which the graph structure changes less frequently) in general such graphs are expected to be highly dynamic. We focus on dynamic networks (either centralized or distributed) but also treat the relatively easier case of static networks. In any case, we assume that the *rate of change of the content* in these networks is high. Given such an environment we define the following two problems of interest.

5.2.1 Sampling Nodes in Social Networks

Using G and starting at v we would like to be able to obtain the set of nodes in the neighborhood $D_d(v)$ of v at some specific depth d (i.e., at most d hops away of v) . However, crawling the entire $D_d(v)$ at runtime may be prohibitively slow, especially as the size of the neighborhood increases in number of nodes. Therefore, we have to resort to efficient approximation methods such as sampling. By sampling we avoid visiting all nodes in $D_d(v)$ and thus attain improved performance. We formally define the following problem:

Problem 2. *Let a graph G and a user $v \in G$. Let $D_d(v)$ be a user specified vicinity of v at depth d. Using G and starting at v, obtain a uniform random sample of the nodes in $D_d(v)$.*

Note here that the sampling process operates on a node v and should respect the underlying network structure of v's neighborhood, in a sense that all users in $D_d(v)$ should have the same chance to be selected in the sample. Thus, we assume (by definition of our problem) that any node in $D_d(v)$ should be equally important for the information task, and ignore other semantics, such as the distance of a node from v.

5.2.2 Sampling Information in Social Networks

For each user in G we assume that a log accumulated over time is available. Let the log at node v have the form $(x, count_x^v)$, where x is an item and $count_x^v$ is the number of times x has been endorsed by user v (or a numeric value that represents the endorsement of user v to item x). Endorsement of an item is defined in a generic sense and it may have various instantiations, for example clicking on a url, rating a movie, etc. Endorsements of items by users in the neighborhood of v comprise valuable social information that may be utilized to provide personalized rankings of items to v. In many social information tasks (such as in social search) we are interested in the *relative order* or *ranking* of a set of items X in the social network of v. Using G and starting at v we would like to obtain the total count of the number of times that each item $x \in X$ has been endorsed by consulting the neighborhood of v at some specific depth d (i.e., at most d hops away of v). Formally, if we define y_v as the quantity $count_x^v$, then for an item $x \in X$ we may obtain its exact aggregate value $Y = \sum_{i \in D_d(v)} y_i$ by visiting and querying the log at every node in the specified vicinity of v, $D_d(v)$. However, visiting any node in $D_d(v)$ and computing the exact aggregate value Y for any item $x \in X$ at runtime may be prohibitively slow, especially as the size of the neighborhood increases in number of nodes. Therefore, we have to resort to efficient approximation methods such as sampling. We formally define the following problem:

Problem 3. *Let a graph G and a user $v \in G$. Let $D_d(v)$ be a user specified vicinity of v at depth d. Let X be a set of items. Obtain through sampling nodes of G in $D_d(v)$ an estimate of the ordering of the items X in $D_d(v)$.*

Once the social information has been collected, a number of personalization strategies are possible to re-rank the items in X taking into account semantics of the collected information, such as the item counts or the distance of a sampled user to user v. Designing and evaluating a re-ranking algorithm that increases the user satisfaction is out of the scope of this research.

5.3 Sampling Methodology

In this section we discuss the foundational ideas behind our sampling-based approaches to solve Problem 2. We first describe an idealized approach in which we assume it is possible to efficiently obtain a uniform random sample of $D_d(v)$. The sampling notation we use is shown in Table 5.1 for reference. Let y_1, y_2, ..., y_N be the values of the nodes in $D(v)$. Suppose we could obtain a uniform random sample S of size $n \ll N$ with $S \subset D_d(v)$ and values y_1, y_2, ..., y_n. Let y be the sample sum, i.e., $y = \sum_{i \in S} y_i$. Then it is well known that the quantity $Y' = y \cdot (N/n)$, i.e., the sample sum scaled by the inverse of the sampling fraction, is an approximation for Y. In fact, Y' is a random variable whose mean and standard deviation can be approximated (for large N) by the following well-known sampling theorem [33].

Theorem 8.

$$E[Y'] = Y, \; sd[Y'] = N \cdot \sigma / \sqrt{n}$$

The standard deviation $sd[Y']$ provides an estimate of the error in estimating Y by Y'. Since σ, the standard deviation of values in $D(v)$, is usually not known in advance, it can be itself estimated by computing the standard deviation σ' of the sample; thus $sd[Y']$ is estimated as $N\sigma'/\sqrt{n}$. More fine-grained error estimations such as *confidence intervals* are also possible; see [33] for details. Remind that in Problem 3 we seek for an approximate *ordering* of the items in a set X. This ordering can be obtained directly by the estimated sample

Table 5.1: Sampling Notation

Notation	Explanation
N	Number of nodes in $D(v)$
S	Set of nodes in Sample
n with $n \ll N$	Number of nodes in Sample
y_1, y_2, ..., y_N	Values of the nodes in $D(v)$
y_1, y_2, ..., y_n	Values of the nodes in S
σ	Standard deviation of values in $D(v)$

sums without the need to scale them by the inverse of the sampling fraction (i.e., N/n). Practically, the total number of nodes in $D(v)$ (i.e., N) from which we form the sample does not need to be known. We will only assume a priori knowledge of this number when evaluating the accuracy of our sampling methods, in order to compare estimated with actual aggregate values. Given this framework, the main challenge confronting us is how to obtain a uniform or near-uniform random sample of the nodes in $D_d(v)$. We discuss this issue under assumption of *static* and *dynamic* network topologies.

5.3.1 Assuming Static Networks

We first consider the case where the topology of the social network is static, or changes only slowly over time (although the clickthrough logs, i.e., the "data" stored at each node are rapidly changing). For this case, a straightforward solution exists where each node, in a precomputation phase, performs a complete crawl of its neighborhood $D_d(v)$ and selects a uniform random sample S of n nodes, whose addresses (or access paths) are then stored at the initiating node. At runtime, the value stored at each sample node is retrieved and aggregated. Clearly, such a precomputation phase is computationally intensive. However, this phase needs to be recomputed infrequently; once the social network topology has undergone significant changes.

5.3.2 Assuming Dynamic Networks

We next consider the case where the topology of the network is dynamic, i.e., where the network structure changes frequently in addition to the data changes at each node. In such a case, it makes little sense to precompute samples of $D_d(v)$ as such samples go stale very quickly. Thus, the task of sampling from $D_d(v)$ has to be deferred to runtime. This problem is challenging because we cannot crawl the entire neighborhood $D_d(v)$ at runtime (this will be prohibitively slow). It becomes even more challenging by the fact that we are constrained to simulate random walks by only following edges of the social network. As we discuss in section 5.6, there are methods to generate a uniform random subset of nodes of a large graph via random walks.

However, in our case, we can improve upon generic random walk methods on graphs as we can leverage the fact that we need to only sample from the neighborhood $D_d(v)$ of a node v with a small depth d (i.e., just a few links away from v). Consequently, we are able to develop even more efficient random walk procedures. We first make a simplifying assumption that the graph structure of the neighborhood $D(v)$ resembles a *tree* rooted at v. The solution that we first present will consist of random walks that are initiated from the root of this tree v and follow edges towards the leaves of the tree. Later, we describe how to generalize this basic approach for more general graph structures that are not trees - essentially by constraining our random walks to only follow edges of a spanning tree of $D_d(v)$ rooted at v.

Assuming that $D_d(v)$ is a Tree

Assume that the subgraph of the social network induced by the nodes in $D_d(v)$ is a tree T with N nodes (a_1, …, a_N), where each node is a member of $D_d(v)$. Assume that $v = a_1$ is the root and that all edges are directed downwards, i.e., from root to leaf. The maximum depth of this tree is d. Recall that each node a_i in T contains a value y_i which we wish to aggregate. To allow for better conceptualization, let us first convert the tree T to another tree T', such that the values are only at the leaf nodes, and not at internal nodes. We do this as follows: for each internal node a_i we add a leaf b_i and connect a_i to b_i via a new edge. We then move the value of a_i to b_i (see Figure 5.1). Note that for each internal node in the original tree, the *degree* in the new tree has been increased by one, and that now the number of leaves is N. To motivate our approach, let us first make the (unrealistic) assumption that for each node a_i, we know $size(a_i)$ the number of leaves in the subtree rooted at a_i. Let us also assume that each edge of the tree is weighted as follows: Let the set of children of node a_i be A_i. Consider any child node a_j in A_i. Then, weight(a_i, a_j) is defined as $\frac{size(a_j)}{\sum_{a \in A_i} size(a)}$. It is easy to see that each weight is in [0, 1] and for each node, the sum of the outgoing edge weights adds up to 1. Once T has been transformed to T', we shall perform random walks on T'. A random walk starts from the root and ends at a leaf. At every internal node, it picks an outgoing edge with probability equal to its weight. Once the walk has ended, the leaf node is returned by the random walk. The following lemma determines the probability of returning any specific leaf node.

Lemma 2. *The probability of the random walk returning any specific leaf node b_i is $p(b_i) = 1/N$*

Proof. (*Sketch*) The proof hinges on the way the edge weights of the tree have been defined. Note that each leaf is picked with probability equal to the product of the weights of all edges encountered along the walk. □

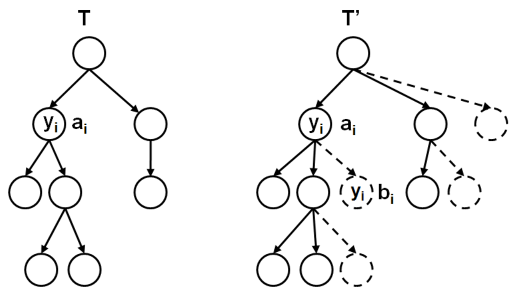


Figure 5.1: Transformation of T to T'.

This random walk procedure can be repeated n times to obtain a uniform random sample (with replacement) of the nodes in $D_d(v)$ of size n. Thus, after we have done n independent random walks, we will have collected n leaves (may contain duplicates), i.e., the set of n leaves is a near-uniform random sample of the entire set of N nodes. Of course, for the above scheme to work, we have to know the sizes of each node and the weights of each edge of the tree. Clearly, computing these at runtime will be prohibitive as it will require a full traversal of the tree. Therefore, without knowing these quantities in advance, we are left with no choice but to select each outgoing edge with equal probability, i.e., $\frac{1}{|A_i|}$. But if we perform the random walk this way, we shall pick leaf nodes in a biased manner, because some leaves are more likely to be destinations of random walks than other leaves. We explore the effect of this bias in the sampling accuracy in the experimental section.

Correcting for the bias: One way to correct for this bias is to let the random walk reach a leaf, but instead of accepting it into the sample, we toss a biased coin and only accept it if the coin turns up as heads. So, we have to determine what should the bias (i.e., the *acceptance probability*) of the coin be. Let the probability of reaching the leaf b_i be $p(b_i)$ (i.e., the product of $1/outdegree$ of all nodes along the path from root to leaf). Let $maxDeg$ be the maximum out-degree of the tree. The following lemma suggests how to set the acceptance probability of a leaf b_i to achieve near-uniform random sample.

Lemma 3. *If the acceptance probability of a leaf b_i (i.e., the bias of the coin) is set to $C/p(b_i)$ where $C \leq 1/maxDeg^d$, then a random walk performs near-uniform sampling.*

Proof. Let C be a constant that controls the probability with which a leaf node is returned by a random walk. We need to find a value of C that ensures the probability of the random walk returning any specific leaf node b_i is the same for all leaves. It is easy to see that an appropriate value for C depends on the structure of the tree and the number of leaves in it, which we cannot assume to be known. Without any prior knowledge of the tree structure we can, at the worst case, assume that it is a full k-ary tree, a tree where each node has either 0 or k children, depending on whether it is a leaf or an internal node. It is known that for a full k-ary tree with height h, the upper bound for the maximum number of leaves is k^h. In our case $k = maxDegree$ and $h = d$. Therefore, the maximum number of leaves in a tree on which our random walks operate would be $maxDegree^d$. Thus, to ensure near-uniform sampling of any leaf node we require that $C \leq 1/maxDegree^d$. Note also that since $C/p(b_i)$ represents a probability, it has to be at most 1. This is guaranteed since $p(b_i) \geq 1/maxDeg^d$. □

Assuming that $maxDeg$ is known in advance is perhaps not that crucial; after all, the maximum degree $maxDeg$ of the tree can be bounded if one has a reasonable idea of the maximum degree of the entire social network. Note that unlike the previous case where each random walk returns a random node, here we are not always guaranteed that a random node will be returned. In fact, often a random walk fails to return a node. The probability of *success* p_s of a random walk returning non-empty is described by the following lemma.

