Social Fingerprinting: Identifying Users of Social Networks by their Data Footprint

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Denise Koessler Gosnell

December 2014

Acknowledgements

I would like to thank my research committee members, Dr. Michael Berry, Dr. Judy Day, Dr. Jens Gregor and Dr. Bruce MacLennan for serving on my committee and providing support and guidance for my research.

I would like to extend my most sincere gratitude to Dr. Michael Berry for serving as my advisor and mentor. Over the past few years, he continually provided motivation and support throughout every development of this research and my academic career. He provided invaluable advice during all stages of this research and helped me find my own path throughout academia. His recommendation to tap into his professional network has made all of the difference in my potential future and career.

In addition, I would like to thank Dr. Cynthia Peterson and Dr. Harry Richards for igniting my research career with funding, mentorship and professional development. Dr. Peterson shared with me her inspiration to her career and it continues to inspire me daily. Dr. Harry Richards provided mentorship throughout all tribulations and successes of my doctoral studies and shaped my young professionalism.

I also would like to thank Dr. Chris Groer for his invaluable insight into the development and motivation of this research problem. Further, I will always be grateful for Michael Adams who made the final home stretch of this project attainable.

Finally, I would like to acknowledge my peers who played an instrumental part in the completion of this PhD. Specifically; I would like to thank Katie Schuman who personally mentored my transition into computer science since my first semester. I will forever be grateful for her continued guidance and support.

Abstract

This research defines, models and quantifies a new metric for social networks: the social fingerprint. Just as one's fingers leave behind a unique trace in a print, this dissertation introduces and demonstrates that the manner in which people interact with other accounts on social networks creates a unique data trail. Accurate identification of a user's social fingerprint can address the growing demand for improved techniques in unique user account analysis, computational forensics and social network analysis.

In this dissertation, we theorize, construct and test novel software and methodologies which quantify features of social network data. All approaches and methodologies are framed to test the accuracy of social fingerprint identification. Further, we demonstrate and verify that features of anonymous data trails observed on social networks are unique identifiers of social network users. Lastly, this research delivers scalable technology for future research in social network analysis, business analytics and social fingerprinting.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In 1914, Edmond Locard started the human privacy discussion by claiming, and later proving, that a minimum of 12 points and a sharp fingermark are all that are required for accurate human identification Champod et al. (2004). A full century later, and with the advent of mobile devices, social networks and wearable technology, humans are generating more personal data than ever before. As reported in Smith (2012), as of 2010 approximately 85% of Americans 18 and older own cell phones and approximately 96% of young adults ages 18 - 26 carry cell phones wherever they go. Naturally, the exploitation (or is it protection?) of human privacy through social network data thrives as one of the most controversial topics among popular media outlets Gross (2013); Gallagher (2013); Knight (2013); Zyga (2013). As such, this conversation calls into question: does a user's social data leave behind a new form of unique human identification? That is, are data trails observed on social networks uniquely distinguishable from one user to the next? If so, what are the new features needed to form a new unique "digital fingermark" and how much data is needed?

This research requires knowledge and insights from a myriad of academic fields including, but not limited to: graph theory, business analytics, applied mathematics, computational social sciences and data analytics. For techniques in business analytics, the computational social sciences aim to quantify human behavior for improved national security, behavioral studies and consumerism Wuchty and Uzzi (2011); Uzzi (1997); Kossinets and Watts (2006);

Berry (2011). With the boom of social networks, the available data for computational researchers has grown exponentially. Given the explosion of data, social networking companies are paying top dollar to understand customer flow, user profiles and fraudulent accounts within their individual networks. For example, mobile giants Verizon and AT&T spent a combined $5.1 *billion* on market analytics and advertising in 2010 Laya (2011). With the high integration of cellular communication into the American society, the data encapsulated by a mobile network is the perfect target for the creation and analysis of a user's social fingerprint. As a result, this research aims to accurately simulate and identify users over time by categorizing and quantifying the components of a social fingerprint.

1.1 Problem Description

A social fingerprint is found within the data generated by social network users. A social fingerprint contains three overarching types of information: the initiator, the selected activity and the recipient. On any given social network, a user chooses how to engage with the network and a specific recipient of the action. On Twitter, users can reply to, favorite, or re-tweet the information of other users. On Facebook, users can select other profiles on the network in order to send a friend request, message, or a wall post. Regardless of the framework, each engagement contains a selected activity and specific recipient on the network. These engagements can be tracked over time and studied to analyze any user's pattern of behavior on the network. The collection of a single user's activity conceptualizes a social fingerprint.

The main question of study is as follows: given social data on a network, is it possible to uniquely identify the network users over time? That is, by examining the patterns of behavior exhibited by two users on a network can we confidently and accurately report if the two profiles of study are, or are not, the same person?

Consider the two images shown in Figure 1.1. Given that both of these images represent two different social network users, we can being to discuss distinguishing attributes between the two users. Even without labels in Figure 1.1, we can draw conclusions regarding the

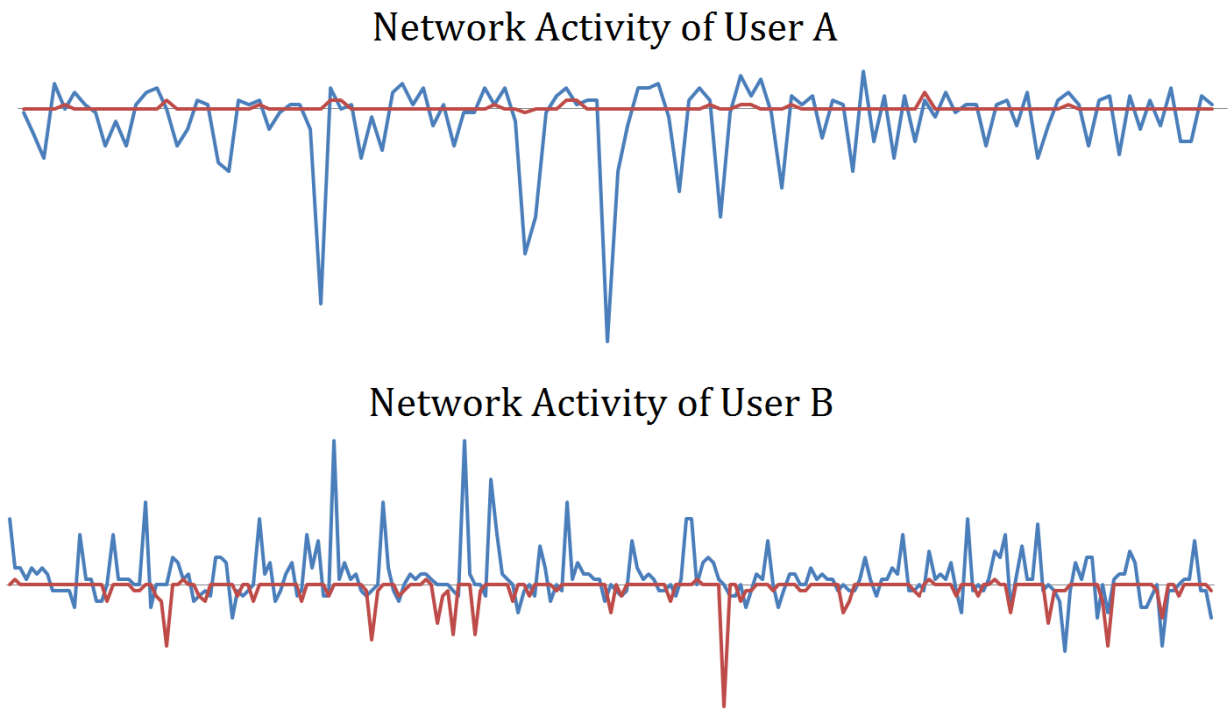


Figure 1.1: The activity of two different social network users over one month's time. The peaks in the positive direction represent incoming activity whereas the peaks in the negative direction represent outgoing activity. The observable differences between User A and User B demonstrate the ability to visually distinguish one data print from another.

distinct differences between the two data trails. For example, we can determine that a signature feature of User A is a large quantity of "blue activity" in the negative direction whereas User B differs with smaller and more frequent peaks. Given similar data trails like those in this example, algorithms in social fingerprinting aim to accurately identify network users by extracting patterns from social data for user identification.

There are numerous elements of this problem which complicate the ability to answer the proposed problem of study in real network applications. First, the data for a social network is generated from one source. That is, we are observing incomplete social graphs. For example, consider Figure 1.2 and let us assume that the social network of study is A. Further, assume that user a from A has friends b and c on social networks B and C, respectively. Then, all information generated or received by user a will be observed; we will

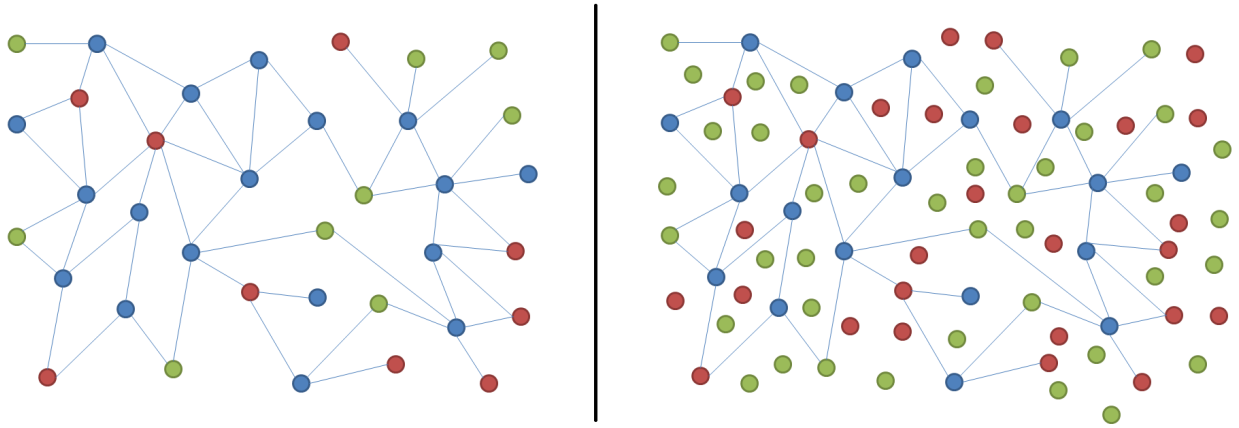


Figure 1.2: An example of network bias when performing cross-platform social analytics. The image on the left illustrates the perspective of the blue social network when attempting to perform common data analytics, such as market share. On the right, we see an example of the hidden network not observed by the social network of interest due to the nature of cross-platform social analytics.

never see communication between b and c. As a result, the quantification of user b's social fingerprint will be biased from the perspective of network A.

Another challenging component of this work pertains to the dynamic structure of social communication. Social data represents human communication that is not identical throughout time. That is, this type of research must account for the dynamic and changing nature of the data of study. Researchers from AT&T labs report in Cortes et al. (2003) that thousands of nodes and edges disappear from their call graphs on a daily basis. Consider Figure 1.3 which shows the dynamic nature of a call graph over 25 weeks of time from Cortes et al. (2003). The authors report that from one week to the next, 1% of all people had not been previously seen in the call network, and 1% will never be seen again. Further, for all of the relationships observed in the AT&T call graph, Cortes et al. (2003) reports that 37.9% of relationships will not be seen from one month to another. These statistics demonstrate the drastic dynamic nature of mining social data.

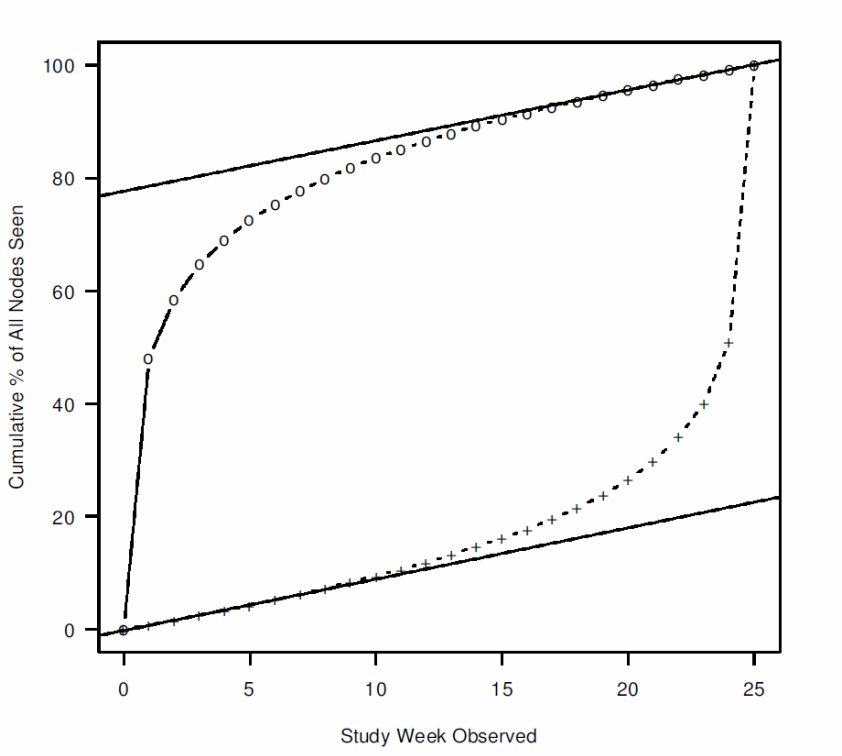


Figure 1.3: The attrition of users in a call graph over 25 weeks from Cortes et al. (2003). The upper curve displays the cumulative percent of unique nodes seen each week that had not been there before. The lower curve showcases the cumulative percent of nodes seen for the last time in each week. A steady state is reached at about 18 weeks.

1.2 Broader Impacts

This research aims to transform social informatics into statistics for advancing government, industry, and academia. The most immediate application of this research is in the area of network security, an avid component in all businesses. The software and methodology herein showcases the ability to identify user accounts across time based on the patterns extracted from social networks. Specifically, in the mobile industry, these social patterns can be used to identify and predict fraudulent accounts, social trends, and market share. Moreover, the techniques presented in this research can be translated across social networks for analogous applications: identification of malicious users, revenue prediction, or network growth.

1.3 Innovation

Previous research and software examine the quantifiable properties found in social network graphs. These studies include in-depth analytics concerning connectedness, graph invariant distribution, and various applications of graph algorithms. Additionally, previous work aimed to identify and predict churning customers in mobile graphs whose behavior was influential on their immediate social network. All considered, previous studies focused on the essential and present components of social networking graphs.

This research aims to define, model and quantify a new metric for social networks: the social fingerprint. The distinguishing aspects of a social fingerprint will be modeled, quantified and tested to determine a methodology that accurately identifies social network users throughout time.

The remaining chapters of this dissertation are as follows. Chapter 2 discusses the related research and software published in the fields of business analytics and social network analysis. Chapter 3 introduces the SOcial Fingerprint Analysis (SOFA) software package for social network analysis with social fingerprinting routines. Chapter 4 presents a community-based approach to detecting separability amongst a community of data trails on a social network. We present a user-based approach to social fingerprinting in Chapter 5 and future research in Chapter 6.

Research support and access to high performance computing resources were granted to the author through the Scalable Computing and Leading Edge Innovative Technologies Graduate Fellowship Program, a National Science Foundation Integrated Graduate Education and Research Training Program, Grant Number 0801540. All tests and simulations in this dissertation were run on the Newton High Performance Computing cluster at the University of Tennessee, Knoxville. The cluster consists of over 332 $x86\_64$ compute nodes with a total of over $4,200$ processor cores Ragghianti (2014). The operating system is Scientific Linux 5.5 and the batch-queue system is Grid Engine. All other cluster documentation is available at Ragghianti (2014).

Chapter 2

Literature Review

The computational social sciences aim to quantify human behavior for improved national security, behavioral studies, and consumerism Wuchty and Uzzi (2011); Uzzi (1997); Kossinets and Watts (2006); Berry (2011). With the boom of social networks, the available data for computational researchers has grown exponentially. However, the majority of cutting edge technologies, data and funding are held within the private sector with little information available to the academic researcher. As such, techniques in graph modeling are dominating the academic space to enable the simulation and analysis of custom graph models. To better understand existing techniques and software, Section 2.1 presents the leading research in social analytics from the perspective of the mobile industry and Section 2.2 gives a brief timeline of approaches in graph modeling. Lastly, Section 2.3 discusses existing software for social network analytics.

2.1 Research in Mobile Analytics

Current research from the mobile industry reports three main applications from mining telecommunications data: hardware fault detection, fraud detection, and customer profiling Weiss (2005). To minimize revenue losses, research involving hardware fault detection attempts to identify areas in which a mobile network does not perform adequately for the mobile users. On another hand, research in improved fraud detection algorithms aims to

minimize a company's losses due to malicious behavior. Lastly, research in mobile customer profiling is used to maximize profits through improved marketing or predicting customer flow Weiss (2005). The common ground between the work in fraud detection and customer profiling lays the story line for quantifying and identifying a social fingerprint.

The Communications Fraud Control Association (cfca.org) estimated that worldwide telecommunications fraud costs approximately between $70 and $78 billion in 2009. As a result, one of the first areas of research involving telecommunications data centered on the classification and detection of fraud from mobile data. Grosser et al. (2005) used a neural network to detect fraud in a live streaming call graph. The authors reported their main problems involved building and maintaining accurate user profiles and applying these profiles to detect live streaming change. They found that the user profile needed to contain a balance of current user patterns mixed with important pieces from history. However, Grosser et al. (2005) concluded that they were able to detect when users who were not using their phone in a traditional way but, they could not ultimately distinguish between abnormal or fraudulent behavior. Becker et al. (2010) confirm these findings and reports that the most difficult aspect of fraud detection is identifying the fraudulent behavior from atypical behavior. Improved technologies for better fraud detection require more accurate user profile data and a better understanding of a user's predictable behavior. Additional research involving fraud detection in mobile call graphs can be found in Schommer (2009); Xing and Girolami (2007); Burge et al. (1997); Rosset et al. (1999).

While improved fraud detection algorithms aim to minimize a company's losses, research in mobile customer profiling is used to maximize profits through improve marketing Weiss (2005). For example, to better understand the flow in and out of a small city, Becker et al. (2011b) examined the user groups found within a suburb of New York City. The authors applied unsupervised clustering to the location based data generated by anonymous mobile users and reported seven types of human behavior found within the data of study.

As seen in Figure 2.1 from Becker et al. (2011b), the authors were able to distinguish commuting cell phone users (the far left plot) from perceived students (the far right plot). Their work further analyzed these groupings by pulling the tower locations for each group's

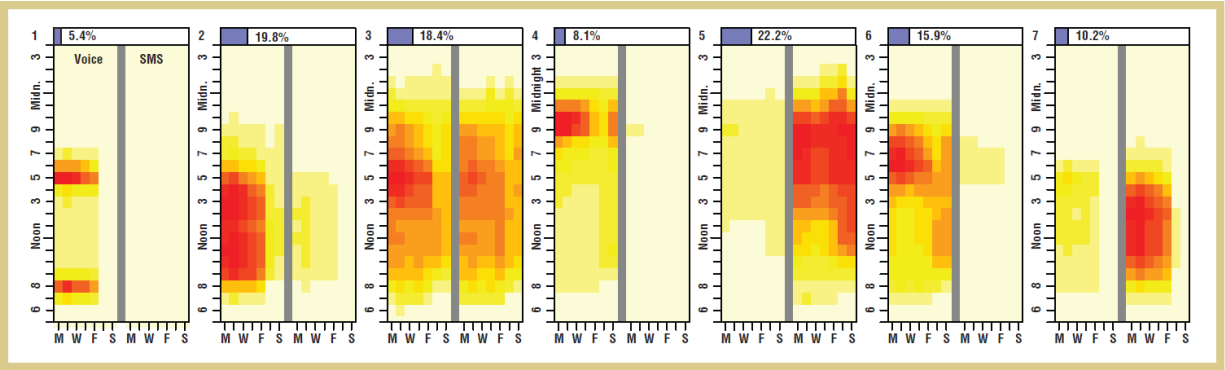


Figure 2.1: The clusters of human telecommunications data from Becker et al. (2011b). For all seven clusters, the left pane showcases the calling frequency and the right pane displays texting frequency. Both panes display the amplitude of volume for the respective data from 6 a.m. until 3 a.m. the following day. The first cluster shows the behavior of commuting workers who primarily talk on their phone before and after the work day. The 7th cluster is theorized to showcase the texting behavior of teenagers during school hours.

calling patterns and confirmed the distinguishing behaviors between commuting workers and students texting at school. Figure 2.2 shows the peak usage for a tower in the downtown area (left image) and a tower near a high school (right image). This work confirms that cell phone users behave in a predictable manner and can be associated with an overarching profile. Additional details on their work were also published in Becker et al. (2011a).

Another main theme throughout current research in the mobile industry emphasizes tower location as the main element for customer analysis. For example, Isaacman et al. (2011) developed a data mining algorithm to detect the home and work location of a group of anonymous cell phone users. The authors created a ground truth data set from 37 volunteers and mapped their findings to a large call graph to detect the main components of human mobility. Next, the findings in Song et al. (2010) report that human mobile behavior is 93% predictable. They report that most individuals are well-localized in a finite neighborhood and few travel widely. However, the authors of Song et al. (2010) selected a very active sample of the mobile user population to study: each user had to be detected between at least two distinct towers and have an average call frequency of 30 minutes.

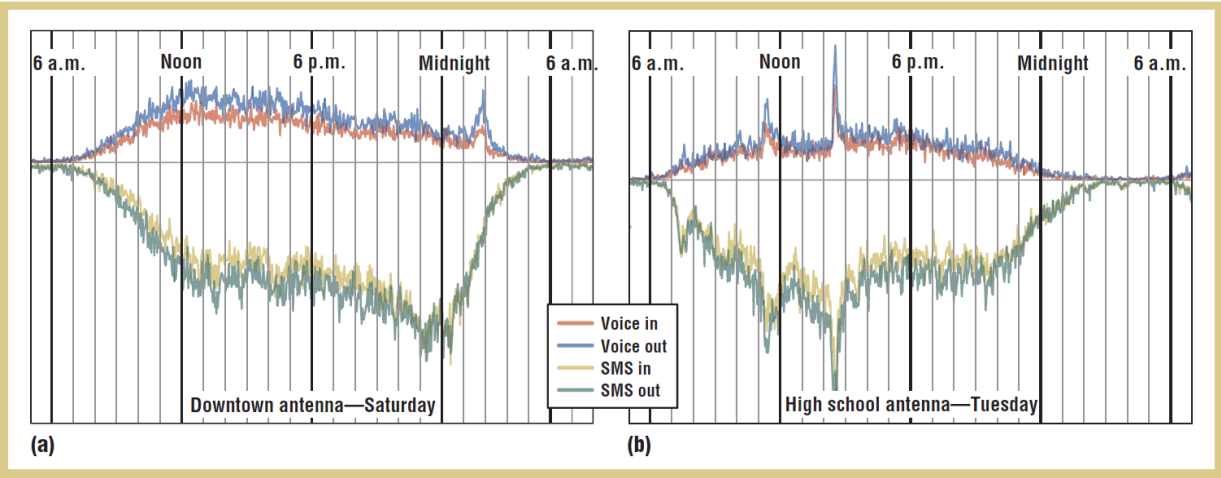


Figure 2.2: The difference between tower usage from Becker et al. (2011b). For both images, the top curves display voice usage and the bottom curves show text usage for the respective towers. The image on the left shows the hourly load for an antenna near the downtown area on a Saturday whereas the image on the right displays the hourly load for an antenna near a high school on a Tuesday.

In 2013, the authors of de Montjoye et al. (2013) laid the ground work for future research in social fingerprints. Their work examined the tower locations available in mobile data for 1.5 million anonymous cell phone customers for 15 months. They demonstrated that four spatial-temporal points are all that is needed to uniquely classify 95% of mobile customers. The publication of their work in *Nature* in March of 2013 created an explosion of coverage by popular science reports on CNN Gross (2013), the MIT Technology Review Knight (2013), and many other online sources Zyga (2013); Gallagher (2013). Although they have demonstrated that the uniqueness of human mobility traces is high, Blumberg and Eckersley (2009) counter that eventually social network users are going to restrict the use of location based services to avoid being physically detected.

As a result, the future of one's social fingerprint is ultimately tied to more general human behavior and the patterns that can be detected from their network activity. So, we are left to ponder: are human interactions on a social network unique?