Lemma 4. *The probability of success of the random walk returning any specific leaf node b_i is $p_s = C$.*

Proof. The probability of success p_s that a random walk returns a leaf b_i is equal to the probability of reaching that leaf ($p(b_i)$) multiplied by its acceptance probability ($C/p(b_i)$). Thus, $p_s = p(b_i) \cdot C/p(b_i) \Rightarrow p_s = C$. □

Thus if we iterate this random walk several times and collect all returned nodes, we will be able to get a near-uniform random sample of any desired size. The following lemma quantifies the expected number of random walks needed to generate a near-uniform sample of size n.

Lemma 5. *The expected number of random walks required to collect a uniform random sample of size n is n/p_s.*

Proof. The proof hinges on the probability of success p_s. Since the probability that a random walk returns a node is C, we need n/p_s random walks to collect n nodes. □

Generalizing When $D_d(v)$ is not a Tree

For purposes of exposition we have been assuming that the induced subgraph of the social network over $D_d(v)$ is a tree; most induced subgraphs are not trees, but graphs with higher connectivity. However, we can adopt our solution of sampling from trees to this specific scenario by ensuring that the union of all random

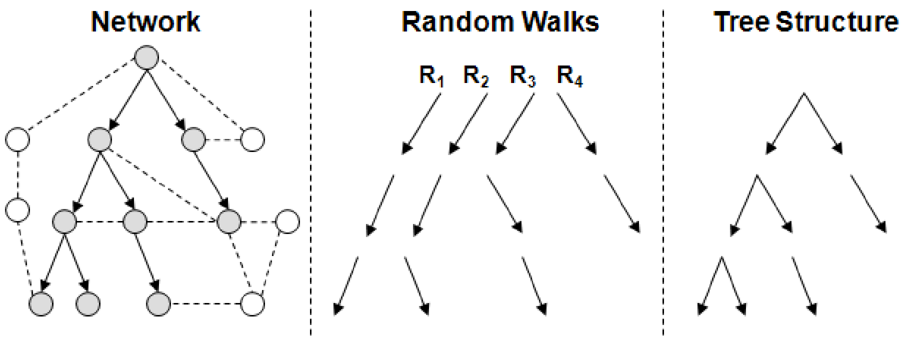
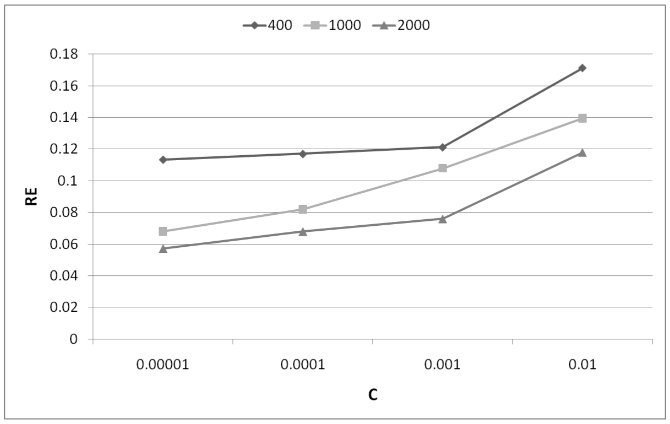


Figure 5.2: Random Walks that Obey Tree Structure

walks made in collecting a sample always resembles a tree. To do so, we have to keep a history of all random walks processed in response to this query, and make sure that at any point in time, their union has no cycles (see Figure 5.2). More precisely, for each fresh random walk we have to ensure that it can be partitioned into two parts; the first part is a prefix of a previous random walk, while the second part is a random walk that does not visit a single node that has been visited by earlier random walks. To comply with the above constraints, when a random walk is progressing, state information can be maintained as to whether it is still a prefix of a previous random walk, or whether it has moved on into the unvisited region of $D_d(v)$. Thus, if the last node a_j along the random walk is a previously visited node, then the set of neighboring nodes that are candidates for the next random step will be the neighbors of a_j *minus* the nodes that have been visited earlier. It is not hard to see that such an effort will ensure that the union of all random walks is a tree which is a subset of the graph induced by $D_d(v)$.

5.3.3 Tuning C

Despite its neatness, our sampling approach suffers one inherent drawback. Setting such a conservative value of C (i.e., $C \leq 1/maxDeg^d$) results in an extremely inefficient process for collecting samples. This is because a very small C, while ensuring near-uniform random samples, almost always rejects a leaf node from being included in the sample, and consequently, numerous random walks may have to be undertaken before a leaf node is eventually accepted into the sample. Let us refer to the maximum value of C that ensures a near-uniform random sample as C_{opt} (i.e., $C_{opt} = \frac{1}{maxDeg^d}$). Note that setting C to be larger than C_{opt} would result in a larger acceptance probability per node (i.e., $\frac{C}{p(b_i)}$), which would eventually result in fewer random walks needed to generate a sample of desired size n. However, a larger C is likely to introduce non-uniformity, or bias into the sample. This is because for all leaves b_i since $C > C_{opt}$ it will be $\frac{C}{p(b_i)} > \frac{C_{opt}}{p(b_i)}$.



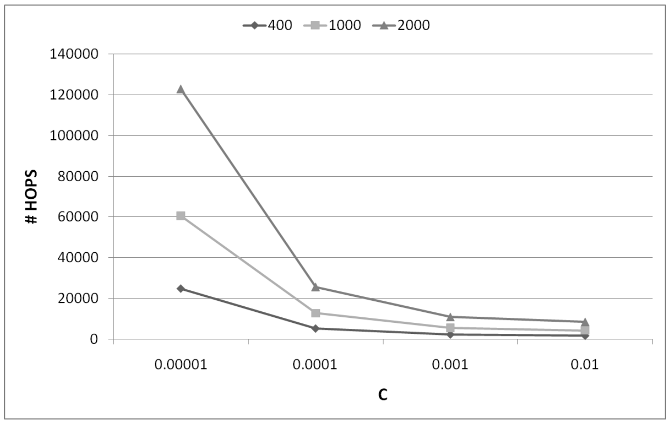


Figure 5.3: Effect of C in Sampling Accuracy (up) and Sampling Cost (down)

This means that once leaves are reached they are more likely to be accepted into the sample and that are therefore going to be unduly over-represented in the sample. Thus, the parameter C can serve to illustrate an interesting tradeoff between ease of collecting sample nodes and the bias of the sample obtained. We further investigate the effect of adjusting the parameter C and demonstrate this tradeoff between accuracy and efficacy by running experiments on a synthetic network of 75k nodes and 450k edges, for network depth d=4 and for variable values of C and sample size $n = \{400, 1000, 2000\}$. We report on the sampling accuracy in terms of relative error RE and the sampling cost in terms of the number of hops in the random walks needed to form the sample. The values of C were arbitrarily selected to clearly exhibit the tradeoff between accuracy and efficiency. Figure 5.3 (left) demonstrates the effect of C in the sampling accuracy, where as C gets larger the relative error increases. Meanwhile, Figure 5.3 (right) demonstrates the effect of C in the sampling cost, where as C gets smaller the number of hops in the random walks needed to form the sample increases and eventually renders sampling inefficient. Depending on the application area, one would need to adjust this parameter to balance time and accuracy performance according to needs. A method to select a proper C is discussed next.

Selecting a proper C

An adequate heuristic would be to set the parameter C to be equal to $1/N$, where N is the number of leaves in the transformed tree from which we want to sample. Recall that N represents the number of nodes in the original tree (before the transformation). This would assume that all leaves have the same probability to be selected, and as $1/N$ is expected to be much smaller than $1/maxDeg^d$, fewer random walks will be needed to generate a sample of desired size n. However, we cannot assume that the number of the nodes in the tree N is known a priori. Finding N would require to perform an exhaustive search on the graph, using depth-first-search (DFS) or breadth-first-search (BFS), but this method is impractical in our setting. The only feasible approach would be to try to estimate N. Estimating the size of a tree is a challenging problem that arises in many domains. It commonly appears as the problem of estimating the size of backtracking trees or branch-and-bound procedures for solving mixed integer programming problems. Knuth [84] was the first to discuss the problem and proposed a method based on random probing that would estimate the size of a search tree and eventually the running time of a backtracking program. In [34] authors predict the size of a branch-and-bound search tree by building a simple linear model of the branching rate using the maximum depth, the widest level and the first level at which the tree is no longer complete. Later, Kilby et al. [78] proposed two methods to solve the problem: one based on a weighted sample of the branches visited by chronological backtracking and another based on a recursive method that assumes that the unexplored part of the search tree will be similar to the part that has already been explored. Finally, statistical techniques have been applied to predict the size of search trees; in [74] authors describe how Bayesian methods can be used to build models that predict the runtimes of constraint satisfaction algorithms; these predictions can then be used to derive good restart strategies. Unfortunately, these methods cannot be applied in our problem. More relevant to our setting are the methods described in [73], where the authors try to estimate the size of an undirected graph or a social network. Their methods work by performing random node sampling and then counting the number of collisions (pair of identical nodes) or the number of non-unique elements (elements that have been sampled at least once before) in the sample. Despite their elegance, these methods cannot be applied in our case, as they assume that nodes are sampled from the graph's stationary probability, which we cannot assume to be known in our setting. The size of a tree can also be estimated using a method known as *mark and recapture*. This method is used in ecology to estimate the size of a natural population and involves marking a number of individuals in a population, returning them to that population, and subsequently recapturing some of them as a basis for estimating the size of the population at the time of marking and release. The main idea hinges on the so called "birthday paradox" effect. After sampling r nodes uniformly at random we expect to encounter $L \approx r^2/2N$ collisions (nodes already picked). Then, an estimate $\hat{N}$ of N can be computed by $\hat{N} = r^2/2L$. However, in order to use this method nodes need to be sampled uniformly. Going back to our problem, the following strategy can be followed to find a proper C:

- set C to a very small value, that can guarantee uniform sampling of nodes, such as $C = 1/maxDeg^d$.
- obtain r samples $\{x_1, ..., x_r\}$ using our sampling methodology, and count the number of collisions L. A collision is a pair of identical samples. Formally, let $P_{i,j}$ to be 1 if $x_i = x_j$ and 0 otherwise. Then, $L = \sum_{i<j} P_{i,j}$.
- use the *mark and recapture* method to estimate the tree size N by $\hat{N} = r^2/2L$.
- re-set the C parameter to $C = 1/\hat{N}$ and use this value for feeding the sampling method in subsequent runs.

We investigate the accuracy of this method by running experiments on a synthetic network of 75k nodes and 450k edges, for variable network depth d={4, 5} and for variable sample size n (expressed as percent of nodes in the original network). We report on the normalized *mean absolute relative error*, i.e., $|N - \hat{N}|/N$, where N is the true size of the network and $\hat{N}$ is our estimate of it. Plots were produced by averaging over 10000 independent experiments (we experimented with 100 random users and for each user we estimate the network size at depth d, 100 independent times). Figure 5.4 demonstrates that as the sample size increases the error decreases, while the error converges faster in the case of the larger network. The advantage of this method is that taking only a small sample can guarantee an accurate estimate of N. For example, by using a sample of 10% of the network the estimation ensures a normalized mean absolute error of less than 20%. Note, that in setting C we only require a rough estimate of the network size, and as such this method is adequate. Throughout the experimental evaluation we set the C parameter to be equal to $1/N$ and do not further investigate the performance of the tree size estimator. Estimating the size of a tree or a network is an orthogonal problem; it is expected that other estimators might be advantageous to our methods with slight modifications, such as the ones presented in [73].

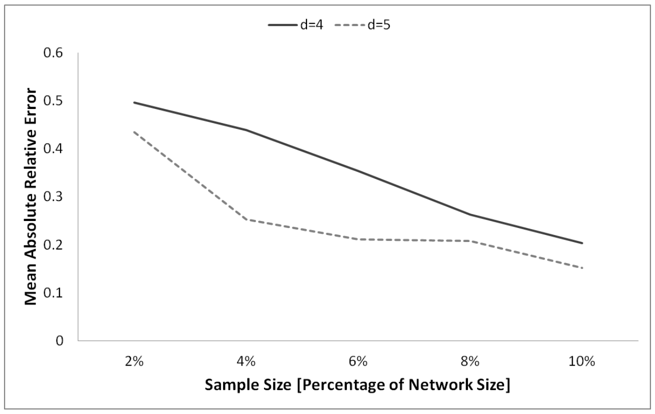


Figure 5.4: Network Size Estimation Error

5.4 Algorithms

In this section we present algorithmic details of our proposed methods. First, we describe *SampleDyn*, an algorithm that is able to compute a near-uniform sample of users in dynamic social networks. Then, we introduce two algorithms, *EvalSingle* and *EvalBatch*, that utilize *SampleDyn* in order to estimate counts for a set of items in a user's vicinity.