In consideration of the current boundaries in mobile research, the foundational techniques in social fingerprinting must address two prominent issues with social network data. First,

as demonstrated in Cortes et al. (2003), the dynamic nature of social network data demands that social fingerprint analysis be both robust and flexible. Second, the future of one's social fingerprint needs to be ultimately tied to non-geographical interactions. While de Montjoye et al. (2013) have demonstrated that the uniqueness of human mobility traces is attainable, Blumberg and Eckersley (2009) argue that eventually social network users are going to restrict the use of location based services to avoid being physically detected. As such, this research presents an approach to unique customer identification that does not rely on geo-spatial data.

2.2 A Timeline of Graph Modeling

While recent interest in consumer-based data analytics has re-ignited graph modeling research, the timeline of graph generation dates back to the 1960's. The first graph models randomly connected nodes according to a Poisson distribution Leskovec and Faloutsos (2007). Current work found in Newman et al. (2002) creates random graph models of social networks and concludes that random graph models do not agree with real world social network data due to a lack of social structure within real human networks. The evolution of graph models extended the use of random graphs to include special cases of random graph structures: configuration graphs and exponential random graph models. Configuration graphs are a special case of random graphs which contain arbitrary degree sequences Leskovec and Faloutsos (2007). Exponential random graphs, referred to as p^* *models* among the community, treat each edge of a graph as a random variable that forms a probability distribution over the entire graph Robins et al. (2007). The p^* *models* are primarily used to model small networks Leskovec and Faloutsos (2007).

Watts and Strogatz introduced *small-world* models in 1998 Watts and Strogatz (1998) into the growing field of graph models. The small-world approach improved upon the random graph models by introducing preference in connectivity based on geographical proximity. However, small-world models created node degree distributions which did not match the observable connectivity of real world data Leskovec and Faloutsos (2007). As such, Barabási

and Réka introduced preferential attachment procedures in Barabási and Albert (1999). This seminal paper brought the field of graph modeling to the use of power-law distributions to create models of scale-free networks. In their publication in *Science*, Barabási and Réka reflect that for scale-free distributions the "development of large networks is governed by robust self-organizing phenomena that go beyond the particulars of the individual systems" Barabási and Albert (1999). The main drawback of the first iteration of preferential attachment algorithms is that each node contains the same number of out-links to other nodes in the graph. As such, Kleinberg integrated an edge copying model into the preferential attachment approach to create observable social communities within the overall network Kleinberg et al. (1999). Kleinberg's approach exploited the observable properties of the connectivity of the World-Wide Web in 1999 by mining the Web's link structure Chakrabarti et al. (1999).

With the emergence of power-law distributions within the World-Wide Web link structure in 1999 Chakrabarti et al. (1999), the graph modeling community turned to implementing community-inspired connection algorithms. Some of the main approaches include the forest fire model Leskovec et al. (2005b), Kroneker multiplication Leskovec et al. (2005a) and power law out degree algorithm Palmer and Steffan (2000). Both the forest fire and Kroneker multiplication models are motivated to preserve graph densification over time as observed in real world structures such as the patents citation graph and autonomous systems graph Leskovec et al. (2005c). The forest fire model generates graphs that densify over time and have a shrinking diameter by assigning each node the following two properties: forward burning probability and backward burning probability Leskovec et al. (2005c). Kroneker multiplication models aim to model power-law distributions while exponentially increasing the size of the graph over time and maintaining a shrinking diameter Leskovec et al. (2005a). While the forest fire and Kroneker multiplication algorithms adhere to known structures of real world data such as the patents citation graph, both algorithms specifically aim to model exponentially increasing graphs over time. As such, we are primarily concerned with the more generalized and scalable model for scale-free graph construction published in Palmer and Steffan (2000). For this work, we adapt the power-law out degree algorithm (or PLOD)

in Palmer and Steffan (2000) to simulate social graph topologies according to the published properties of mobile communities in Cortes et al. (2003).

2.3 Current Social Network Modeling Software

Over the past decade, a multitude of studies have begun to examine the interdisciplinary space between marketing, computational mathematics and stochastic agent-based modeling. This niche is home to various social networking simulations which aim to model consumerism, social influence and general topologies of social networks. Of primary interest to the work presented herein are those models which provide foundational knowledge for understanding dynamic, large social networks for consumer studies.

Existing static models construct weighted graphs and examine topological properties to analyze social influence or information flow Gruhl et al. (2004); Kempe et al. (2003); Delre et al. (2007); Janssen and Jager (2003); Liu and Chen (2011). Specifically, Delre et al. (2007) aimed to apply techniques from epidemiological diffusion models as an approach to simulating consumer decision-making as affected by social influences. Furthermore, the model presented in Janssen and Jager (2003) applies an understanding of information diffusion to calculate market share. The simulation built in Liu and Chen (2011) models the viral behavior commonly observed across large social networks.

Most related to this work, the open-source Stanford network analysis project (SNAP) is a large open-source library for graph creation and analysis Leskovec (2014) by J. Leskovec. The SNAP library provides a platform for most general purpose social network analysis and graph mining tasks. For the purposes of this work, we note the use of SNAP to generate synthetic Kronecker graph models in Gomez Rodriguez et al. (2013) to mimic the diffusion of information over time. This study modeled different edge transmission rate evolution patterns for a synthetic network with 1,024 nodes and 2,048 edges. These researchers confirm that the INFOPATH algorithm accurately provides on-line estimates of time varying-edge transmission models and report that they are able to accurately model the virality of internet memes Gomez Rodriguez et al. (2013).

To complement the growing creation of synthetic models, other efforts provide analysis into the constructs of real social network data Bonchi et al. (2011); Bachrach et al. (2012); Gonzalez et al. (2008); Seshadri et al. (2008). Of particular interest is the work published by researchers from AT&T labs which provides insight into the dynamic nature of mobile communication graphs. The research in Cortes et al. (2003) reports that thousands of nodes and edges disappear from the AT&T call graphs on a daily basis. Specifically, they report a 1% attrition rate for nodes and 37% attrition rate for edges in the AT&T mobile graphs. This makes for a large variability in call graphs over various snapshots of time. Additionally, the seminal paper which presents *real* mobile communication network analysis is published by Yves et al. in *Nature* de Montjoye et al. (2013). Their work examined the tower locations available in mobile data for 1.5 million anonymous cell phone customers over 15 months.

In the middle ground between synthetic simulation and real network statistics exists a competitive and growing field called "Business Analytics" or "Business Intelligence", an auspicious division within private industry of statistical social network analysis. Here, social networking companies are paying top dollar to understand customer flow, user profiles and fraudulent accounts within their individual networks. For example, mobile giants Verizon and AT&T spent a combined $5.1 *billion* on market analytics and advertising in 2010 Laya (2011). To address this need, a quickly emerging field of private companies are creating custom software solutions to address the demand for accurate business analytics Alteryx (2014); de C Gatti et al. (2013); Detectives (2014). One of the leading competitors in this space, IBM Research, introduced the Social Media Networks Modeling and Simulation (SMSim) software in late 2013 de C Gatti et al. (2013); Gatti et al. (2013). The SMSim software enables businesses to explore the impact of social media events through simulation. According to de C Gatti et al. (2013), SMSSim simulates social phenomena to capture social influence via word-of-mouth, customer churn, market share, campaigns and other personalized recommendations. SMSim merges simulation with real world data by integrating the Twitter API with a snowball sampling method to synthetically construct a model of a social network. With this integrated approach, SMSim can calculate influence within social media behavior. To demonstrate the effectiveness of this approach, de C Gatti

et al. (2013) modeled Barak Obama's Twitter feed and reported that he was an influential node within the Twitter network. Further details regarding their approach and findings are available in de C Gatti et al. (2013).

While the behavior of mobile users presented in Cortes et al. (2003); de Montjoye et al. (2013) provides interesting insight into human behavior, their work cannot be extended without privy access to the data of study. Furthermore, exclusive competitive business solutions, such as IBM's SMSim, exist due to their monetary success within private industry. Similar to the simulations found in Liu and Chen (2011); Janssen and Jager (2003); Leskovec (2014), the model presented in this work seeks to provide a toolkit for future academic researchers who wish to explore new avenues within social network analytics. Chapter 3 demonstrates that the software created for this dissertation provides a novel approach to the space of social network simulation by creating a multi-edge, scale-free graph via a power-law out degree algorithm for modeling the dynamic structure of large (10 million nodes) social graphs.

Chapter 3

SOcial Fingerprint Analysis Software

3.1 Introduction

The explosion of online social networks has ignited a new field of study for academic researchers Aggarwal (2011); Easley and Kleinberg (2010); Knoke and Yang (2008). Popular social networks such as Facebook, LinkedIn and Twitter have created an expansive collection of data regarding the social behavior of their respective users and consequently, a whole new field in data analytics. At the very core of each social network lies a rudimentary understanding of human communication which is most commonly represented as a graph where people are nodes and interactions between people are edges. First approaches in social network analysis gathered statistics regarding the basic graph theoretic properties of these networks such as diameter, centrality, degree distributions and a variety of other graph properties Aggarwal (2011); Easley and Kleinberg (2010); Knoke and Yang (2008).

However, the rise in popularity for graph analytics as a driver for industrial progress and marketing has ignited a deathly triad of demands for the academic researcher: one for customer privacy, another for data security and a third for more advanced technologies. The coupling of these initiatives has moved the field of social data analytics into the private sector and therefore created an impasse for the academic researcher: we can either get access to common over-analyzed data or develop theoretical representations of advanced methodologies

without testing them on the privy data sets. As such, this dissertation aims to fill the void for future work in academia by providing open-source software with which researchers can test the performance and scalability of cutting edge technologies in social network analysis.

The SOcial Fingerprint Analysis software (SOFA software or SOFAS) aims to combine the published understanding of social relationships over time with stochastic assumptions of network structure to create a scalable environment. We present this model to fill the void in the academic setting for researchers wishing to explore new topics in social network analysis. The primary objectives in the first iteration of the SOFA software are as follows: (1) to model large social networks according to published empirical statistics, (2) to create software that can model up to 10 million users, and (3) to create a dynamic multi-edge graph for analysis. Section 3.2 details each step of the construction process and presents our methods of validation. Section 3.3 presents the statistics on the scalability and accuracy of our model with a discussion in Section 3.4. Section 3.5 concludes with a discussion of the features for a future version of this software.

3.2 Software Methodology

The SOcial Fingerprint Analysis software (SOFA) creates a dynamic multi-edge model of weighted relationships observed in a social network across time based on user-provided settings. A social network is a graph $G(V, E)$ where $V = \{v_i\}$ is the set of vertices in the graph, $|V| = N$, $E = \{e_{i,j}\}$ is the set of edges and $|E| = M$. For this simulation, we construct a dynamic multi-edge graph across time at a set of discrete time points $t_1, t_2 \dots t_n$. At time point t_i, we construct a *scale-free* distribution of relationships across the N nodes in the graph. A scale-free distribution is one that follows an asymptotic power law; further details are outlined in Section 3.2.1. Each observed relationship $e_{i,j}$ is described via a set of disjoint edge types $b_1, b_2 \dots b_m$ and weighted according to the mutual activity level between the pair of vertices. Each edge type has a likelihood of observability $l_1, l_2 \dots l_m$ and is continued from one time step to another based on a given rate of relationship attrition ω.

For reference purposes, Table 3.1 contains a listing of all symbols and terminology used throughout Section 3.2.

Table 3.1: A listing of all symbols and terminology used throughout the remaining sections and subsequent chapters of this dissertation.

Symbol	Usage
$V(G)$	The set of vertices in the graph G
N	Number of vertices (agents) in the model
v_i	A vertex (agent) in the model with unique identifier i
δ	Minimum degree of $V(G)$
Δ	Maximum degree of $V(G)$
c_i	Maximum capacity for vertex v_i
$E(G)$	The set of edges in the graph G
M	Number of edges (behaviors) in the model
b_m	Edge type
m	Number of edge types (behaviors)
w	Weight of an edge
l_b	Likelihood of each edge type
$e(i,j)$	Edge between v_i and v_j
$N(v_i)$	Neighborhood of $v_i : N(v_i) = \{v_j \in V(G^t) \vert e(i,j)_b^t \in E(G^t)\}$
α	Controls the steepness of the power-law distribution
β	Controls the tail of the power-law distribution
t	Time
n	Number of time steps
G^t	Graph at time t
ω	Attrition of an edge from $t-1$ to t
$c_{i,b}^t$	Capacity level for v_i at time t for behavior b
$r_{i,b}^t$	Remaining capacity for v_i at time t for behavior b after capacity flow
$e(i,j)_{b,w}^t$	Edge $e_{i,j}$ at time t for behavior b with weight w

Section 3.2.1 describes the initialization and construction of the vertices in the graph. Section 3.2.2 details the construction of the dynamic relationships from time points $t_1, t_2 \ldots t_n$. Section 3.2.3 outlines the distribution of weight throughout the edges of the graph and Section 3.2.4 presents the underlying algorithm used by the SOFA software followed by a description of validation procedures in Section 3.2.5.

3.2.1 Network Connectivity

In the SOcial Fingerprint Analysis software, the first step towards simulating a full social network is to create the agents in the model and lay the groundwork for connectivity. During this stage of initialization, each vertex v_i in the graph is awarded a maximum potential degree and a maximum capacity c_i. We first describe the construction and distribution of each vertex's maximum potential degree. This section concludes with the procedure for assigning each vertex's activity level c_i.

There are five input parameters which directly affect the maximum potential degree of each vertex: N, α, β, δ, and Δ. Each parameter is defined as follows: N represents the number of agents or vertices in the model and is formally defined as $N = |V(G)|$. The parameters α and β control the steepness and tail, respectively, of the scale-free connectivity between the nodes (see Equation 3.1). An example of a scale-free distribution is displayed in Figure 3.1. The endpoints of the connectivity distribution are δ and Δ, where δ represents the minimum degree and Δ represents the maximum degree within the base model.

It is well understood that the distribution of edges in a social network is scale-free Clauset et al. (2009); Nanavati et al. (2008); Doran et al. (2012). For this model, we implement the power-law distribution defined in Equation 3.1. This scale-free distribution establishes a maximum potential degree for each vertex where the user is able to control the input parameters α and β. Each vertex v_i in the graph receives a maximum potential degree where the probability of award is given by Equation 3.2. That is,

$$p(x) = \frac{\alpha \cdot x^{-\beta}}{AUC}, \tag{3.1}$$

where AUC stands for *area under the curve* and is calculated as

$$AUC = \sum_{i=\delta}^{\Delta} \alpha \cdot i^{-\beta}. \tag{3.2}$$

As seen in Equation 3.2, the final maximum potential degree for any agent in the model is controlled by the user's selection for the minimum and maximum degrees. This normalization

has two purposes. First, it ensures that each agent is awarded a maximum potential degree within the input parameters and secondly, normalizes the cumulative distribution function to 1. An example of the effect of this normalization procedure is shown in Figure 3.1. In this graph, the input power law distribution was created with $\alpha = 3$, $\beta = 2.5$, $\delta = 4$, and $\Delta = 15$. The original power law of $p(x) = 3 \cdot x^{-2.5}$ in the range of $[\delta, \Delta]$ has a cumulative distribution of 0.26. As such, the probabilities for each degree were normalized and the resulting distribution $p(x) = \frac{3 \cdot x^{-2.5}}{0.26}$ has a cumulative distribution of 1.00.

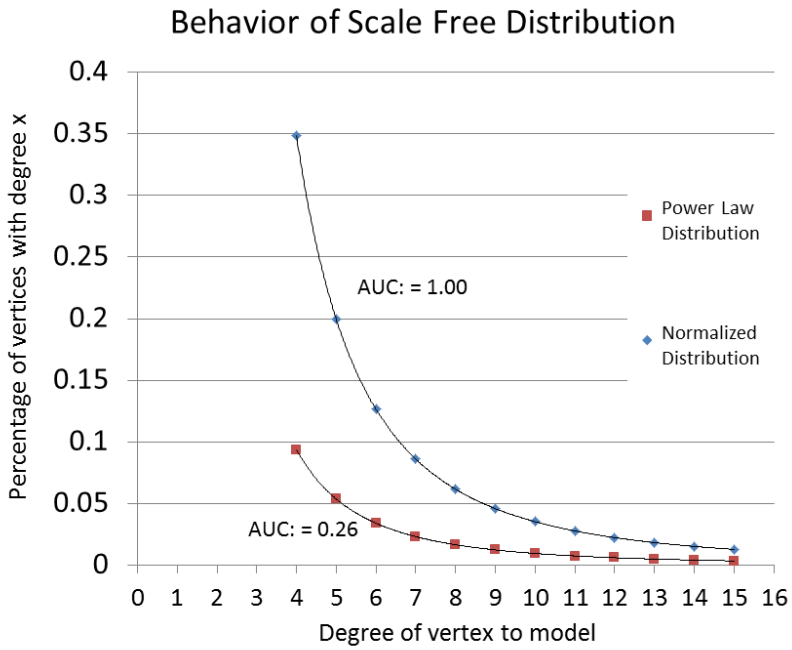


Figure 3.1: An example of the effect of normalization on the power law distribution shown in Equation 3.1. In this graph, the input power law distribution was created with $\alpha = 3$, $\beta = 2.5$, $\delta = 4$, and $\Delta = 15$. The original power law of $p(x) = 3 \cdot x^{-2.5}$ in the range of $[\delta, \Delta]$ has an area under the curve (AUC) of of 0.26. The resulting normalized curve has an AUC of 1.00.

For the first iteration of the SOFA software, the capacity c_i for each vertex v_i is a stochastic assignment which is normally distributed within the interval $[1, 100]$. We denote the capacity of a vertex v_i during time step t for edge type b with $c_{i,b}^t$. This activity initializes the capacity of contribution that each vertex can give to its potential social connections during time step t for each different edge types in the model. Section 3.2.2 describes the

multi-edge model that aims to simulate multiple behaviors between nodes, which is possible on any social network. For example, on Twitter, two users can choose to follow, re-tweet, favorite, or direct message each other thereby forming a multi-edge graphical relationship. To model the weight of these various interactions, Section 3.2.3 details how a user's activity level is diffused throughout the network.

3.2.2 Stochastic Event Creation

An event occurs in the SOFA software when there is an edge between two vertices in the graph: v_i and v_j. An event has three properties, each represented with a superscript for time t and subscripts for edge type (behavior) b and weight w. We let $e(i,j)^t_{b,w}$ represent an edge between vertices v_i and v_j during time step t of type b with weight w. Section 3.2.3 details the construction of the weight of an edge during simulation.

There are two likelihoods assigned during initialization that directly affect the observation of a specific behavior b during time step t: relationship attrition ω and behavior likelihood l_b. At run time, the user specifies the attrition of an edge where attrition models the likelihood that a relationship is *not* observed from one time step to another. The default attrition rate is set to 37% according to the AT&T edge attrition rates reported in Cortes et al. (2003). Next, the user can specify the likelihood that an edge type b is observed throughout the model; the default value for l_b is normally distributed between $[1, 100]$. These two input parameters combine to create the likelihood that any two nodes in the graph interact via behavior b during time step t.

As outlined later in Algorithm 2, the edges are built in the simulation after all of the vertices are awarded a maximum degree potential. As such, an edge is created between two vertices if and only if each vertex has remaining degree potential. To be considered for a relationship, two vertices are drawn at random from the list of all vertices in the model. Since this simulation aims to model up to 10 million vertices, we integrated the Boost Siek et al. (2001) random number generator to ensure a normal distribution across the selection of remaining vertices for any size of graph.

Given that two vertices are connected, additional stochastic elements contribute to the observation of a relationship over time; Figure 3.2 demonstrates this selection process and it is described as follows. Given that two vertices are connected, the relationship is added to the base graph G^0. That is, $e(i,j)^0_{0,1} \in E(G^0)$. The edges added into the graph G^0 represent the base-line of observable relationships between any two nodes in the model. Then, each relationship is spliced up to m times thereby creating a multi-edge model between the two agents. To splice an edge, a die is rolled for each of the m edge types and compared to the behavior's likelihood l_b. If the roll is less than the likelihood l_b for behavior b, than an edge of type b with weight 1 is added to the base graph G^0. That is, $e(i,j)^0_{b,1} \in E(G^0)$. The multi-edge model enables the simulation to mimic the multiple forms of communication between any two people on the same social network. With hardware limitations in mind, the maximum number of edges between any two vertices in the model is 10 and the default value is 3.

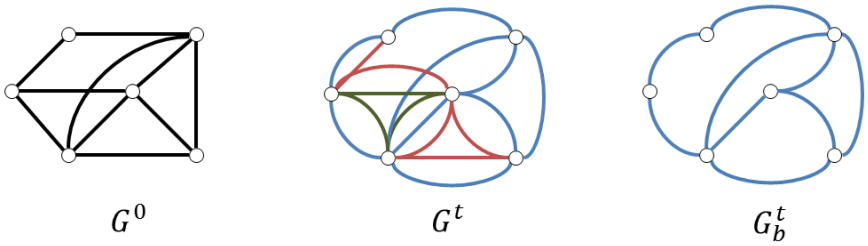


Figure 3.2: An example of splicing the base graph to represent a multi-edge structure. Each relationship shown in the leftmost image goes through a transformation to describe both the behaviors and observances of the relationship over time. The middle image portrays a graph during time step t with three behaviors. The rightmost image portrays the same graph from time step t, but only the subgraph for the blue behavior.

Given the information from the base graph G^0, the global attrition rate ω and behavior likelihoods l_b are used to create observation windows throughout time for each relationship. The likelihood that a behavior b between vertices v_i and v_j is observed during time step t is $(1-\omega)\cdot l_b$. This creates observable dynamic relationships from time step t to $t+1$. As such, all analysis based on applications in social networking that require a dynamic graph should mine the observable graph between time steps $[1,t]$ which are represented in graphs

$\{G^1, G^2, \ldots, G^t\}$. The base graph G^0 serves as a baseline constructor since it represents $\{G^1 \cup G^2 \cup \ldots \cup G^t\}$.

At this stage in the construction process, we have a fully connected multi-edge dynamic graph where each edge in the model has a weight of 1. We call this the *binary graph*. Those applications which require a connected graph can skip the diffusion process in Section 3.2.3 and examine the observable relationships between $G^1, G^2, \ldots, G^t$.

3.2.3 Dynamic Diffusion of Node-Based Capacity in a Multi-commodity Network

For some applications, we seek to model the quantity of social interaction between the agents in our model. To do so, we treat every vertex in the network as both a source and a sink to its immediate neighborhood; thereby establishing a special instance of a max-flow type problem with multiple commodities, constraints on vertex capacities and unlimited edge weight. In this flow network, we aim to optimally disseminate the vertex capacity $c^t_{i,b}$ onto the edges $e(i,j)^t_{b,1} \in N(v_i)$ of v_i during time t for behavior b. An optimal flow would reduce the remaining capacity $r^t_{i,b}$ of each vertex v_i to zero for each time step t and each behavior b. This is a novel instance of a max-flow problem and the algorithm presented herein is a first solution. Definition 3.2.1 provides a formal description of the **maximal diffusion problem** and Algorithm 1 details an approach in $O(|V(E + V \cdot log(V))|)$ time.

Definition 3.2.1. Given a graph $G(V, E)$ where each vertex $v_i \in V(G)$ has capacity $c_{i,b}$, there are m commodities $b_1, b_2, \ldots b_m$, and each vertex v_i is both a source and a sink for the flow of commodity b_m in $N(v_i)$, find an assignment of flow which satisfies the following constraints:

1. Capacity constraints: $\sum_{v_j \in N(v_i)} w(e(i,j)) \leq c_{i,b}$ (the flow of a vertex cannot exceed its capacity);

2. Flow conservation: $r_{i,b} + \sum_{v_j \in N(v_i)} w(e(i,j)) = c_{i,b}$ (the sum of the flow exiting a node and the remaining flow equal the original vertex capacity)

If the **value of diffusion** for vertex v_i is represented by $r_{i,b}$ and the **value of flow** is given by $|f| = \sum_{v_j \in N(v_i)} w(e(i,j))$, then the **maximal diffusion problem** is to maximize $|f|$ such that $\sum_{v_j \in V(G)} r_{i,b} \longrightarrow 0$.