5.4.1 Sampling Dynamic Social Networks

Let $D_d(v)$ be the vicinity of a user v at depth d. We introduce the algorithm *SampleDyn* that takes as input the user v, the size of the sample n, the network depth d, and a constant value for parameter C and obtains a near-uniform random sample of users by performing random walks on the nodes of $D_d(v)$. The pseudocode is given in Algorithm 6. Let $children(u)$ denote the nodes that are directly connected to the current node u and are either nodes that have not been visited by any of the previous random walks (unseen nodes) or nodes that extend on the prefix random walk that has been followed so far. Then, $children(u) \cup u$ represents the set of candidate nodes for the next step of the walk (line 2). Each of the candidate nodes is selected with the same probability. Thus, a random walk starts at user v and ends either when a self-link is followed, a link that connects a node with itself (line 4) or when a node in depth d has been reached (line 1). Note that, as the random walk progresses, state information is maintained regarding previous walks and visited nodes that ensures the random walk obeys structural properties of a tree. We require that $T \cup v$ *has no cycle* to represent this information (line 3). Once a node has been reached it is selected to the sample with probability equal to the acceptance probability C/p (line 5). For example, Figure 5.5 shows a network of depth $d = 4$

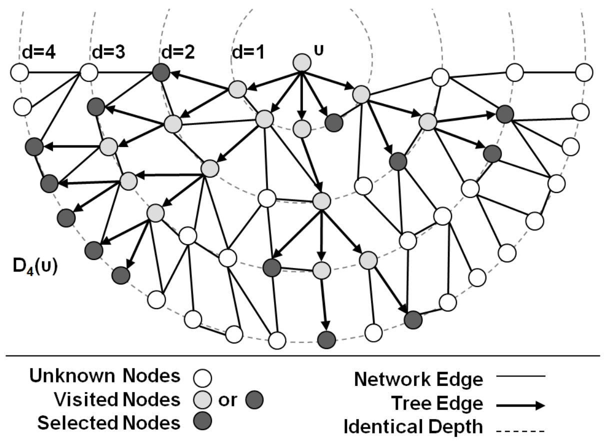


Figure 5.5: Example Random Walks Guided by *SampleDyn*

around user v along with a set of nodes that have been visited by previous walks (light and dark shadowed nodes) and nodes that have already been selected for the sample (dark shadowed nodes). Note that the random walks are forced to obey structural properties of a tree. The tree is defined by the union of the earlier successfull random walks (indicated by the directed edges). The structure of the tree remains valid throughout the current query evaluation (i.e., for as long as n sample nodes have been selected). Different tree structures are possible every time a query is evaluated at v depending on the sequence of the successful random walks and the underlying network structure of the user's vicinity at depth d.

5.4.2 Estimating Item Counts

Recall that our goal, as defined in Problem 3, is to compute the counts of items in a set X, which are then used to assume an ordering. We present two approaches to estimate the ordering of a set of items in $D_d(v)$ using sampling.

Using Separate Samples

A first approach is to draw a separate independent sample from $D(v)$ and estimate the aggregate counts for each item. Formally, we introduce an algorithm that for each $x \in X$, obtains an approximate value of $\sum_{i \in D_d(v)} count_x^i$ through sampling nodes of $D_d(v)$. The algorithm takes as input v, d, C, n and X and returns an array of the approximate counts. We refer to this algorithm as *EvalSingle* because it evaluates a single item at each visit to a sampled node. The pseudocode is given in Algorithm 7. While such an approach is statistically sound, the drawback is *efficiency* - this approach is unlikely to allow us to complete

ALGORITHM 6: Sampling in Dynamic Social Networks

Input: A user u, the size of the sample n, the network depth d and a constant value C
Output: A near-uniform random sample of users in the vicinity of u

```
Function SampleDyn(u, n, d, C)
    $T$ = NULL;
    $samples = 0$;
    $Sample$ : array of size $n$;
    while $samples <= n$ do
        if $(v = RandomWalk(u, d, C, T))\ != \ 0$ then
            $Sample[samples++] = v$;
        end
    end
    return $Sample$;

Procedure RandomWalk(u, d, C, T)
    $depth = 0$;
    $p_s = 1$;
1   while $depth < d$ do
2       pick $v \in children(u) \cup u$ with $p_v = \frac{1}{degree(u)+1}$;
3       if $T \cup v$ has no cycle then
            add $v$ to $T$;
            $p_s = p_s * p_v$;
4           if $v = u$ then
5               accept with probability $\frac{C}{p_s}$;
                if accepted then
                    return $v$;
                else
                    return 0;
                end
            else
                $u = v$;
                $depth++$;
            end
        end
    end
    return 0;
```

the re-ranking process fast enough to satisfy end users.

Using the Same Sample

An alternate approach is to draw a sample S only once, and reuse the same sample to estimate the aggregate counts for each item $x \in X$. We refer to this algorithm as *EvalBatch* because it evaluates a batch of items at each visit to a sampled node. Algorithm 8 presents the pseudocode for this case. Clearly this approach will be much faster, since we need to compute only one sample. Note however, that though practical, this process is flawed since the same sample is reused for a set of items, which are likely to exhibit strong correlations, i.e., a bad sample can affect the counts of *all* $|X|$ items. This phenomenon is well studied in statistics,

ALGORITHM 7: Counts Estimation - Separate Samples

Input: A user v, the size of the sample n, the network depth d, a constant value C and a set of items X
Output: An array of the approximate count of items of X in the vicinity of u

```
Function EvalSingle($v$, $n$, $d$, $C$, $X$)
    $S$ : array of size $n$;
    $Count$ : array of size $|X|$;
    forall the $x \in X$ do
        $S = SampleDyn(v, n, d, C)$;
        forall the $i \in S$ do
            $Count[x] = Count[x] + count_x^i$;
        end
    end
    return $Count$;
```

ALGORITHM 8: Counts Estimation - Same Sample

Input: A user v, the size of the sample n, the network depth d, a constant value C and a set of items X
Output: An array of the approximate count of items of X in the vicinity of u

```
Function EvalBatch($v$, $n$, $d$, $C$, $X$)
    $S$ : array of size $n$;
    $Count$ : array of size $|X|$;
    $S = SampleDyn(v, n, d, C)$;
    forall the $i \in S$ do
        forall the $x \in X$ do
            $Count[x] = Count[x] + count_x^i$;
        end
    end
    return $Count$;
```

and is known as *simultaneous statistical inference* (see [105]). The classical solution proposed in [105] is to make Bonferroni corrections to ensure that the estimated counts of the items fall within their confidence intervals. A similar problem also arises in sampling-based approximate query answering techniques. For example, popular approaches in approximate query answering is to pre-compute a sample and use the same sample to answer a stream of aggregation queries (see [52]). Likewise, due to practical considerations, our proposed approach is to also reuse the same drawn sample for estimating the counts of all returned items. We experimentally evaluate the impact of such correlations and results indicate that in practice, the errors in the approximations are not unduly severe.

5.4.3 Cost Analysis

Our sampling algorithms provide an alternative to performing an exhaustive search or *crawling* on the network of a user using a depth-first-search (DFS) or breadth-first-search (BFS). Both DFS and BFS assume that there is a designated initiator node from which the search starts and define a DFS or BFS tree. At the

Table 5.2: Algorithm Complexity

Algorithm	Communication	Time
Distributed DFS	$O(E)$	$O(E)$
Distributed BFS	$O(E)$	$O(E)$
Random Walks	$O(n/p_s \cdot d)$	$O(n/p_s \cdot d)$

end, nodes at distance d from the initiator appear at level d of the tree. In this paragraph we present we simple cost model that helps to analyze and compare the complexity of our basic sampling algorithms to that of crawling.

Cost model: Let $D_d(v) = (N, E)$ be the neighborhood of a user v at depth d, where N is the set of nodes and E the set of links in the network. Nodes are autonomous in that they perform their computation and communicate with each other only by sending messages. Each node is unique and has local information, such as the identity of each of its neighbors. We assume that each node handles messages from and to neighbors and performs local computations in *zero time*, meaning that communication delays outweigh local computations on the nodes. The same assumption is made by Makki and Havas [102]. We evaluate the complexity of our algorithms using standard complexity measures. The *communication complexity* is the total number of messages sent, while the *time complexity* is given by the maximum time elapsed from the beginning to the termination of the algorithm. We assume that delivering a message over a link corresponds to traversing an edge to visit a node, a *hop*, and that delivering a message over a link (i.e., performing a hop) requires at most one unit of time. Therefore, for our algorithms, we use as a surrogate for both communication and time costs the number of hops performed during the execution of the algorithm.

Table 5.2 presents the communication and time complexities of the algorithms. Note that both DFS and BFS algorithms require at least one message to be sent on each edge, yielding a communication and time complexity of at least $O(E)$ [32, 14]. On the other hand, our sampling algorithm has communication and time complexity of $O(n/p_s \cdot d)$, where d is at most the length of each random walk (i.e., the number of hops per random walk) and n/p_s is the expected number of random walks required to collect a uniform sample of size n as shown in lemma 5. Given that typically $n << N$ and assuming an appropriate value for parameter C as discussed before, our sampling algorithm is expected to outperform both DFS and BFS algorithms for both the communication and time complexities. We give examples of performance on typical networks in the experimental section.

5.5 Experimental Evaluation

Having presented our sampling methods and algorithms we now turn to evaluation. For the needs of our experiments we make a case of a social search application. Let G be a graph depicting connections between users in a social network, where each node in the graph represents a user. For each user we assume availability of a clickthrough log accumulated over time via browsing. The log, in its most simple form, at node v, has the form $(q, url_q, count^v_{url_q})$ where q is a query, url_q is the url clicked as a result of q and $count^v_{url_q}$ is the number of times url_q has been clicked by v. A few years ago, it would be difficult to assume that such a log exists due to privacy issues. However, recently, a number of Web2.0 services gather such kind of information. The most notable example might be Google's web history service[1]. But also, Bing's and Facebook's attempts to incorporate in search results a feature that shows you the opinion of your friends as it relates to that search, through the Facebook Instant Personalization feature[2]. Therefore, is reasonable to assume existence of such information. Now, consider the scenario where a query q is submitted to a popular search engine by a user v and a set of urls r_{q_v} is returned. A social search algorithm would try to personalize this result. Intuitively, an algorithm might collect information from v's social network and use this information to re-rank the results according to a re-ranking strategy. Using G and starting at v we can obtain the total count of the number of times that each url $url_q \in r_{q_v}$ has been clicked by consulting the neighborhood of v at some specific depth (number of hops) d. Then a re-ranking r'_{q_v} of r_{q_v} is possible that incorporates the behavior of the users with which v has some social relationship.

5.5.1 Description of Datasets

For the needs of our experimental evaluation we consider real and synthetic *network topologies* along with real and synthetic *user search history logs* (i.e., clickthrough data). To come up with social search logs suitable for our experiments we had to combine these sources. Below we provide details on data characteristics and the data collection process.

Network Topologies: In our experiments we consider one *real* and two *synthetic* network topologies. The real network topology, *epinions-net*, is an instance of the Epinions'[3] real graph. The first synthetic topology, *uniform-net*, simulates a uniform random network topology, while the second synthetic topology, *prefatt-net*, simulates a preferential attachment network topology. When generating the synthetic networks, we set the

[1]Google Web History. http://www.google.com/psearch
[2]Facebook Instant Personalization.
http://www.facebook.com/instantpersonalization
[3]Epinions.com is a general consumer review site.

Table 5.3: Network Topologies

Name	Type	Model	Nodes	Edges
epinions-net	Real	Epinions	75888	450740
uniform-net	Synthetic	Uniform	75888	455292
prefatt-net	Synthetic	PrefAttach	75888	455292

Table 5.4: User Search History Logs

Name	Type	Users	Queries	Urls
real_log	Real	75888	4026350	2789866
synth_log	Synthetic	75888	1000000	3000000

parameters of the graph generators so that the formed networks have similar number of vertices and edges to the real network of Epinions (The JUNG[4] tool has been used to generate the networks.). Note however that since they are not based on the same model they depict different topology characteristics, such as average path length, clustering coefficient and degree distribution. Our objective is to study the application of our sampling based algorithms under diverse assumptions of network connectivity and stress any interesting differentiation on performance due to network topology. Table 5.3 summarizes the characteristics of the three networks we consider.

User Search History Logs: We experiment with *real* and *synthetic* user search history logs. For the needs of our experiments we create *real_log* by randomly selecting 75888 users from the AOL dataset along with their search history logs (about 4M queries, 3M urls) [51]. The synthetic log, *synth_log*, consists of the same users as the *real_log* but we populate user's history logs with high numbers of queries and url counts. A summary of these logs is provided in Table 5.4.