Generally speaking, our approach detailed in Algorithm 1 starts with the vertex v_0 of lowest capacity c_0 and sums all of the capacities of its neighbors. Then, the capacity c_0 of vertex v_0 is proportionally distributed onto the neighboring edges according to the contribution of each neighbor's capacity to the community. Then, each edge weight is deducted from each neighbor's respective capacity and the nodes are re-sorted to account for the updated node capacities. The process re-starts with the lowest capacity node from the re-sorted population.

One of the main challenges to the **maximal diffusion problem** is the demand on a vertex v_i from adjacent neighborhoods. For example, consider Figure 3.3 which steps through one instance of the maximal diffusion problem. At initialization, there is a total weight of $\sum_{v_i \in V(G)} c_i = 172$ to diffuse throughout the network. As observed in steps 1 through 5 of Figure 3.3, the capacity of a vertex is affected by the maximum diffusion of each of its neighbors; thereby influencing the contributing social capacity to the vertex's remaining neighbors. This cascading influence percolates throughout the network as each vertex's capacity is diffused into its immediate neighborhood. As a result, the overall remaining capacity in Figure 3.3 is 34.66, which is 20% of the original total capacity of the vertices in the network.

3.2.4 Algorithm Design

Up to this point, we have detailed each step of the underlying algorithm for constructing a scale-free weighted social network. Algorithm 2 summarizes our customized power-law out degree algorithm and Table 3.2 lists each input parameter in the algorithm.

The design logic of our implementation differs from existing open source models, such as this by Leskovec (2014) and Siek et al. (2001). We prioritize the degree distribution of the

Result: A fractional estimation of an optimally weighted graph via the diffusion of node capacity.
Input: time step t and edge type b;
Sort all nodes such that $c^t_{i,b} \leq c^t_{i+1,b}$;
for *each* $v_i \in V(G)_{sorted}$ **do**
  Initialize: $r^t_{i,b} = c^t_{i,b}$;
  Determine total community capacity C of v_i;
  $$C = \sum_{v_j \in N(v_i)} c^t_{j,b};$$
  where $N(v_i) = \{v_j \in V(G^t) | e(i,j)^t_b \in E(G^t)\}$;
  for *each* $v_j \in N(v_i)$ **do**
    $w = c^t_{i,b} \cdot \frac{c^t_{j,b}}{C}$;
    $e(i,j)^t_b = w$;
    $c^t_{j,b} = c^t_{j,b} - w$;
    $r^t_{i,b} = r^t_{i,b} - w$;
  end
  Sort all nodes starting from v_i such that $c^t_{i,b} \leq c^t_{i+1,b}$;
end

Algorithm 1: An $O(|V(E + V \cdot log(V))|)$ algorithm to solve the **maximal diffusion problem**. This approach starts with the vertex v_i of lowest capacity c_i and sums all of the capacities of its neighbors. Then, the capacity c_i of vertex v_i is proportionally distributed onto the neighboring edges according to the contribution of each neighbor's capacity to the immediate neighborhood of v_i. Then, each edge weight is deducted from each neighbor's respective capacity and the nodes are re-sorted to account for the updated node capacities. The process re-starts with the lowest capacity node from the re-sorted population.

vertices instead of the number of edges in the model. As such, the SOFA software allows the user to control the distribution of node degree within the model instead of pre-determining the number of edges in the model. This design create models which adhere to known network distributions without being constrained to a specific density.

The general logic of the Algorithm 2 is as follows. First, the user input is checked for validity and the nodes are constructed. Then, each node is assigned a maximum potential degree which is equivalent to the maximum number of unique friends which could be observed throughout the model. A summation of each vertex's assigned degree gives an upper bound on the number of potential edges for the base model. Then, for every potential edge in the

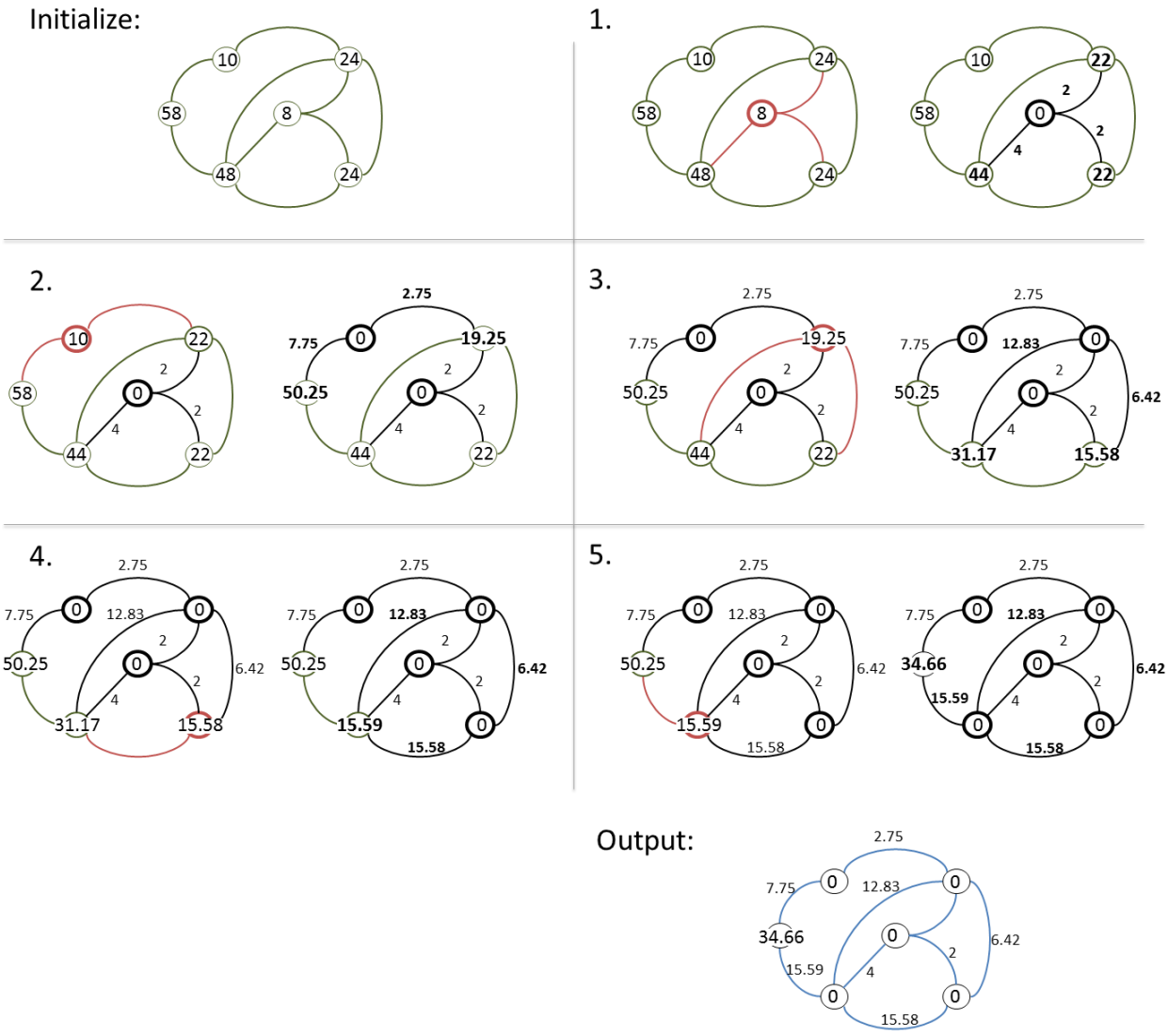


Figure 3.3: An example of the **maximal diffusion problem**. In ascending order according to the capacity c_i of vertex v_i, each neighbor receives a proportion of c_i according to the overall community weight. Each vertex v_j in the neighborhood of v_i receives and updated capacity c_j. Then, the next process is repeated with the next lowest node capacity from the entire graph. After the diffusion process, the remaining capacity in this example is 34.66, which is 20% of the original total capacity of the vertices in the network.

model, we attempt to connect two random people. If a connection occurs, the relationship is recorded in the base graph and transformed throughout time for each of the edge type in the model. This transformation creates a multi-edge dynamic relationship between the two

nodes. If a connection does not occur, we re-draw two random nodes and try the connection again. An upper limit is enforced on the number of connection attempts per edge, therefore providing a small amount of non-determinism in the number of resulting edges in the model.

Table 3.2: A summary of the input settings for the SOcial Fingerprint Analysis software. Each parameter's default values are shown in addition to an accepted range of values. The accepted range of values for each parameter were designed with hardware limitations in mind. For a full description of hardware specifications of the computing resources used in this work, see Ragghianti (2014).

Parameter	Usage	Default	Accepted
N	Number of vertices in the model	10000	$[100, 10000000]$
δ	Minimum degree of V(G)	2	$[2, N-2]$
Δ	Maximum degree of V(G)	$\delta+1$	$[\delta+1, N-1]$
α	Controls the steepness of the degree distribution	1.8	$[1, 4]$
β	Controls the tail of the degree distribution	2.3	$[1, 4]$
m	Number of edge types	3	$[1, 10]$
n	Number of time steps	6	$[1, 100]$
ω	Attrition of an edge from t to $t+1$	37	$[0, 99]$

3.2.5 Validation

Throughout the entire graph initialization process, there are a few checks to ensure that the input parameters construct a valid graph. The validation process includes checking user input for values within accepted ranges and the ability to construct a valid graph with the given input parameters. While these checks are necessary, we only aim to mention their existence for inclusivity in this section.

The main validation procedures of interest are those which quantify the fit of our model to validate the accuracy of the node distribution based construction approach in Algorithm 2. The first validation procedure measures the goodness of fit for the power-law distribution of the observed node degree within the model. The observed distribution is collected by accumulating the total number nodes with degree x where $x \in [\delta, \Delta]$. Then, the observed distribution is normalized according to the total number of vertices in the model. The resulting distribution contains the percentage of nodes in the model with degree x. We

Result: Scale free model of a multi-edge, dynamic social network.

1. Initialize and validate parameters, P = 0 ;
2. Build Nodes: **for** *each* $v_i \in V(G)$ **do**
   - Create a node and assign a unique identifier i;
   - Assign maximum potential degree x where the probability of award is: $p(x) = \frac{\alpha \cdot x^{-\beta}}{N}$ and $N = \sum_{i=\delta}^{\Delta} \alpha \cdot i^{-\beta}$;
   - Increment edge counter: $P = P + x$;

   end
3. Build Edges: **while** $\exists$ P **do**
   - newEdge = false;
   - **while** $newEdge = false$ **do**
     - Randomly draw two nodes $v_i, v_j \in V(G)$;
     - **if** $e_{i,j} \notin E(G)$ *and* $deg(i) < x_i$ *and* $deg(j) < x_j$ **then**
       - **for** *each edge type b to model* **do**
         - **if** $roll() > l_b$ **then**
           - Create $e(i, j)$ in the base graph G^0;
           - newEdge = true;
           - **for** *each time step t to model* **do**
             - **if** $roll() > \omega$ **then**
               - Create $e(i, j)$ in the subgraph G^t

             end

           end

         end

       end

     end
   - $deg(v_i) = deg(v_i) + 1$;
   - $deg(v_j) = deg(v_j) + 1$;
   - $P = P - 1$;

   end
4. Distribute node capacity for each time step t and for each edge type b;
5. Gather graph statistics;
6. Translate the graph for social fingerprint analysis;

Algorithm 2: Construction of the scale free graph model.

implement a least-squares regression Draper and Smith (1981) to fit the observed degree

distribution to a function of the form:

$$y = A \cdot x^B.$$

The least squares fitting yields:

$$b = \frac{n \cdot \sum_{i=1}^{n}(ln(x_i) \cdot ln(y_i)) - \sum_{i=1}^{n}(ln(x_i)) \cdot \sum_{i=1}^{n}(ln(y_i))}{n \cdot \sum_{i=1}^{n}(ln(x_i)^2) - (\sum_{i=1}^{n} ln(x_i))^2},$$

$$a = \frac{\sum_{i=1}^{n}(ln(y_i) - b \cdot ln(x_i))}{n},$$

where $B = b$, $A = e^a$ and $n = \Delta - \delta + 1$ Weisstein (2011). We then calculate the **coefficient of determination**, more popularly known as **R-squared** or R^2, to provide a quantifiable measure to how well the observed degree distribution fits to the model. Section 3.3 showcases the range of R^2 values observed over $5,000$ trials.