Formation of Final Data Sets: Thus far, we have come up with proper network topologies and user search history logs. In order to generate suitable final data sets for our experiments we randomly map each of the 75888 AOL users in a search history log to the 75888 nodes of a network topology. Note that since the focus of our work is on performance we do not require that "similar" users are placed in adjacent network nodes. Thus, we do not care for any semantics of the data destroyed, such as the fact that friends may have similar interests and so. The objective of our work is to efficiently collect information in social networks. In doing so, we consider all users in the network to be equal, independently on whether they lie a few or many hops away from the initiator node. In a real setting one may make use of such semantics to design a better re-ranking algorithm, but this is orthogonal to the objective of this research.

[4]JUNG. http://jung.sourceforge.net

5.5.2 Evaluation Metrics

We assess the performance of our algorithms according to *accuracy* and *efficiency* measures. *Accuracy* concerns how well our sampling framework estimates the exact count of a url or the ordering of a set of urls in the vicinity of a user. To assess the accuracy in the first case we use the *Relative Error* (RE) metric, which is usually employed to express accuracy of an estimate. Formally, the relative error between an exact value y and an estimated value $\hat{y}$ is given by:

$$RE = |\frac{y - \hat{y}}{y}|$$

To assess the accuracy in the second case we employ two metrics that are usually considered for comparison of ranked lists, the *Normalized Spearman's Footrule Distance* and the *Precision at k*. *Spearman's Footrule Distance* measures the distance between two ranked lists. Formally, given two full lists r' and r that rank items of the set r, their Spearman Footrule Distance is given by:

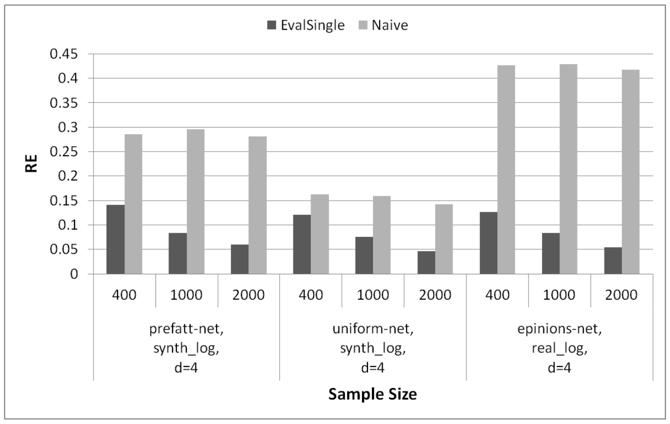
$$F(r, r') = \sum_{e \in r} |r'(e) - r(e)|$$

After dividing this number by the maximum value $(\frac{1}{2})|r|^2$, one can obtain a normalized value of the footrule distance, which is always between 0 and 1. *Precision at k* $(P@K)$ measures the precision at a fixed number of retrieved items (i.e., top k) of the ordered list r' and the ordered list r. Assume $TopK$ and $TopK'$ are the retrieved items of r and r' respectively, then the *precision at k* is defined as:

$$P@K = \frac{|TopK' \cap TopK|}{k}$$

Efficiency concerns the cost of our sampling framework. To assess the efficiency of our sampling algorithms we use as a surrogate for cost the number of *hops* in the random walks performed to obtain n samples from the network. Formally, the cost of the sampling is

$$Cost = \#HOPS$$



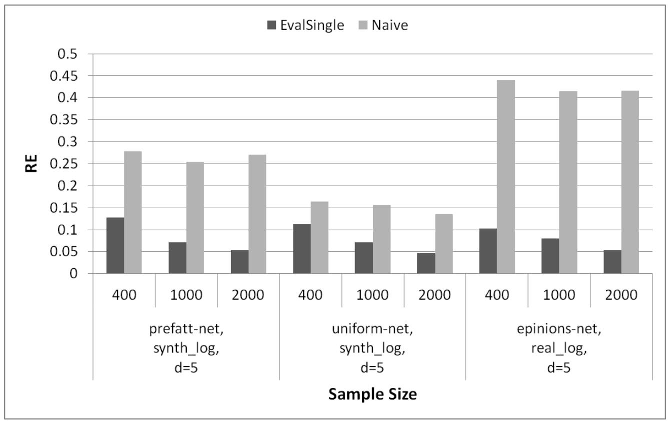
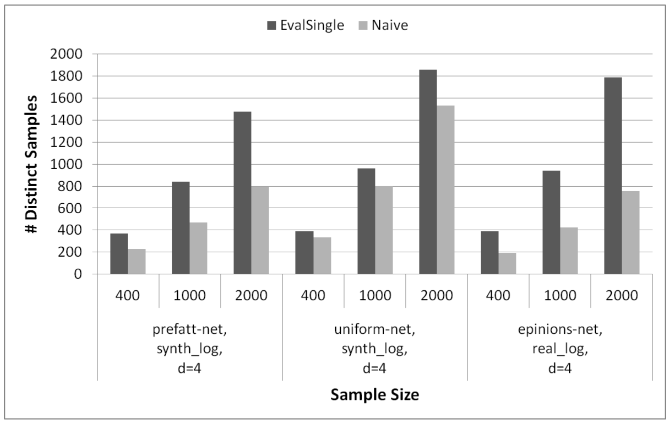


Figure 5.6: Sampling Accuracy: EvalSingle vs. Naive, RE

5.5.3 Experimental Results

Sampling Accuracy

In section 5.3, we have seen that performing random walks by selecting each outgoing edge with equal probability shall pick leaf nodes in a biased manner. This is because some leaves, e.g. leaves that are close to the root, are more likely to be destinations of random walks than other leaves. In our first set of experiments, we explore the effect of this bias in the sampling accuracy and compare the performance of the aforementioned naive sampling method, say *Naive*, to our sampling method, *EvalSingle*. To determine the accuracy performance of the sampling methods, we design experiments that aim to estimate the count of a predetermined url in the network of a user. More specifically, we first determine a target url by submitting a query q to a search engine, such as Google, and obtaining the *top* url. Then, we apply *EvalSingle* or *Naive* to quickly compute an estimate of its count in $D_d(v)$. The process is repeated many times for different queries



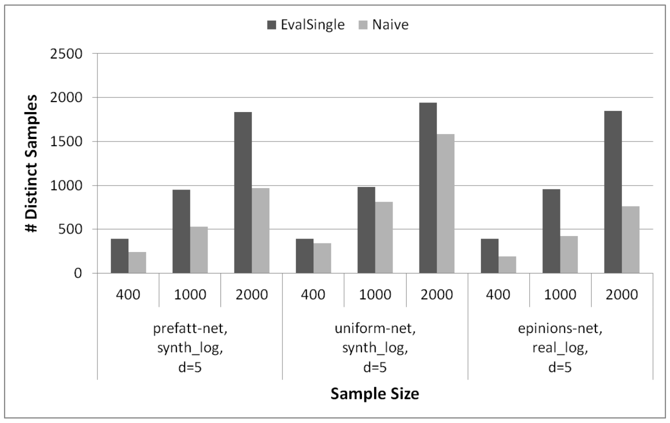
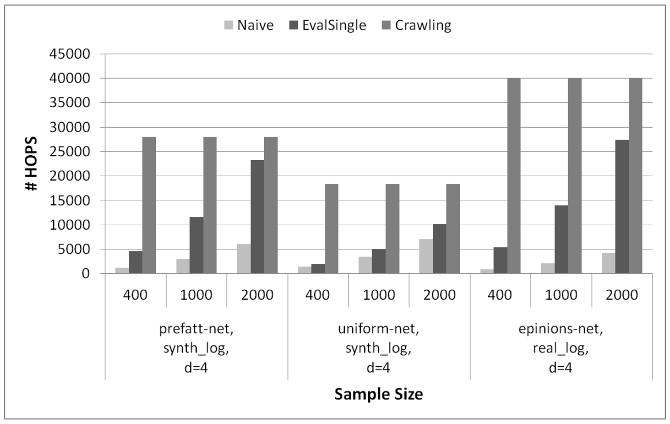


Figure 5.7: Sampling Accuracy: EvalSingle vs. Naive, # Distinct Samples

(typically 100 random queries) and users (typically 100 random users) and at each iteration the relative error of the *estimated url count* to the *exact url count* is computed. We report the average value over all instances. Note that the *exact url count* can be easily obtained by exhaustively crawling the neighborhood of a user, using a breadth-first-search or depth-first-search algorithm, and aggregating the occurences of the targeted url. We experiment for variable network topology and sample size n. Figure 5.6 presents the results for network depths $d = 4$ and $d = 5$ respectively. Let us first focus on the behavior of *EvalSingle* alone and then turn to the comparison of the sampling methods. In all topologies *EvalSingle* is able to estimate the targeted values with a very small relative error (always smaller than 15%). Furthermore, for a fixed network depth d the sampling accuracy of *EvalSingle* increases with the sample size n (i.e., the average relative error decreases for larger sample sizes) and for a fixed sample size n the sampling accuracy decreases as d increases. The observed behavior is in accordance with theory. The total population N (from which we sample) increases with the network depth d ($d \sim N$) and from the Theorem 8 is true that the sampling standard error is proportional to the total population N ($sd \sim N$) and inversely proportional to the number of samples ($sd \sim \frac{1}{n}$). In the case of the synthetic network topologies, for a fixed depth d and a fixed sample size S the sampling accuracy in the *uniform-net* is better than the one in the *prefatt-net.* This can be explained by the fact that the preferential model has a systematically shorter average path length than the random graph model [5]. As a result, for a fixed depth d the average total population N of the *prefatt-net* is larger than the one of the *uniform-net.* Since N is larger in *prefatt-net* than in *uniform-net* for a fixed d the standard error will also be larger according to Theorem 8 ($sd \sim N$). In the case of the real network topology (*epinions-net* with *real_log*) the sampling accuracy results are not directly compared to the results in the synthetic data. This is due to the fact that the exact url counts in the real data are much smaller and leverage the sampling performance. As a result, even if trends are identical to the ones observed in the synthetic data, slightly larger absolute errors are demonstrated. Now, going back to the comparison of the two sampling methods we see in Figure 5.6 that in all topologies and for all network depths and sample sizes, *EvalSingle* considerably outperforms *Naive.* This suggests huge savings in accuracy. The reason for which *Naive* performs so poor is that in the computations it utilizes more and more of the same sampled users, turning the estimator to be biased towards these sampled users. Remind that sampling is performed with replacement, thus, the same node can be selected in more than one random walks. On the other hand, *EvalSingle* succeeds in selecting samples in a near-uniform way, avoiding to a large extent selection of previously selected nodes. For example, if a sample set of size n is requested, then the number of distinct samples in the sample set formed by *Naive* should typically be much lower than the number of distinct samples in the sample set formed by *EvalSingle.* This effect is demonstrated in Figure 5.7, where the number of distinct samples in the sample sets formed by *Naive* and *EvalSingle* is showed, for $d = 4$ and $d = 5$ respectively. Clearly, *EvalSingle* succeeds on considerably alleviating the bias of *Naive* by selecting more distinct samples into the sample sets. This is true for all network topologies tested and for variable sample sizes.

Sampling Cost

EvalSingle performs considerably better in terms of accuracy than a naive sampling method, but as many of the performed random walks end up rejecting a selected leaf node, it can be expensive. In this experiment we evaluate the cost of our sampling method against the naive sampling method and against the cost of crawling the entire neighborhood of a user. As discussed in section 5.3 we use as a surrogate for cost the number of *hops* that the algorithm needs to perform before it ends. Crawling of the entire neighborhood of a user can



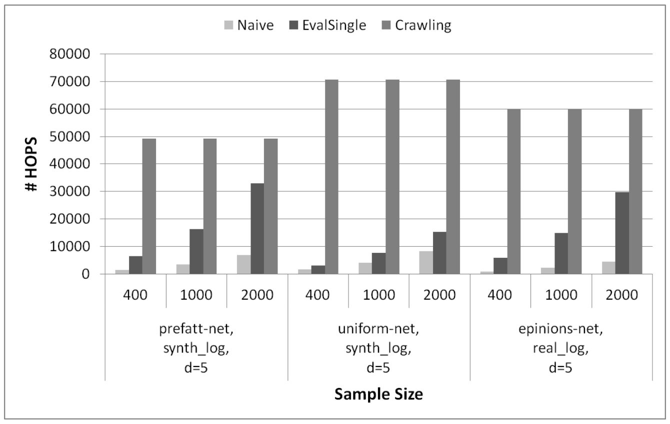
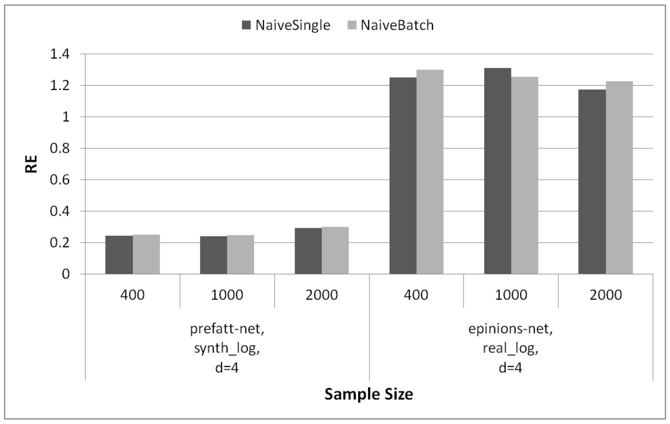


Figure 5.8: Sampling Cost: Naive vs. EvalSingle vs. Crawling

be performed with either of the depth-first-search (DFS) or breadth-first-search (BFS) algorithms. As DFS and BFS have the same performance - linear to the size of the graph - we report only the cost for DFS, which is equal to the number of edges in the spanning tree that has the user as root and extends to a limited depth d. We experiment for variable network topology, network depth d and sample size n. Results are shown in Figure 5.8, where a balance of our sampling method between the naive sampling and the exhaustive crawling methods is illustrated. In all topologies and for all network depths and sample sizes, the performance of the naive sampling is better. This is expected, as the naive sampling sacrifices accuracy for efficiency. On the other hand, our sampling method retains high accuracy while at the same time performs considerably better than the exhaustive method in all settings and, as such, suggests huge savings in efficiency. The savings of the sampling method are greater for larger network sizes, where the number of nodes is also larger; savings are greater, as well, when smaller sample sizes are needed, since fewer random walks are required to form the sample. This is true for all network topologies. For the rest of the experimental evaluation we set the



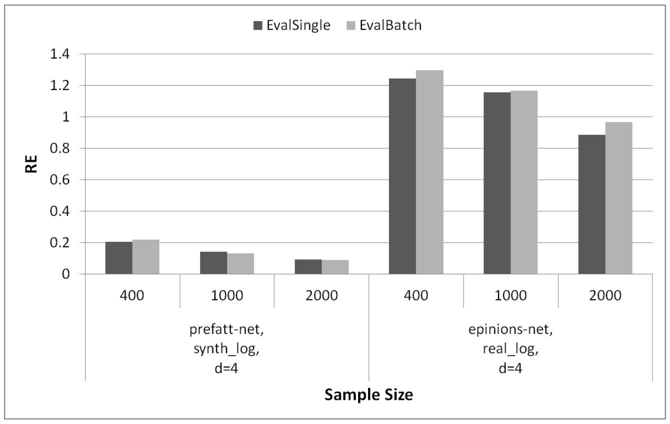


Figure 5.9: Batch Sampling Effect: Naive Sampling (up), Our Sampling (below)

parameter $d = 4$. This is a reasonable choice for our research. For $d = 1$ and $d = 2$ the network populations are small and sampling is not needed. For $d = 4$ we are able to reach almost all nodes in the networks we examine, thus, there is no need to consider $d = 5$ or larger. Between $d = 3$ and $d = 4$ we choose $d = 4$ that will increase the network population and therefore make the approximation problem harder.

Batch Sampling Effect

In section 5.4 we discussed how sampling can be used to estimate the popularity (ordering) of a set of items X in the network of a user and introduced the idea of using the same sample for estimating the count for each item (instead of using a fresh sample for each item). This method is much faster as it needs to compute only one sample and computes estimates for all urls at once (batch processing). However, it is also flawed since the same sample is reused for a set of urls, which are likely to exhibit strong correlations. In this set of experiments we evaluate the impact of such correlations for both the naive sampling and our sampling method. In particular, we compare the accuracy of *NaiveSingle* to *NaiveBatch* and the accuracy of *EvalSingle* to *EvalBatch*. These algorithms represent the single and the batch way of estimating the ordering of a set of items, using the naive sampling and our sampling method, respectively. Accuracy results reported are average relative errors over a number of runs for random users and queries. At each run a set of top-k urls is determined, where k=10, by submitting a query q to Google and retrieving the top-k results. Then, *EvalSingle*, *EvalBatch*, *NaiveSingle* and *NaiveBatch* compute the estimates of the url counts. The estimates are then compared to the exact counts to find the relative error of each method. Figure 5.9 presents the results of the experiment for the naive sampling and our sampling method respectively. For each method, we experiment for a synthetic network (*prefatt-net* with *synth_log*) and a real network (*prefatt-net* with *synth_log*) for network depth $d = 4$ and variable sample size n. Results for both synthetic and real network topologies show that *NaiveBatch* has similar behavior to *NaiveSingle* and *EvalBatch* has similar behavior to *EvalSingle*, which indicates that using the same sample for evaluating url counts has a low effect in the sampling accuracy. This small effect is more obvious in the case of the real network topology (*epinions-net* with *real_log*). This can be explained by the fact that the real data might exhibit some sort of correlation in the urls (while, in the synthetic topologies, we tried to avoid any data correlation by assigning queries and urls to users randomly). As aforementioned, the sampling accuracy results in the real data are not directly comparable to the results in the synthetic data. This is due to the fact that the url counts in the real data are much smaller and leverage the sampling performance. As a result, even if trends are similar to the ones observed in the synthetic data, larger absolute errors are demonstrated. Note, as well, that in the case of synthetic networks we only present the results for *prefatt-net* and omit the details for *uniform-net*. This is because similar trends were demonstrated over the two synthetic network topologies. The preferential attachment model is favored since it widely exists in the social information networks of interest and is also more challenging for our sampling methods. From now on, whenever similar trends are demonstrated between the two synthetic network topologies, we only present the results for *prefatt-net* due to space constraints. Since the errors in the computation of estimates that are due to the use of the same sample are not unduly severe, our proposed approach is to use batch sampling for estimating the counts of many urls at once, which is more efficient. Figure 5.9 illustrates as well how *EvalBatch* behaves better than *NaiveBatch*, where as the number of samples increases, *EvalBatch* becomes more accurate; this is no true for *NaiveBatch*. We further investigate the accuracy of *EvalBatch* and *NaiveBatch* in estimating the ordering of items in the next paragraph.

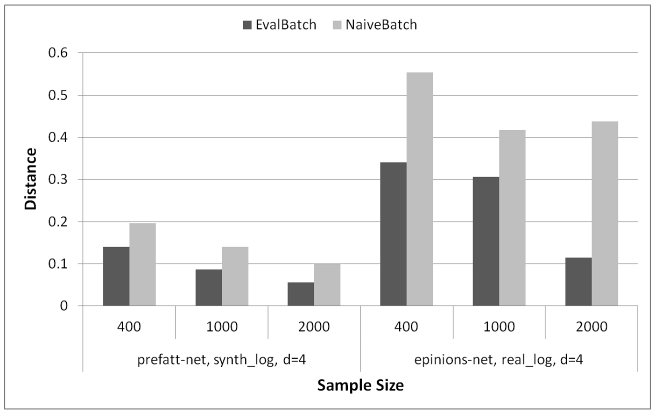


Figure 5.10: Ordering Accuracy: Distance Between Lists

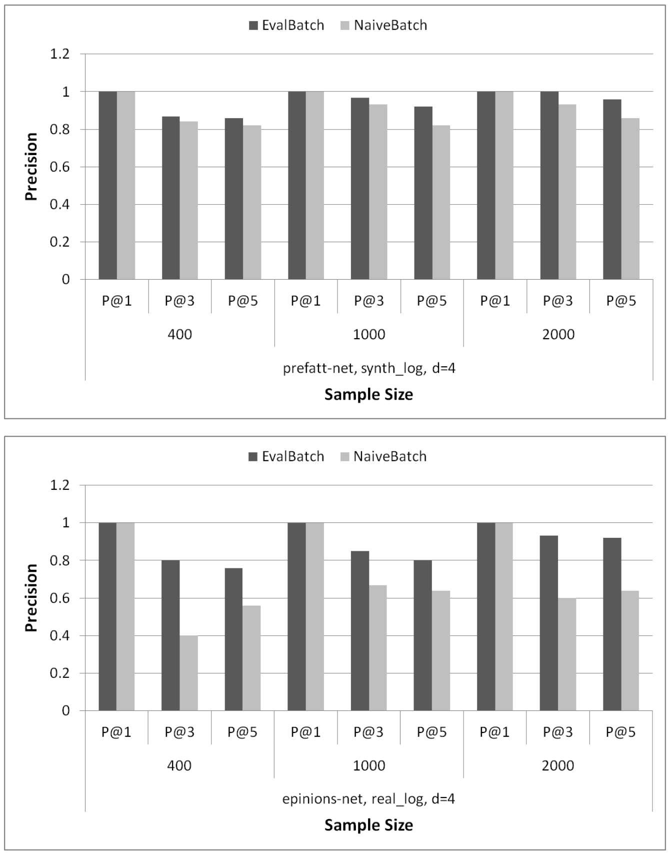


Figure 5.11: Ordering Accuracy: Precision at K for Synthetic (up) and Real (down) Network

Ordering Accuracy

The end objective of our method is to approximate the ordering of a set of urls in a user's neighborhood and not necessarily their exact counts. In this set of experiments we assess the ordering accuracy of the batch sampling algorithms. Formally, for each query q we retrieve the top-k urls returned by a search engine, such as Google. Let this set of urls be r_q. For a given user v and network depth d we first compute the ordering r_{q_v} of these urls in the $D_d(v)$ of v according to their exact counts. Then, we apply any of the batch sampling methods (*NaiveBatch* or *EvalBatch*) to estimate the url counts of each $url_q \in r_q$ and come up with an approximate ordering r'_{q_v}. The two lists, r_{q_v} and r'_{q_v}, are then compared using the *Normalized Spearman's Footrule Distance* and the *Precision at k* metrics. Results are averaged over a number of random users and queries. Figure 5.10 illustrates the accuracy performance of *NaiveBatch* and *EvalBatch* in estimating the correct order of the top-k urls, where $k = 10$. For each method we report the Normalized Spearman's Footrule Distance between the exact and the approximate ordering. We make cases of a synthetic network (*prefatt-net* with *synth_log*) and a real network (*epinions* with *real_log*) for $d = 4$ and variable sample size n. In all cases and in both networks, *EvalBatch* ourperfoms *NaiveBatch* the distance is smaller, signifying a more accurate sampling method. Moreover, as the sample size increases *EvalBatch* the distance decreases, for both network types, which means that our sampling method becomes increasingly more accurate. *NaiveBatch* tends to behave better for larger sample sizes, as well, but this is an artifact of the number of unique users that is able to sample during the sampling process, as such its behavior is not always expected. We further illustrate the accuracy performance of the two methods in correctly estimating the top-k items in the lists. We report on the *Precision at k* (P@K) of the two methods in the case of a synthetic and a real network (See Figure 5.11) for $d = 4$ and variable sample size n. In all cases and for both networks, *EvalBatch* ourperfoms *NaiveBatch* as it is able to retain higher precision, which increases as well with the size of the sample. Note, again, that in the case of the real network topology (*epinions-net* with *real_log*) the ordering accuracy results are not directly comparable to the results in the synthetic network. This is due to the fact that the exact url counts in the real data are much smaller and leverage the sampling performance. As a result, even if trends are identical to the ones observed in the synthetic data, slightly worse performance of the ordering accuracy is demonstrated in the course of both metrics.

5.6 Related Work

Our work is related to work on *sampling large graphs via random walks.* Generating a uniform random subset of nodes of a graph via random walks is a well studied problem; it frequently arises in the analysis of convergence properties of Markov chains (e.g., see [56, 4, 65, 53]) or the problem of sampling a search engine's index [19, 18]. The basic idea is to start from any specific node, say v, and initiate a random walk by proceeding to neighbors selected at random at every iteration. Let the probability of reaching any node u after k steps of this walk be $p(u)$. It is known that if k is suitably large (the value of k depends on the topological properties of the graph), this probability distribution is *stationary* (i.e., it does not depend on the starting node). However, this stationary distribution is not uniform; the probability associated with each node is inversely related to its degree in the graph. This stationary distribution can be made uniform using techniques such as the Metropolis Hastings algorithm (see [63]), or using rejection sampling (where, after reaching a final node, the node is included in the sample with probability inversely proportional to its degree). This process can be repeated to obtain random samples of a desired size. Similar approaches have been employed in [36] where authors describe sampling based methods to efficiently collect information from users in a social graph and in [55] where sampling techniques are used to collect unbiased samples of Facebook. Likewise, in [72] authors design algorithms for estimating the number of users in large social networks via biased sampling, and in [101] sampling methods are proposed to approximate community structures in a social network. Our research presents ways to improve upon these generic random walk methods on graphs by leveraging the fact that we need to sample from the neighborhood of a node v (i.e., a few links away from v).