Further, we implement the standard **Pearson's chi-squared test** or χ^2 to quantify the error in the model Draper and Smith (1981):

$$\chi_e^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i},$$

where the observed degree distribution in the base graph G^0 is compared against the user's input distribution for α and β. Our hypothesis is that there is no significant difference between constructed distribution and the requested distribution. Section 3.3 contains the range of χ^2 values observed over $5,000$ trials.

3.3 Results

We seek to produce a scalable model in both memory usage and user time. Therefore, we tested the upper limits of our model for memory usage, user time and model accuracy. Appendix A contains complete tables of all results in this section. We discuss all results in Section 3.4.

3.3.1 Memory Usage

First, we examine the total amount of memory required to construct a graph with the SOcial Fingerprint Analysis software as N varies from $10,000$ vertices up to 10 million vertices. Figures 3.4 and 3.5 illustrate the memory required to construct a graph with no edge weights. Figure 3.6 shows the memory required to construct a graph with edge weights when using the node-based diffusion algorithm. As confirmed in the tables given in Appendix A, the standard deviation for all memory tests was extremely low and therefore is negligible for each of the each respective images in this section.

The number of trials varied depending on the size of the graph and is indicated in each corresponding figure; the memory and run time constraints on the Newton limited the number of trials we were able to collect for the larger models. Otherwise, all parameters for each trial were consistent: $\alpha = 1.8$, $\beta = 2.3$, $m = 1$, $n = 1$, $\delta = 2$, and $\Delta = 25$. Memory resources were collected using the Scientific Linux TIME(1) command Keppel (2000).

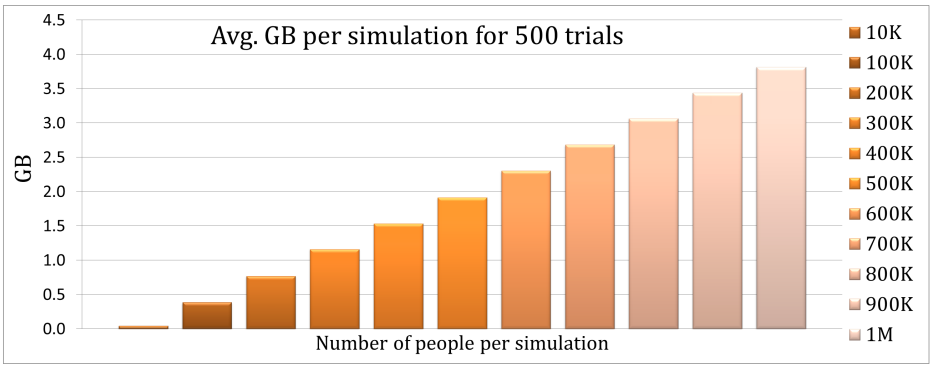


Figure 3.4: The amount of RAM required to execute a simulation of size N without edge weights as N ranges from $10,000$ to 1 million. The total RAM required during execution was recorded for 500 trials for each simulation. The standard deviation for each simulation fell in the range of [0.0 MB, 3.5 MB]. Exact values are shown in Appendix A.

3.3.2 Run Time

Next, we note the total amount of run time required to construct a graph with the SOcial Fingerprint Analysis software as N varies from $10,000$ vertices up to 10 million vertices.

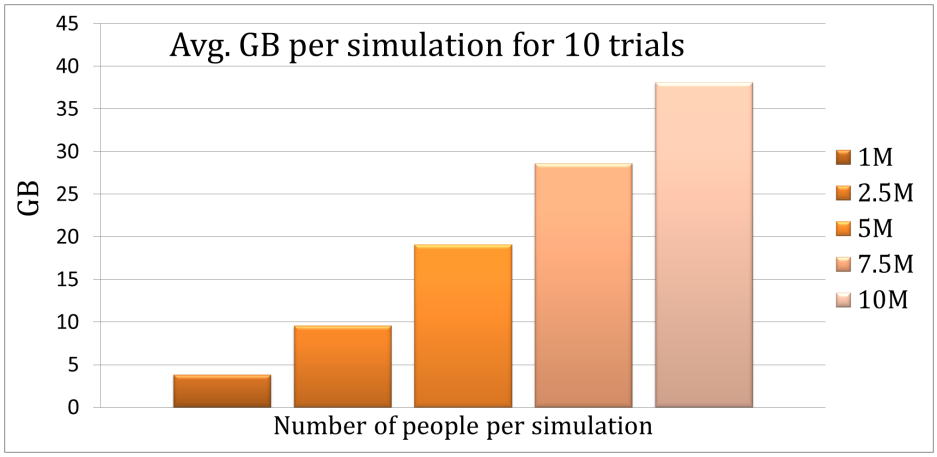


Figure 3.5: The amount of RAM required to execute a simulation of size N without edge weights as N ranges from 1 million to 10 million. The total RAM required during execution was recorded for only 10 trials for each simulation. The standard deviation for each simulation fell in the range of [0.06 MB, 7.9 MB]. Exact values are shown in Appendix A.

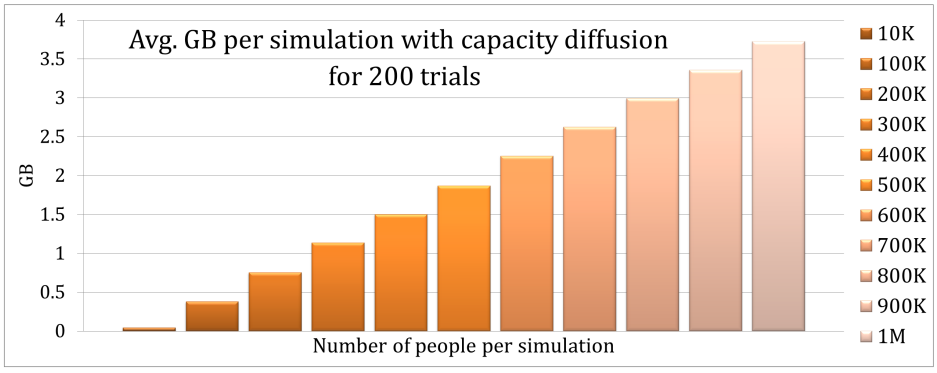


Figure 3.6: The amount of RAM required to execute a simulation of size N with edge weights as N ranges from $10,000$ to 1 million. The total RAM required during execution was recorded for only 200 trials for each simulation. The standard deviation for each simulation fell in the range of [0.0 MB, 2.7 MB]. Exact values are shown in Appendix A.

Figures 3.7 and 3.8 illustrate the run time required to construct a graph with no edge weights. Figure 3.9 shows the run time required to construct a graph with edge weights by implementing the node-based diffusion algorithm. All figures contain an error bar for each test set which shows one standard deviation from the average of the displayed results. See Appendix A for exact values of run times summarized in Figures 3.7 , 3.8 and 3.9.

The number of trials varied depending on the size of the graph and is indicated in each corresponding figure; the memory and run time constraints on the Newton cluster limited the number of trials we were able to collect for the larger models. Otherwise, all parameters for each trial were consistent: $\alpha = 1.8$, $\beta = 2.3$, $m = 1$, $n = 1$, $\delta = 2$, and $\Delta = 25$. Run time and other timing parameters were collected using the Scientific Linux TIME(1) command Keppel (2000).

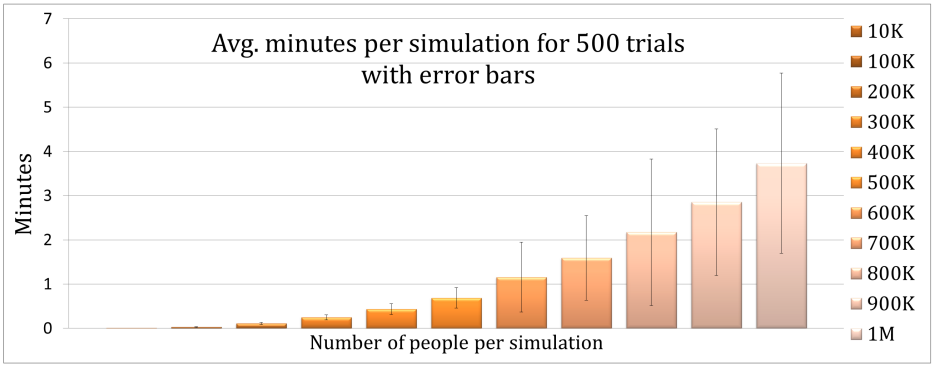


Figure 3.7: The amount of run time required to execute a simulation of size N without edge weights as N ranges from $10,000$ to 1 million. The total time required during execution was recorded for 500 trials for each simulation. The standard deviation across all simulations is shown for each average with an error bar. Exact values are shown in Appendix A.

Noting the significant increase in time from Figure 3.7 to Figure 3.9, we implemented further tests to determine the bottleneck in our implementation of Algorithm 1. Table 3.3 shows the run time needed to construct the graph separate from the time required to diffuse node capacity throughout the network. The final column of Table 3.3 shows the overall percentage of run time spent on the diffusion process.

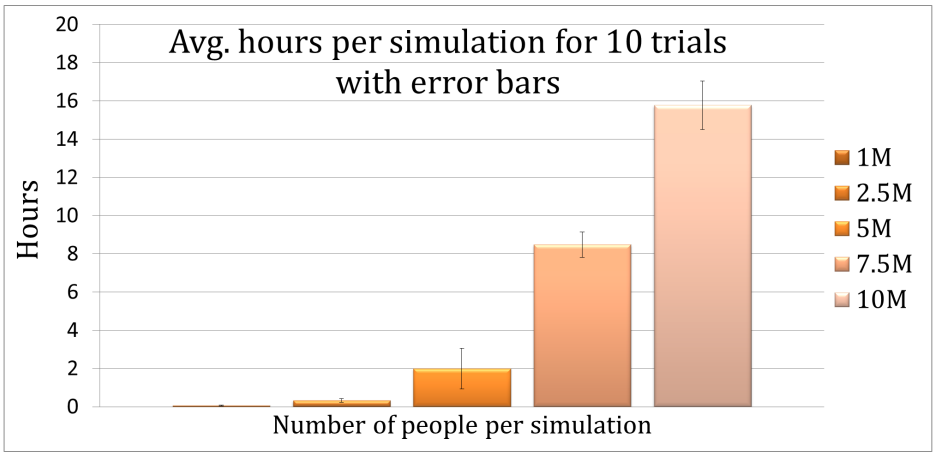


Figure 3.8: The amount of run time required to execute a simulation of size N without edge weights as N ranges from 1 million to 10 million. Due to constraints on computing resources, total time required during execution was recorded for only 10 trials for each simulation. The standard deviation across all simulations is shown for each average with an error bar. Exact values are shown in Appendix A.

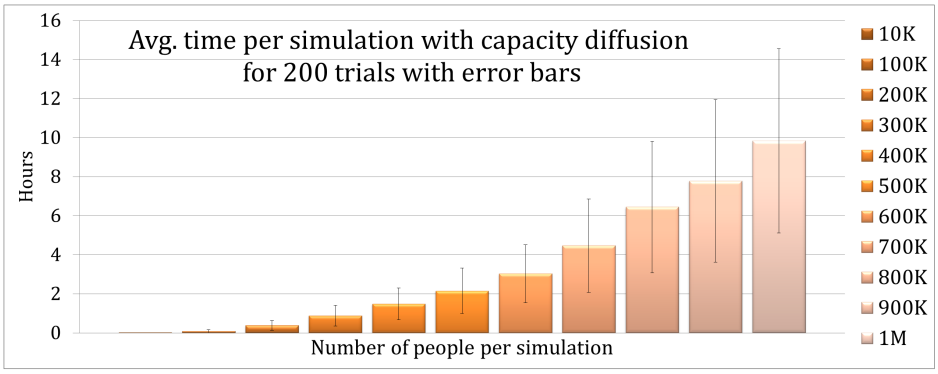


Figure 3.9: The amount of run time required to execute a simulation of size N with edge weights as N ranges from $10,000$ to 1 million. Total time required during execution was recorded for 200 trials for each simulation. The standard deviation across all simulations is shown for each average with an error bar. Exact values are shown in Appendix A.

Table 3.3: Breakdown of average run time in minutes for weighted graph construction.