Our work is also related to work on *personalized and social search*. The premise of personalized search is that by tailoring search to the individual improved result accuracy may be brought off. A vast amount of literature on search personalization reveals significant improvement over traditional web search. In [141], the CubeSVD approach was developed to improve Web search by taking into account clickthrough data of the type “user, query, url”. Further studies showed that taking into account such data and building statistical models for user behavior can significantly improve the result ranking quality [142, 3]. Other approaches exist, as well, that utilize some notion of relevance feedback to re-rank web search results [42, 147]. Social search informed by online social networks has actually gained attention as an approach towards personalized search. In a sense, utilizing information from one’s social environment to improve on user satisfaction is a form of “extended” personalization, with the extent being defined as a function of the neighborhood of an individual in the network. Many ideas have been suggested to realize online social search; from search engines that utilize humans to filter the search results [64], to systems that utilize real-time temporal correlations of user web history logs [122, 121], to tag-based social search systems [149]. Analyses suggest that integration of social search models improves the overall search experience. Our research is complementary as we aim to offer performance improvements, via sampling, to the process of collecting information from user logs by exploring the graph structure offered by a social network.

5.7 Conclusions

Our research suggests methods for quickly collecting information from the neighborhood of a user in a dynamic social network when knowledge of its structure is limited or not available. Our methods resort to efficient approximation algorithms based on sampling. By sampling we avoid visiting all nodes in the vicinity of a user and thus attain improved performance. The utility of our approach was demonstrated by running experiments on real and synthetic data sets. Further, we showed that our algorithms are able to efficiently estimate the ordering of a list of items that lie on nodes in a user's network providing support to ranking algorithms and strategies. Despite its competence, our work inherits limitations of the sampling method itself and is expected to be inefficient for quantities with very low selectivity. A similar problem arises in approximately answering aggregation queries using sampling. Solutions there rely on weighted sampling based on workload information [31]. However, in our context where data stored at each node are rapidly changing this method is not directly applicable. Our algorithms assume that information for each user in a network, such as web history logs, is available. Access to personal information infringes on user privacy and, as such, privacy concerns could serve as a major stumbling block towards acceptance of our algorithms. Systems that utilize our algorithms should adhere to the *social translucence* approach to designing social systems that entail a balance of visibility, awareness of others, and accountability [45].

Chapter 6

Conclusions

As the thesis is structured in a number of self-contained chapters that each deal with a significantly different problem, comprehensive conclusions of our research findings can be found at the end of each chapter. In this chapter we make an effort to summarize our overall contributions and further elaborate on the significance of the findings. We also provide some thoughts and insights on broader issues and discuss possible future research directions.

6.1 Summary of Contributions

We list below a summary of the research contributions of this thesis.

- We have conducted a large-scale analysis of how information cascades among blogs. Our analysis utilized one of the largest data sets available at the time and was distilled into a comprehensive report that contributed to a better understanding of the overall linking activity in the blogosphere. In particular, we analyzed: (i) trends on the degree of engagement and reaction of bloggers in stories that become available in blogs, (ii) the structure of cascades that are attributed to different population groups constrained by factors of gender, age, and continent, and (iii) topic-sensitive information cascades. Our analysis was the first to incorporate heterogeneity (differences in the groups) and revealed notable variations in the structural properties of cascades and in the ability of blogs to trigger prompt and widely-spread cascades that mainly depend on the blogger's profile and the topic of a post. Analyzing information cascades can be useful in various domains, such as providing insight of public opinion and developing better prediction models with applications in health, business, politics and more. More importantly, our study established evidence of a cascading effect of online communication occurring in online social media and stimulated the next research questions.

- We presented a method that given a social network and a log of user actions over a time period, detects and quantifies the occurrence of social influence among peers. This is important because cascading effects in a social network can, to a large extent, be attributed to social influence. We applied this method to a popular online social network (Flickr[1]) and showed that expression of a user behavior (i.e., adoption of Flickr's photo geotagging feature) does not happen randomly, but can to a large extent be attributed to social influence that specific users exert to their friends in the network. The method is generic and can probably be easily adapted in any social system.

- We presented a method that allows to characterize the credibility of users in an online social network by assessing the precision with which they express a behavior. While this credibility metric incorporates characteristics of the behavior itself, the design principle is generic and suggests an interesting strategy for assessing the credibility of users in any social system. User credibility is an essential indicator of the quality of a user's behavior and as such can be utilized in diverse domains and applications that employ some form of user reputation.

- We presented an analytical, experimental framework that allows to investigate the causality between changes of individual behavior and social influence that individuals exert in their network. To the best of our knowledge, our work was one of the first that tried to bridge the gap (and find points of convergence) between aggregate level cascade processes and expressions of individual behavior, using large data sets.

- An interesting finding of our research is that while the quantity of an expressed behavior (how many times) is usually clearly communicated in a social system, the quality of the expressed behavior (how well) is systematically not (e.g., due to lack of feedback mechanisms related to quality). However, the quality of a user's expressed behavior is of paramount importance to a user's social capital (a notion of social influence) that currently largely remains hidden. Our research suggests that users should have the option to reveal certain qualities of their behavior to their social network. Acknowledging the social capital of users offers a design opportunity for social systems.

- It is known that slight modifications in the network topology of a graph, might have a dramatic effect

[1] www.flickr.com

on its connectivity and thus to its capacity to carry on social processes, such as information cascades. We approached network modification as a graph augmentation problem where we seek to find a few non-existing edges to add to the graph. To this end, we presented methods that can accurately and efficiently evaluate the importance of non-existing edges to guide the graph augmentation process. In the augmented graph (i) social processes evolve faster and can reach more nodes, and (ii) random walks can converge a multitude times faster, giving rise to faster graph simulations.

- An interesting, but challenging data mining task in (static and dynamic) social networks is given a specific user (or node) to probe the nodes (and/or the information they are associated with) that are found a few hops away from that user. We presented sampling-based algorithms that given a user in a social network can quickly obtain a near-uniform random sample of users in its neighborhood. Further, we employed these algorithms to quickly approximate the number of users in a user's neighborhood that have endorsed an item. Our methods are highly accurate and very efficient and can be utilized in a number of applications where we aim to rank items in a network, such as, to improve the quality of the user's search experience in a social search engine, or to improve the accuracy of collaborative filtering methods in a recommendation engine.

Overall, the methods and algorithms we have presented are simple to understand and implement, but accurate, very fast and general, so they can probably be easily adopted in a variety of strategies. As such, we expect the thesis to be beneficial to diverse settings, disciplines, and applications ranging from social to technological networks.

6.1.1 Published Work

The results of our research have appeared in the proceedings of international peer-reviewed journals, conferences and workshops. They have also led to two US patent applications. Below, we present the published work in reverse chronological order for easy reference:

- *Refining Social Graph Connectivity via Shortcut Edge Addition.* Manos Papagelis. IEEE Transactions on Knowledge and Data Engineering (ACM TKDD, In Press, X(X), 20XX.) [123].

- *Sampling Online Social Networks.* Manos Papagelis, Gautam Das, Nick Koudas. IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE, Vol. 25, NO. 3, Mar 2013) [126].

- *Suggesting Ghost Edges for a Smaller World.* Manos Papagelis, Francesco Bonchi, Aristides Gionis. Proc. of the 20th ACM Conference on Information and Knowledge Management (ACM CIKM 2011) [125].

- *Individual Behavior and Social Influence in Online Social Systems.* Manos Papagelis, Vanessa Murdock, Roelof van Zowl. Proc. of the 22nd ACM Conference on Hypertext and Hypermedia (ACM Hypertext 2011) [127].

- *Engagement and Reaction in the Blogosphere.* Manos Papagelis, Nilesh Bansal, Nick Koudas. Technical Report. (University of Toronto, 2010) [124].

- *Measuring or Estimating User Credibility.* Vanessa Murdock, Roelof Van Zwol, Manos Papagelis. Publication date: 2010/10/5, Patent office:US, Application number: 12898644 [109].

- *Media or Content Tagging Determined by User Credibility Signals.* Vanessa Murdock, Roelof Van Zwol, Manos Papagelis. Publication date: 2010/10/5, Patent office:US, Application number: 12898654 [110].

- *Information Cascades in the Blogosphere: A Look Behind the Curtain.* Manos Papagelis, Nilesh Bansal, Nick Koudas. Proc. of the 3rd International AAAI Conference on Weblogs and Social Media (ICWSM 2009) [124].

- *Efficient Sampling of Information in Social Networks.* Gautam Das, Nick Koudas, Manos Papagelis, Sushruth Puttaswamy. Proc. of the Search in Social Media Workshop (ACM CIKM/SSM Workshop 2008) [36].

6.2 Future Directions

Research work in this thesis draws several directions for future work. In the following paragraphs, we describe future work that we find interesting in three levels of abstraction: (i) direct extensions of the current research work, separated by chapter, (ii) natural extensions of the current work that define new research problems in (probably) new domains, (iii) longer-term research directions in the area.

6.2.1 Extensions of Current Research

In this paragraph we discuss some extensions of the thesis chapters that can serve as concrete future work.

In **Chapter 2** we presented a case-study of how information propagates in blogs and provided evidence (and characterization) of online communication occurring online. As the social media landscape is dramatically changing over the last years, with more social media services available and bigger and richer user-generated content, it is highly desirable that such case studies be automatically maintained. This calls for tools and mechanisms for real-time monitoring and reporting services that can help to easily characterize and analyze the current state of the social media landscape.

In **Chapter 3** we presented a method for detecting social influence effects in a social network setting. While the method is generic and efficient, there are a few inherent weaknesses of it. First, it is based on a loose definition of social influence (i.e., it models the potential of influence to occur) and second, it does not distinguish social influence to homophily (the tendency of individuals to associate and behave similarly with others). As such, alternative methods can be developed that take care of these weaknesses. Moreover, we presented a measure of user credibility that is based on the wisdom of the crowds. User credibility has become of great importance in recent years where we experience an excess of user-generated content. We believe that further refinements of our credibility measure are possible that could ideally easily be incorporated to online social media services. User credibility can then enable a number of interesting applications, related to the determining the importance of content, reputation schemes and recommendations, spam control and alerting systems, and more. Another direction is related to the observation that we made about the stall design of social systems that can direct affect the social capital (i.e., importance) of an individual in its network. As social influence that an individual exerts to its network is becoming of great importance in many domains (marketing, politics, etc.) users should have the option to reveal specific qualities of their expressed behaviors. This observation challenges the way social systems are currently designed and suggests many design opportunities.

In **Chapter 4** we introduced graph augmentation methods that try to optimize some graph connectivity properties. The methods operate on two phases: (i) in the first phase, all non-existing edges in the graph are ranked based on a utility score (i.e., how much they will help if added in the graph), (ii) in the second phase, an augmentation policy is followed to decide which subset of the set of ranked non-existing edges should be added in the graph, based on some restrictions, to optimize the connectivity property. Therefore, there are ways to improve on the methods by improving any of the two phases. Regarding the first phase, improvements can occur by not having to compute the Dijkstra's single-source shortest path algorithm any time, as this operation alone can be very expensive on large graphs. To this extend, sampling methods can be explored that could compute a uniform sample of all-pairs shortest paths (out of the set of all-pairs shortest paths) by not having to perform the expensive Dijkstra algorithm at all. This can provide the opportunity to scale the computation to much larger graphs. Regarding the second phase, more sophisticated ways of selecting k edges (out of the ranked set of edges) can result in a more accurate augmentation of the graph, and as such, to much better results of our algorithm. An important issue to note is that candidate edges might have an overlapping benefit (e.g., adding the best candidate edge can decrease the benefit of adding the second best candidate edge, and so on), so algorithms that try to optimize on this issue will have a great impact on the methods' performance. The problem can be designed as a variation of the well-known maximum coverage problem, however we have proved in the theses that the function of adding k new edges in the graph is not submodular, thus standard greedy methods won't provide any performance guarantees. Therefore, alternative methods (or heuristics) are needed that can guide the augmentation of the graph.

An interesting outcome of our research is that while we try to minimize the characteristic path length of a graph by adding new edges, in the augmented graph social processes evolve faster and can reach more nodes (i.e., information spread increases). However, while we prove in the thesis that adding a new edge in the graph will always decrease the characteristic path length of the graph, there is no guarantee that the information propagation spread will always be increasing by adding new edges. In fact, adding new edges can also lead to the information spread being contained. For example, if the new edges (shortcuts) are added in the network in a way that is effectively increasing clustering properties of the graph, then there might be cases where the information propagation spread can get smaller. The reason is that by making specific areas of the network more dense, the likelihood of a cascade to break into other clusters is effectively diminishing. In our domain, we do not need to worry about such cases, as our methods were effectively loosening the clustering properties of the network. But, this remains an interesting direction of future research as further theoretical and experimental evaluation is required. Interesting variations of this problem exist as well, where one would like to add or remove edges of the graph, in order to maximize or minimize the information propagation cascade [144].

This chapter also highlighted the fact that the global optimization problem of augmenting a graph by adding new edges can be seen as an alternative for *friend suggestion* in a social network. The premise is that in the augmented network users are more well connected and information can spread more efficiently in the network. Current state-of-the-art friend suggestion algorithms are mainly based on the idea of suggesting friends with which an individual shares a large number of friends; that way the social network tries to maximize the probability of a new connection to occur. However, from a graph perspective, such an algorithm operates only locally by making the local network (cluster) more dense, but does not succeed in optimizing global graph properties (such as better information propagation). An alternative friend suggestion algorithm would try to blend ideas of our graph augmentation algorithms and the current-state-of-the-art by suggesting a mix of friends:

- friends that make the local network more dense (coming from current state-of-the-art friend suggest algorithms), and
- friends that try to optimize global network properties (coming from our graph augmentation algorithms)

It becomes clear that a number of hybrid algorithms is possible and it would be desirable to perform a user evaluation to decide on specific parameters of the hybrid algorithm; the evaluation should seek to find how many friends to suggest from each algorithm and in what rank they should be positioned before presented to the end-user.

It would be interesting to see a methodology that tries to evaluate this alternative to friend suggestion.

In **Chapter 5** we described how we can efficiently sample information that lies in a friend's network and we advocated that such algorithms can be the basis for interesting applications, such as social search engines, or variations of collaborative filtering. To this extend, it would be interesting to see how these algorithms can improve the user experience by designing a number of user-based evaluations in a real system or setting. Moreover, it would be interesting to see how these algorithms can be varied to incorporate specific constraints, such as, geographic constraints, time constraints or even content-specific constraints (e.g., groups of individuals). Some of these variations are straightforward adaptations of the suggested methods, but others offer an opportunity for more sophisticated and challenging algorithms that can be useful in specific domains.

6.2.2 Extensions that Define New Problems

Competitive Influence Maximization in Social Networks

We describe the problem of competitive influence maximization in social networks, which can be collocated in a body of research focused on maximizing the spread of influence in a social network, with applications in viral marketing [41, 131, 75]. In the traditional setting, assuming a social network G, with estimates for the extent to which individuals influence one another and a cascade model M, the problem asks to find the k-node set for seeding the influence propagation process that maximizes the number of influenced nodes at the end of the cascading process.

Recent research has considered the phenomenon of multiple competitive ideas and practices that spread contagiously in a social network forming *competitive cascades* [21, 27, 25, 128, 11, 59, 99]. This problem can also be described as *viral marketing for competitive products.* Formally, given a graph G, a cascading model M, a parameter k and a set of competitive cascades C (e.g., competitive products), we ask to find the k-node set for each cascade $c \in C$ that maximizes its influence (i.e., the number of influenced nodes) at the end of all the cascade processes. The instance of the above problem for $|C| = 1$ reduces to the traditional influence maximization problem. For $|C| > 1$, the problem draws connections to game theory. A number of interesting variations of the competitive cascades problem exists. For example, we can assume different cascade models $M = \{M_1, M_2, ...\}$ for each competitor, non-binary decision of a node (i.e., more than one of the competitive items may be obtained by the same user/node), different budget of each cascade is available (i.e., different sizes of the seed node sets for each cascade $A_0 = \{A_0^1, A_0^2, ...\}$ with sizes k_1, k_2,... accordingly.), cascades are triggered at different times $t_0 = \{t_0^1, t_0^2, ...\}$, where $t_0^1 < t_0^2 < t_0^3 < ...$, there exist geographical constraints of the seed nodes (e.g., availability of free products only in a specific country), the budget of each cascade is allocated in many phases (i.e., not all seed nodes are selected at once), and so on.

Competitive cascades present a great opportunity for various innovative research work in understanding and building theories of large online social systems and social processes with applications in everyday practice. A prominent challenge lies in modeling and mining competitive cascades and developing novel algorithms and tools that scale well. Another challenge deals with the theoretical foundations of such research.

Anticipatory Algorithms in the Presence of Online Social Networks

Not long ago (Dec 2013), Amazon[2] was granted a patent that describes a method for what it calls *anticipatory shipping* [140]. More specifically,

"...a method may include packaging one or more items as a package for eventual shipment to a delivery address, selecting a destination geographical area to which to ship the package, shipping the package to the destination geographical area without completely specifying the delivery address at the time of shipment, and while the package is in transit, completely specifying the delivery address for the package."

The idea is to cut delivery times by predicting what buyers are going to buy before they actually do. We generalize the idea of anticipatory shipping that rather refers to tangible products to include tangible and intangible goods and services that one can purchase online and call it *anticipatory shopping.* There are many ways in which anticipatory shopping algorithms can be designed, deployed and manifested in an online store.

[2]www.amazon.com

For example, a straightforward approach is to base the predictions on scheduled calendar events (such as, holidays or seasonal products), on special personal events (such as, birthdays or anniversaries) or even on repetitive purchases based on a user's history of purchases (such as, yearly subscriptions in an online service or summer vacation packages).

An interesting research direction is to explore opportunities for designing *anticipatory shopping algorithms* in the presence of *social ties* among users, as is the case in *online social networks.* The main assumptions to be made is that the social network structure is given and that each user in the network maintains a *wishlist* - a list of items that the user would eventually like to purchase sometime in the future. Cases where these assumptions are not always true can also be considered. This problem is related to models of *social influence* and *influence propagation.* By monitoring and analyzing the purchasing behavior of users over time, one needs to predict which other users in the network are most likely to buy an item in the near future. Predictions can thus be attributed, to a large extent, to the social influence that specific users exert to their social network.

Overall, given a social network and a set of wishlists, one for each user in the social network, one needs to design anticipatory algorithms that try to predict which items (in the wishlist) and the time at which these items will be purchased. There are many variants of the problem, but in a simple case items in a wishlist have a time of maturity and/or a priority index, and the problem asks to maximize the accuracy of the prediction given constraints of the number of purchases in a specified time period (e.g., a year) and/or user budget (what can the user afford). In the end, each item in a user's wishlist is assigned a probability to be purchased within a time frame and decisions about what to shop are based on a pre-defined threshold. There are several interesting approaches of how the problem of interest can be formulated. In all cases, one needs to utilize the social network of a user to decide which items in a wishlist are more likely to be purchased by the user in the time to come.

To the best of our knowledge, anticipatory shopping algorithms in the presence of social networks is a new problem. In fact, the only context in which anticipatory algorithms are discussed in the literature is related to scheduling problems (e.g., in scheduling hard drive I/Os that seeks to increase the efficiency of disk utilization by "anticipating" synchronous read operations), which is very different to this context. The problem is different to the typical viral marketing problem, where a few influential nodes are sought that can maximize the cascade spread. Here the focus is on the accuracy of purchase predictions of individuals within a constraint time period. It is also different to traditional recommendation algorithms in the context of social networks, as the prediction models are based on influence propagation models among individuals and not on static similarity-based prediction models that try to detect similarities between user profiles in a social network.

The problem can be modeled as a prediction model, where we ask to predict the likelihood with which a user will purchase an item listed in her wishlist within a specified period of time. The outcome of the anticipatory shopping algorithm should be triplets of the form <*user, item, probability*> that represent the *probability* with which a *user* will purchase an *item* in her wishlist. The premise is that products that have a probability over a pre-defined threshold to be pre-ordered before she eventually buys them. Formally, given a graph $G(V, E)$, a cascading model M, a set of goods or services that can potentially be purchased in an online store P, a set of user wishlists W, a time period $\Delta = [0, T]$ and a threshold ϵ, we ask to find the triples (u, p, ϕ) that represent the probability ϕ, where $\phi > \epsilon$, of a user $u \in V$ to purchase the product $p \in P$ at the end of the cascade process.

This defines a novel, interesting and challenging problem with a number of potentially useful applications. Moreover, a number of interesting variations to the original problem can be considered. For example, one can assume that the social network structure or the users' wishlists are not available. One would also like to explore the idea of grouping items (or users) together so that the likelihood of a single item in a group of items to be purchased (or a single user in a group of users to purchase a specific item) is increased, turning anticipatory shopping more efficient and practical.

6.2.3 Longer-term Research Directions

While online social networks are establishing themselves as a rich interconnection and communication tool, an interesting question is how to utilize the online social networks for social good. Below, we are briefly describing how this question can motivate research in many diverse directions.

Decision-making in Social Networks

So far, social media have served as conduits of information. An interesting research question is whether and how online social networks can be employed to take collective decisions. Group decision-making (such as voting) is a situation where individuals collectively make a choice from a predefined set of alternatives presented to them. In this context, social processes among participants, such as coalitions, persuasion and influence become very relevant. Moreover, as these social processes contribute to the outcome, the decision is no longer attributable to any single individual who is a member of the group. While there has been research on decision support systems, these do not operate under assumption of a social setting. An interesting research direction is to introduce algorithmic foundations that can describe and support decision-making in a social setting [24].

Connecting Virtual and Real Social Networks to Solve Real Life problems

While online social networks operate in a virtual world, the individuals that participate in the network have a real-world existence. An interesting research direction is to try to find ways to connect virtual networks and real networks to solve some of the real life problems. As real-life problems, real networks have also a geographic dimension, so there is a need to match geographically dispersed populations to tasks of variable complexity. To a great extend, these problems have a time dimension, so there is a need to solve the problem under specific time constraints. Many variations of the problem exist. For example we might assume the existence of a bounded budget allocated to each task and ask to find what incentives to give to individuals to convince them to participate. Which individuals to target? Note that in this case the position of an individual in the network structure is as important as her geographic location. Therefore, an interesting research question is how to find and engage users of a social network with potentially diverse background and interests under various constraints towards a common goal.

This line of research makes connections to research in *crowdsourcing*. Crowdsourcing refers to the process of completing tasks by soliciting contributions from a large group of people. It combines the efforts of numerous individuals (workers), where individual contributions add a small portion to the greater outcome. Crowdsourcing has gained significant interest for solving various data-intensive problems, such as, annotating labels and image tagging [80, 119]. It is also related to research on finding a team of experts to solve a particular problem or task, assuming that user skills are known a priori and given as input [89, 7, 8]. In order to engage users more effectively, incentives to participate in crowdsourcing tasks can be provided through rewarding mechanisms [10].

6.3 Concluding Remarks

Since their inception, social media have transformed the Web in a powerful communication medium for creating, sharing or exchanging information and ideas. As social media introduce substantial and pervasive changes to communication between organizations, communities, and individuals, their impact is enormous in many aspects of the society spanning from politics and business to the way we learn or socialize. We expect that social media will continue to evolve in the next years, probably at an even faster pace, and while some of its evolution may be invasive, they will continue to provide the ability to create positive outcomes to people's lives and contribute to the society as a whole. We feel fortunate to have witnessed the rapid growth and popularity of online social media over the course of this thesis work and we believe that the challenges (and opportunities) of the future will be equally (or even more) exciting for the research community. The discipline of computer science is suited at the core of this evolution and we anticipate that will continue to play an essential role. The ability to model complex social interactions and processes, monitor and analyze huge streams of user-generated data, design robust algorithms and methods that solve problems efficiently and accurately, build useful applications and tools will remain the focus of the data mining and knowledge discovery community for the years to come.

Bibliography

[1] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: divided they blog. In *LinkKDD*, 2005.

[2] Nitin Agarwal, Huan Liu, Lei Tang, and Philip S. Yu. Identifying the influential bloggers in a community. In *WSDM*, 2008.

[3] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, 2006.

[4] M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic simulation in logspace. In *STOC*, 1987.

[5] R. Albert and I. Barabasi. Statistical Mechanics of Complex Networks. *Modern Physics Reviews*, 2002.

[6] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2), 1986.

[7] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Power in unity: forming teams in large-scale community systems. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 599–608. ACM, 2010.

[8] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online team formation in social networks. In *Proceedings of the 21st international conference on World Wide Web*, pages 839–848. ACM, 2012.

[9] Aris Anagnostopoulos, Ravi Kumar, and Mohammad Mahdian. Influence and correlation in social networks. In *KDD*, 2008.

[10] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Steering user behavior with badges. In *Proceedings of the 22nd international conference on World Wide Web*, pages 95–106. International World Wide Web Conferences Steering Committee, 2013.

[11] Krzysztof R Apt and Evangelos Markakis. Diffusion in social networks with competing products. In *Algorithmic Game Theory*, 2011.

[12] B. Awerbuch. A new distributed depth-first-search algorithm. *Information Processing Letters*, 20(3), 1985.

[13] Baruch Awerbuch and Robert G. Gallager. Distributed bfs algorithms. In *FOCS*, 1985.

[14] Baruch Awerbuch and Robert G. Gallager. A new distributed algorithm to find breadth first search trees. *IEEE Trans. Inf. Theor.*, 33(3), 1987.

[15] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM*, 2011.

[16] Eytan Bakshy, Brian Karrer, and Lada A. Adamic. Social influence and the diffusion of user-created content. In *EC*, 2009.