$\|V(G)\|$	Avg. Time (Create G)	$\sigma(Time)$ (Create G)	Avg. Time (Diffusion)	$\sigma(Time)$ (Diffusion)	% Total Time on Diffusion
10,000	0.0010	0.0005	0.0125	0.0111	0.9178
100,000	0.0330	0.0122	5.4291	4.6237	0.9918
200,000	0.1157	0.0335	22.8966	15.3187	0.9940
300,000	0.2556	0.0836	52.6388	31.4914	0.9944
400,000	0.4563	0.1851	88.7804	48.3318	0.9944
500,000	0.6739	0.3587	128.5984	69.6280	0.9945
600,000	0.9052	0.2923	180.7734	89.2405	0.9946
700,000	1.3066	0.5762	266.9807	143.3318	0.9947
800,000	1.8263	0.8846	385.0150	200.9264	0.9949
900,000	2.3287	1.2578	464.8948	248.5209	0.9948
1,000,000	3.0408	1.8949	587.2191	281.5058	0.9948

3.3.3 Edge Counts

A unique approach to this construction algorithm is the use of the node degree distribution to determine the number of relationships in the model. As such, we observed the resulting number of edges in the G^0. Figure 3.10 illustrates the average number of edges in the model as N varies from $10,000$ to 1 million nodes. Figure 3.11 shows the average number of edges in the model as N varies from 1 million node to 10 million nodes. Due to timing constraints, we only ran 10 trials for the larger models. All figures contain an error bar for each test set which shows one standard deviation from the average of the displayed results. As confirmed in the the tables in Appendix A, the standard deviation for the number of edges in the model was extremely low and therefore is negligible for each of the graphs in this section.

3.3.4 Fit of Model

Next, we implement three different metrics to quantify the overall fit of the model created by the SOcial Fingerprint Analysis software as N varies from $1,000$ vertices up to $100,000$ vertices. Each simulation collected data where $N \in \{1000, 5000, 10000, 50000, 100000\}$. For each given N, we created 961 different test cases as α ranged from $[1, 4]$ in increments of 0.1 and β ranged from $[1, 4]$ in increments of 0.1. Each simulation for a given N, α, and β was

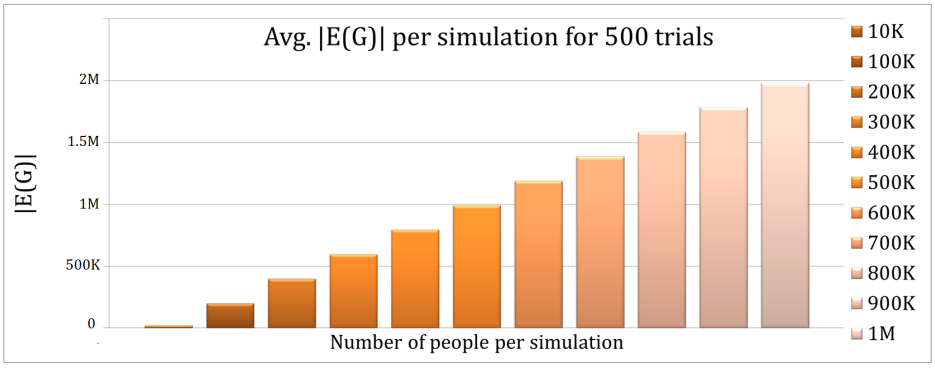


Figure 3.10: The average number of edges observed as N ranges from $10,000$ to 1 million for 500 simulations.

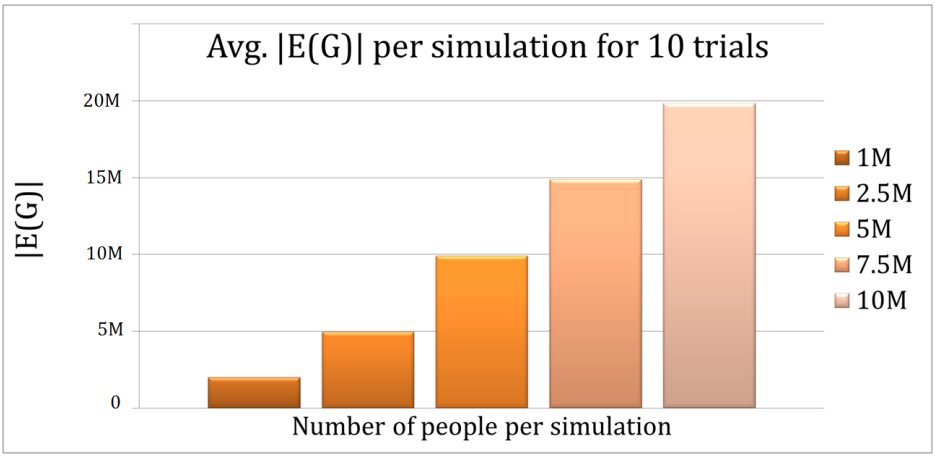


Figure 3.11: The average number of edges observed as N ranges from $10,000$ to 1 million for 500 simulations.

run 5 times, for a total of $24,025$ simulations. Otherwise, all other input parameters were the same for every simulation: $\delta = 4$, $\Delta = 50$, $\omega = 10$, $t = 6$, and $m = 3$. Appendix A contains the separate heat map images for each N across all three metrics.

Percent difference between user input and observed data

The first indicator of correctness is the observed difference between the user's requested parameters and the observed parameters in the model. Recall Equation 3.2 which dictates the maximum potential degree awarded to each vertex during the construction process. In this equation, the parameters α and β are input parameters for the construction process. Then, as detailed in Section 3.2.5, the observed graph is fit to a power law to assess the accuracy of the model. Figure 3.12 shows the percent difference between the input α and the observed α across $24,025$ different simulations. Figure 3.13 shows the percent difference between the input β and the observed β across $24,025$ different simulations. The percent difference is calculated as

$$\frac{|\alpha_i - \alpha_o|}{average(\alpha_i, \alpha_o)},$$

where the subscript i indicates the input parameter and the subscript o indicates the observed parameter. The percent difference between the input β and the observed β is calculated similarly.

In Figures 3.12 and 3.13, the x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across the grids. For each combination of input parameters $[\alpha, \beta]$, the average percent difference over 5 trials is recorded and shaded according to the scale on the right.

Fit of model to observed data

Another metric to assess the validity of the SOFA software measures the goodness of fit of the power-law distribution for the observed node degrees within the model. As described in Section 3.2.5, we implement a least-squares regression Draper and Smith (1981) to fit the

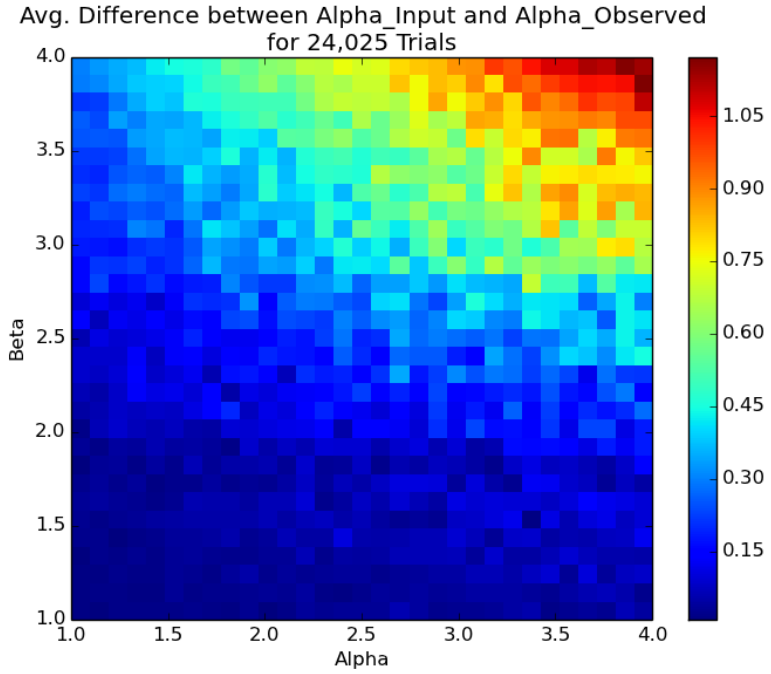


Figure 3.12: The difference between the input α and the observed α across $24,025$ different simulations.

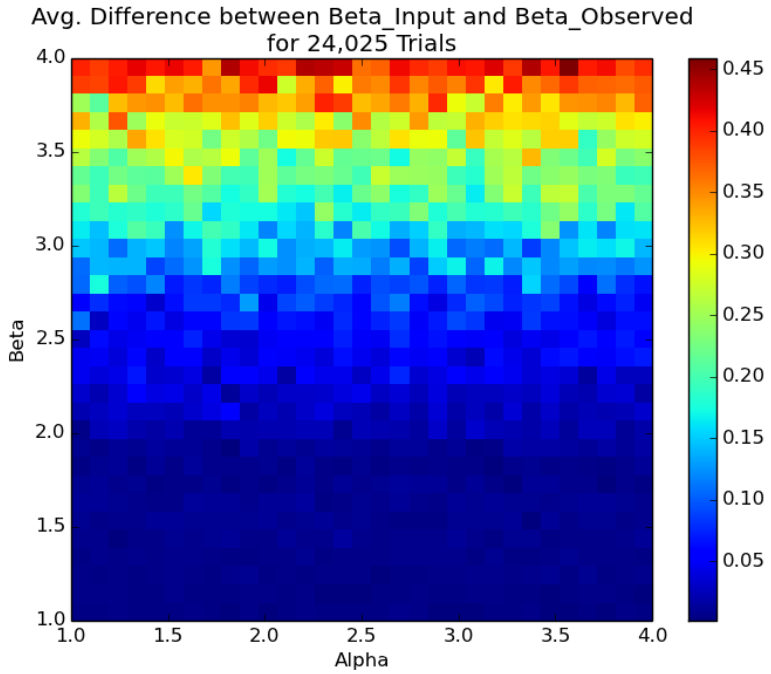


Figure 3.13: The difference between the input β and the observed β across $24,025$ different simulations.

observed degree distribution to a function of the form

$$y = A \cdot x^B.$$

Following the least squares process, we calculate the coefficient of determination, more popularly known as R-squared, to quantify how closely the observed distribution fits to the resulting power-law. Figure 3.14 shows the average R-squared error across $24,025$ different simulations. In Figure 3.14, the x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The R-squared error of the least-squares regression for 5 trials is recorded and shaded according to the scale on the right in Figure 3.14.

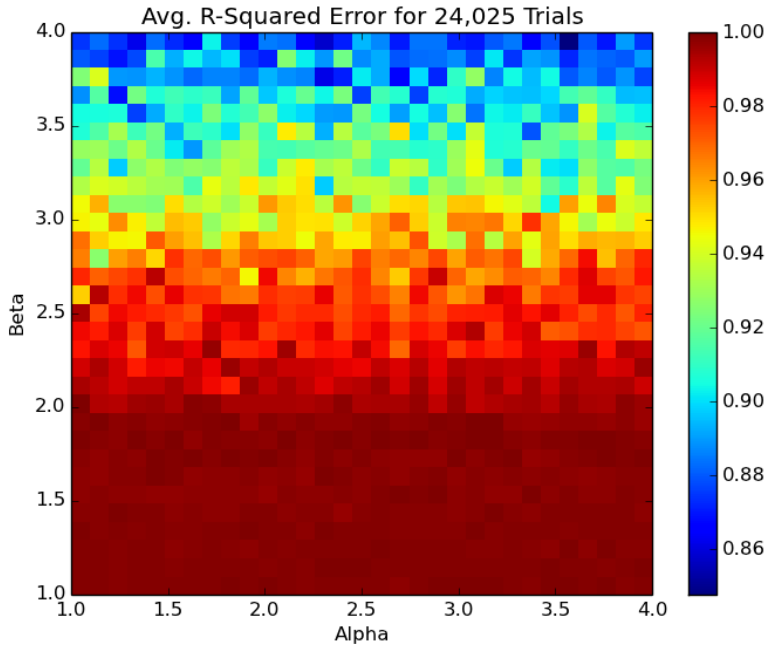


Figure 3.14: The average R-squared error across $24,025$ different simulations.