[17] Nilesh Bansal and Nick Koudas. Searching the blogosphere. In *WebDB*, 2007.

[18] Ziv Bar-Yossef, Alexander Berg, Steve Chien, Jittat Fakcharoenphol, and Dror Weitz. Approximating aggregate queries about web pages via random walks. In *VLDB*, 2000.

[19] Ziv Bar-Yossef and Maxim Gurevich. Random sampling from a search engine's index. In *WWW*, 2006.

[20] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439), 1999.

[21] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. In *Internet and Network Economics*, 2007.

[22] Derek E. Blackman. *Operant Conditioning: An Experimental Analysis of Behaviour*. Routledge Kegan & Paul, 1974.

[23] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4), 2006.

[24] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. Voting in social networks. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 777–786. ACM, 2009.

[25] Matthias Broecheler, Paulo Shakarian, and VS Subrahmanian. A scalable framework for modeling competitive diffusion in social networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, 2010.

[26] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research*, 31, 2003.

[27] Tim Carnes, Chandrashekhar Nagarajan, Stefan M Wild, and Anke Van Zuylen. Maximizing influence in a competitive social network: a follower's perspective. In *Proceedings of the ninth international conference on Electronic commerce*, 2007.

[28] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *WWW*, 2009.

[29] K Mani Chandy and Jayadev Misra. Distributed computation on graphs: Shortest path algorithms. *Communications of the ACM*, 25(11), 1982.

[30] Vineet Chaoji, Sayan Ranu, Rajeev Rastogi, and Rushi Bhatt. Recommendations to boost content spread in social networks. In *WWW*, 2012.

[31] Surajit Chaudhuri, Gautam Das, Mayur Datar, Rajeev Motwani, and Vivek R. Narasayya. Overcoming limitations of sampling for aggregation queries. In *ICDE*, 2001.

[32] To-Yat Cheung. Graph traversal techniques and the maximum flow problem in distributed computation. *IEEE Trans. Softw. Eng.*, 9(4), 1983.

[33] William G. Cochran. *Sampling Techniques, 3rd Edition.* John Wiley, 1977.

[34] Grard Cornujols, Miroslav Karamanov, and Yanjun Li. Early estimates of the size of branch-and-bound trees. *INFORMS J. on Computing*, 18, 2006.

[35] David Crandall, Dan Cosley, Daniel Huttenlocher, Jon Kleinberg, and Siddharth Suri. Feedback effects between similarity and social influence in online communities. In *KDD*, 2008.

[36] Gautam Das, Nick Koudas, Manos Papagelis, and Sushruth Puttaswamy. Efficient sampling of information in social networks. In *SSM*, 2008.

[37] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *OSDI*, 2004.

[38] Erik D. Demaine and Morteza Zadimoghaddam. Minimizing the diameter of a network using shortcut edges. In *SWAT*, 2010.

[39] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), 1959.

[40] Martin Doerr and Manos Papagelis. A method for estimating the precision of placename matching. *IEEE Trans. on Knowl. and Data Eng.*, 19(8), 2007.

[41] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *KDD*, 2001.

[42] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW*, 2007.

[43] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[44] David Easley and Jon Kleinberg. Cascading behavior in networks. In *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[45] Thomas Erickson and Wendy A. Kellogg. Social translucence: an approach to designing systems that support social processes. *ACM Trans. CHI*, 7(1), 2000.

[46] Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4), 1976.

[47] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4), 1998.

[48] Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *FOCS*, 2012.

[49] Greg N. Frederickson and Joseph JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2), 1981.

[50] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3), 1987.

[51] C. Torgeson G. Pass, A. Chowdhury. A Picture of Search. *InfoScale*, 2006.

[52] Minos N. Garofalakis and Phillips B. Gibbons. Approximate query processing: Taming the terabytes. *VLDB Tutorial*, November 2001.

[53] David Gillman. A chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4), 1998.

[54] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12), June 2002.

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

[56] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.*, 63(3), 2006.

[57] Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *SODA*, 2005.

[58] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Mark. Lett.*, 12(3), 2001.

[59] Sanjeev Goyal and Michael Kearns. Competitive contagion in networks. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, 2012.

[60] Mark Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6), 1978.

[61] Daniel Gruhl, R. Guha, Ravi Kumar, Jasmine Novak, and Andrew Tomkins. The predictive power of online chatter. In *KDD*, 2005.

[62] Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkins. Information diffusion through blogspace. In *WWW*, 2004.

[63] W. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1), 1970.

[64] Damon Horowitz and Sepandar D. Kamvar. The anatomy of a large-scale social search engine. In *WWW*, 2010.

[65] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *FOCS*, 1989.

[66] Nicholas A. Christakis James H. Fowler. Cooperative behavior cascades in human social networks. *PNAS*, 107(12), 2010.

[67] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal*, 18(6), 1989.

[68] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1), 1977.

[69] Srikanth Kandula and Ratul Mahajan. Sampling biases in network path measurements and what to do about it. In *IMC*, 2009.

[70] Michael Kapralov, Ian Post, and Jan Vondrak. Online submodular welfare maximization: Greedy is optimal. In *SODA*, 2013.

[71] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1), 1998.

[72] L. Katzir, E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *WWW*, 2011.

[73] Liran Katzir, Edo Liberty, and Oren Somekh. Estimating sizes of social networks via biased sampling. In *WWW*, 2011.

[74] Henry Kautz, Eric Horvitz, Yongshao Ruan, Carla Gomes, and Bart Selman. Dynamic restart policies. In *AAAI*, 2002.

[75] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[76] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[77] Samir Khuller and Ramakrishna Thurimella. Approximation algorithms for graph augmentation. *J. Algorithms*, 14(2), 1993.

[78] Philip Kilby, John Slaney, Sylvie Thiébaux, and Toby Walsh. Estimating search tree size. In *AAAI*, 2006.

[79] Jamie King. Conductance and rapidly mixing markov chains. In *Waterloo Technical Report*, 2003.

[80] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.

[81] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *FOCS*, 2004.

[82] Jon Kleinberg. Cascading behavior in networks: Algorithmic and economic issues. *Algorithmic game theory*, 24:613–632, 2007.

[83] Jon Kleinberg. The convergence of social and technological networks. *Commun. ACM*, 51(11), 2008.

[84] Donald E. Knuth. Estimating the efficiency of backtrack programs. *Mathematics of Computation*, 29(129), 1975.

[85] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The structure of information pathways in a social communication network. In *KDD*, 2008.

[86] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. In *WWW*, 2003.

[87] Timothy La Fond and Jennifer Neville. Randomization tests for distinguishing social influence and homophily effects. In *WWW*, 2010.

[88] Nikolaos Laoutaris, Laura J. Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. Bounded budget connection (bbc) games or how to make friends and influence people, on a budget. In *PODC*, 2008.

[89] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 467–476. ACM, 2009.

[90] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), 2007.

[91] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, 2009.

[92] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *KDD*, 2008.

[93] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

[94] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[95] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, 2008.

[96] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie Glance, and Matthew Hurst. Cascading behavior in large blog graphs. In *SDM*, 2007.

[97] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM*, 2003.

[98] Lászl Lovász. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

[99] Wei Lu, Francesco Bonchi, Amit Goyal, and Laks VS Lakshmanan. The bang for the buck: Fair competitive viral marketing from the host perspective. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.

[100] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4), 2003.

[101] Arun S. Maiya and Tanya Y. Berger-Wolf. Sampling community structure. In *WWW*, 2010.

[102] S. A. M. Makki and George Havas. Distributed algorithms for depth-first search. *Information Processing Letters*, 60(1), 1996.

[103] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[104] Adam Meyerson and Brian Tagiku. Minimizing average shortest path distances via shortcut edge addition. In *APPROX-RANDOM*, 2009.

[105] R. G. Miller. *Simultaneous Statistical Inference.* Springer Verlag, 1981.

[106] Alan Mislove, Krishna P. Gummadi, and Peter Druschel. Exploiting Social Networks for Internet Search. *HotNets*, 2006.

[107] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010.

[108] Stephen Morris. Contagion. *The Review of Economic Studies*, 67(1), 2000.

[109] Vanessa Murdock, Roelof Van Zwol, and Emmanouil Papangelis. Measuring or estimating user credibility. u.s. patent# 2012/0084226 a1., 04 2012.

[110] Vanessa Murdock, Roelof Van Zwol, and Emmanouil Papangelis. Media or content tagging determined by user credibility signals. u.s. patent# 2012/0084302 a1., 04 2012.

[111] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1), 1978.

[112] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1), 1978.

[113] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27, 2005.

[114] M.E.J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), 2006.

[115] James H. Fowler Nicholas A. Christakis. The spread of obesity in a large social network over 32 years. *N Engl J Med.*, 357(4), 2007.

[116] Luhmann Niklas. The world society as a social system. *International Journal of GeneraL Systems*, 8(1), 1982.

[117] Luhmann Niklas. The world society as a social system. *International Journal of General Systems*, 8(3), 1982.

[118] Zeev Nutov. Approximating connectivity augmentation problems. In *SODA*, 2005.

[119] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[120] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[121] Athanasios Papagelis, Manos Papagelis, and Christos Zaroliagis. Enabling social navigation on the web. In *WI-IAT*, 2008.

[122] Athanasios Papagelis, Manos Papagelis, and Christos Zaroliagis. Iclone: towards online social navigation. In *HT*, 2008.

[123] Manos Papagelis. Refining social graph connectivity via shortcut edge addition. *ACM TKDD, In Press*, X(X), 20XX.

[124] Manos Papagelis, Nilesh Bansal, and Nick Koudas. Information cascades in the blogosphere: A look behind the curtain. In *ICWSM*, 2009.

[125] Manos Papagelis, Francesco Bonchi, and Aristides Gionis. Suggesting ghost edges for a smaller world. In *CIKM*, 2011.

[126] Manos Papagelis, Gautam Das, and Nick Koudas. Sampling online social networks. *IEEE Trans. Knowl. Data Eng.*, 3(25), 2013.

[127] Manos Papagelis, Vanessa Murdock, and Roelof van Zwol. Individual behavior and social influence in online social systems. In *ACM HyperText*, 2011.

[128] Nishith Pathak, Arindam Banerjee, and Jaideep Srivastava. A generalized linear threshold model for multiple cascades. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 2010.

[129] Alexandrin Popescul, Rin Popescul, and Lyle H. Ungar. Statistical relational learning for link prediction. In *IJCAI03 Workshop on Learning Statistical Models from Relational Data*, 2003.

[130] Miao Qiao, Hong Cheng, Lijun Chang, and Jeffrey Xu Yu. Approximate shortest distance computing: A query-dependent local landmark scheme. In *ICDE*, 2012.

[131] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002.

[132] Everett M. Rogers. *Diffusion of innovations.* Free Press, 5th edition, 2003.

[133] Elizeu Santos-Neto, David Condon, Nazareno Andrade, Adriana Iamnitchi, and Matei Ripeanu. Individual and social behavior in tagging systems. In *HT*, 2009.

[134] Thomas C. Schelling. *Micromotives and Macrobehavior.* W. W. Norton & Company, Inc., 1978.

[135] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *MSST*, 2010.

[136] Alistair Sinclair. *Improved bounds for mixing rates of Markov chains and multicommodity flow.* Springer, 1992.

[137] Adish Singla and Ingmar Weber. Camera brand congruence in the flickr social graph. In *WSDM*, 2009.

[138] Parag Singla and Matthew Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW*, 2008.

[139] Burrhus F. Skinner. *The Behavior of Organisms. An Experimental Analysis.* Appleton-Century-Crofts, 1938.

[140] Joel R Spiegel, Michael T Mckenna, Girish S Lakshman, and Paul G Nordstrom. Method and system for anticipatory package shipping, December 20 2012. US Patent 20,120,323,645.

[141] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. Cubesvd: a novel approach to personalized web search. In *WWW*, 2005.

[142] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, 2005.

[143] Yuan Tian, Qi He, Qiankun Zhao, Xingjie Liu, and Wang-chien Lee. Boosting social network connectivity with link revival. In *CIKM*, 2010.

[144] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, 2012.

[145] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 22(176), 1975.

[146] Jan Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008.

[147] Qihua Wang and Hongxia Jin. Exploring online social activities for adaptive search personalization. In *CIKM*, 2010.

[148] S. Wasserman and K. Faust. *Social network analysis: Methods and applications.* Cambridge Univ Pr, 1994.

[149] Robert Wetzker, Carsten Zimmermann, Christian Bauckhage, and Sahin Albayrak. I tag, you tag: translating tags for advanced user models. In *WSDM*, 2010.

[150] Yahoo! Geoplanet. *http://developer.yahoo.com/geo/*, 2009.

[151] Zhuojie Zhou, Nan Zhang, Zhiguo Gong, and Gautam Das. Faster random walks by rewiring online social networks on-the-fly. In *ICDE*, 2013.