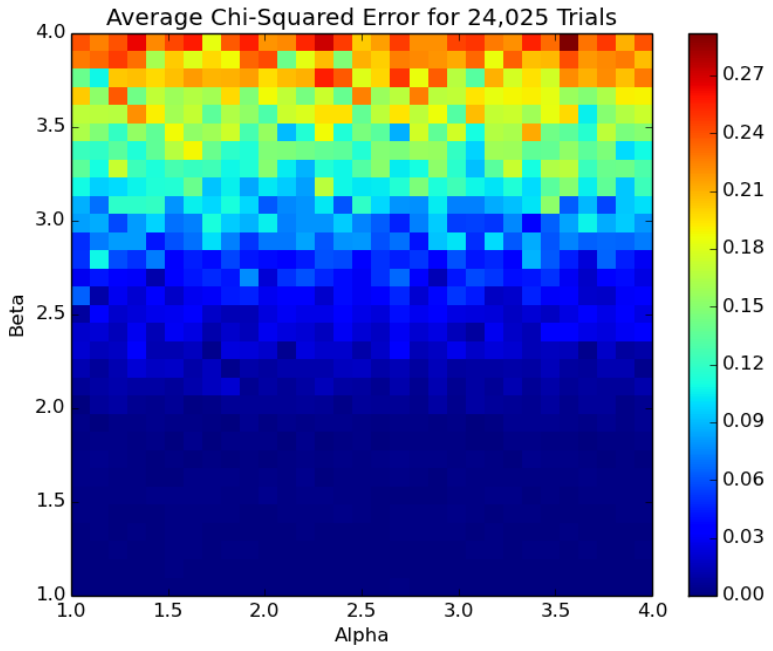


Figure 3.15: The average χ-squared error across $24,025$ different simulations.

Fit of observed model to user input

Lastly, we seek to quantify the correctness between the expected and observed distributions created by the SOFA software. To do so, we examine the χ-squared distribution between the expected distribution, the distribution generated by the input parameters, and the observed distribution. The chi-squared test indicates whether or not the error we observe in our distributions are due to chance, or are a result of one of the parameters in the model. As such, we calculate the χ-squared error to quantify how confident we are that the hypothesis stated in Section 3.2.5 is represented in the data. Figure 3.15 displays the average χ-squared error across $24,025$ different simulations. In Figure 3.15, the x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The χ-squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the right. There are $\Delta - \delta - 1 = 45$ degrees of freedom in the simulations.

3.4 Discussion

3.4.1 Memory and Time Usage

First, we note that the Newton HPC cluster is a shared resource for research at the University of Tennessee, Knoxville. While our jobs were designed to optimize private use of the resources in the grid system, we note that the differences in time and memory usage are affected by a myriad of factors. First, the computing system contains a variety of nodes, each of which have varying amounts of RAM per node. We took steps to ensure that the requested models did not exceed the available RAM on the computing cluster, therefore prioritizing the execution of our job within an adequate environment. Secondly, most tests were run on shared processing space. As such, we calculated the total user run time instead of the wall clock time. This enabled us to measure the time spent on our process and not the scheduling overhead. As with any simulation, results will vary from machine to machine and those presented herein are to provide a guide for future implementation.

Table 3.3 shows that the main bottleneck in using the SOFA software to construct weighted graphs is the serial implementation of Algorithm 1. Specifically, when switching from a binary to a weighted model, Table 3.3 confirms that 99% of the increased user time is spent diffusing node capacity into the network. We note this result is much improved over the original implementation which exclusively used the $C++$ standard template library *sort* routine. We improved our original approach by implementing a linked list data structure and a smart sort routine. As such, the final increase in time is due to the serial implementation of this algorithm which must sequentially traverse each node and edge throughout the large graph.

It is important to note the increase in standard deviation for user run time when $N = 1$ million or more nodes. As shown in Figure 3.5, the largest models were created in only 10 different trials for this research. This was due to limitations on time and memory constraints on the shared computing resources, though the average results in both time and memory for

these larger models could be easily handled by a computing cluster with 50 GB or more of RAM.

The results in Figures 3.4 and 3.7 show promising time and memory constraints for future enhancements. The timing and memory usage of the SOFA software presented herein showcase the feasibility of modeling a non-weighted graph of up to 1 million people on a modest computing platform. While the memory required for both a binary or weighted graph model are negligibly different, it is the overall time required for simulation which is vastly different between the two graph models. The results presented herein showcase the SOFA software's consistent memory usage over 1,000s of trials for both weighted and binary graph models. As a result, we propose that the software presented herein provides a robust and viable vehicle for future research on multi-edge dynamic graphs.

3.4.2 Fit of Model

Most results regarding the fit of the model display statistics that support a high overall accuracy of our model. The most supportive results are shown in Figures 3.14 and 3.15. First, Figures 3.14 demonstrates that the observed distribution has an R-squared value higher than 0.85 for every possible model. Most importantly, for all models where $\beta < 3$, we observe an R-squared error of 0.95 or better. These results continue to improve as we observe the differences in R-squared error across the different simulations as N approaches $100,000$. The series of figures in Appendix A demonstrates that the overall average of observed R-squared error across all combinations of inputs converges to 1 as N increases. That is, our model more accurately matches the requested input distribution as the size of the graph increases.

Further supporting a highly accurate model, Figure 3.15 demonstrates there is no significant difference between the constructed and requested distributions. The chi-squared error is very low and therefore passes the confidence test for even the best of critical values in Pearson's Chi-Squared test. The consistency of our low chi-squared error is further demonstrated in the series of results in Appendix A.

The only result which requires further exploration is the percent difference between the input alpha and observed alpha shown in Figure 3.12. This figure shows very large differences between the input and observed alpha when the model has input parameters $\alpha > 2.5$ and $\beta > 2.8$; Figure 3.12 shows that the percent difference in the input and observed alpha is greater than 50% for this range of models. Recall that in a power-law distribution, α controls the steepness of the curve and as such is very sensitive to small changes in the model. As a result, it is expected to observe more dramatic changes in alpha for it is a more sensitive parameter. Further, consider the full list of figures in Appendix A. These figures show that for models of $N \geq 100,000$, the observed difference in α is consistent for the entire range of possible inputs. As such, Figures 3.12 and 3.13 illustrate the sensitivity of the input parameters α and β for a range of models, but the sequence of heat maps in Appendix A demonstrates convergence for α and β as $N \longrightarrow 100,000$.

3.5 Conclusion and Future Work

During the initial construction and design of this software, some assumptions had to be made due to the lack of published statistics regarding various features in the model. First, as noted in Section 3.2.1, the edge weights were dependent on the activity level of each vertex in the graph and each vertex's activity level was a stochastic assignment from a normal distribution. This made the assumption that human activity on a social network is normally distributed between very inactive to very active. Given published statistics at the vertex level, a second iteration of this software could improve upon this assumption by assigning activity levels to each person based on empirical data. Further, in order to create a base model, each vertex's activity level was uniform across time and each activity. As such, future research could implement more empirical-based distributions on the dynamic structure of user activity throughout both the discrete time steps in this model and different event types.

Another assumption in our model was to construct a node-based model over an edge based model, as previously designed in Leskovec (2014). As such, one of the design choices

made during construction was to not permit the user to pre-determine the number of edges in the model. This is a unique approach to the construction process over existing open-source simulations. This design choice was made to give precedence to the input distribution instead of the sparsity of the model. Further, the modular design of the SOFA software enables the user to implement different input distributions in the node construction process without needing to manipulate any other portion of the code. As such, future work for the next iteration of this software would be to add in a feature which implements different degree distributions during the construction process such as a log-normal distribution or those observed in Seshadri et al. (2008). Further, one could study the effect of this choice on the number of edges in the model.

Next, current published statistics do not address the dynamic or distributed nature of observable human relationships in a social network. At the time of this study, little to no information was available in regards to the distribution of different relationship strengths, quantities or types over time. As such, a major assumption in this model was to create edge weights based on each individual's overall contribution to his or her immediate social community. This approach applies a node-based diffusion model to quantify social network activity instead of an edge-based study. Given empirical statistics regarding the distribution of either approach, a second iteration of this software could adjust the initialization of the weight functions to create a different approach to modeling the weights of observed relationships.

For future research in weighted graphs, the main bottleneck in using the SOFA software to construct weighted graphs is the implementation of Algorithm 1. Future research would aim to improve upon our time to estimate an optimal solution to the maximal diffusion problem, defined in Section 3.2.3.

The motivation behind the creation of the SOcial Fingerprint Analysis software was to construct a simulation which accurately models known social network constructs at the time of publication. Further, we aimed to create robust software which can model large, dynamic, multi-edge human networks for academic research. The results presented herein support that we achieved our goals and have created a platform for additional future work. Given the

assumptions and approach outlined herein, we conclude that accurate graphs can be created by the SOFA software using a modest high-performance computing environment. Additional discussion regarding future versions of this software are presented in Chapter 6.

With the construction of a multi-edge, dynamic social network model from the SOFA software, we are perfectly positioned to explore new problems and methodology in social network analytics. Novel techniques for social fingerprinting are of particular interest. As such, the following chapters of this work present two foundational approaches in social fingerprinting: a community based approach for detecting individual separability versus individual print detection from tracing a users's immediate social neighborhood. In Chapter 4, we test the ability to distinguish individuality amongst a community of data trails by creating graphs with the SOFA software and translating them into sparse matrices for techniques in data compression and information retrieval. Then, Chapter 5 presents an individual tracing algorithm which aims to accurately identify a user over time by only examining the properties of his/her immediate social neighborhood. We discuss the findings of both approaches and the open problems of this research in Chapter 6.

Chapter 4

Social Fingerprinting with Large-Scale Matrix Factorizations

"Realizing you're completely unique... even in a crowd." -Antony John

4.1 Introduction

This chapter introduces and describes a community-based approach for distinguishing multiple users from one another. This technique is motivated by the need to create separability amongst observed data trails as a means to establish uniqueness amongst the users of the network. We use the SOFA software to generate a community of social network users and then outline a procedure for data collection and user identification. We implement matrix decompositions to create a condensed representation of the social network. Then, we use cosine similarity to quantify the closeness of any two users. We hypothesize that the application of techniques in data reduction and information retrieval will enable us to distinguish individual users within the community of data.

The remaining sections are organized as follows: Section 4.2 describes the data used for analysis in this work. Section 4.3 details the application of semidiscrete matrix decomposition

for unique user identification. The results of this technique are presented in Section 4.4 and the concluding remarks are discussed in Section 4.5.

4.2 Data

We generated multi-edge dynamic graphs with the SOcial Fingerprint Analysis (SOFA) software. Given a range of input parameters, the SOFA software creates a customizable model of a dynamic social network which can then be translated into a variety of social network analysis applications. For a full description of the SOFA software, its capabilities and performance, see Chapter 3.

We generated dynamic graphs $G = (V, E, T)$ and varied the number of nodes $N = |V(G)|$ (persons) in the model between 100, 500 and 1000. For additional parameters, we rely on the statistics of the AT&T Mobile network published in Cortes et al. (2003) to most accurately create dynamic network models. We designed two main series of simulations for observation with $N \in \{100, 500, 1000\}$. For the first set of simulations, the attrition of a relationship was set to 37% and $t = 6$. This simulation modeled the month-to-month nature of the mobile graphs presented in Cortes et al. (2003). For the second set of simulations, the attrition of a relationship was set to 11% and $t \in \{2, 4, \ldots, 52\}$. This simulation modeled the week-to-week nature of the mobile graphs presented in Cortes et al. (2003). All other parameters are set to their default values, as discussed in Chapter 3.

4.3 Method

To examine the accuracy of social fingerprinting we employ the semidiscrete decomposition of dynamic graphs for information retrieval. This approach to social fingerprinting mirrors the well-documented approach called *latent semantic indexing* Dumais (1991). Algorithm 3 outlines the approach for applying matrix decompositions and information retrieval concepts to address the social fingerprinting problem. Each step is detailed below.

Result: Validation Measurements of SDD for Social Fingerprinting
for $t_p = [1, p] \cup [p + 1, t]$ **do**
    1. Generate a dynamic graph with SOFAS;
    2. Construct the community matrix A and feature vectors;
    for $k = 5\%$ *of* $|V(G)| = N$, *increment by 5%* **do**
        3. Perform SDD of matrix A;
        4. Transform query vectors;
        5. Determine the best match for each feature vector via cosine similarity;
    end
end

Algorithm 3: Validation Procedure for Social Fingerprint Algorithm via Semidiscrete Decomposition

4.3.1 Construct Community Matrix and Feature Vectors

The validation of this approach is made possible through the division of the dynamic graph into two disjoint sets of data. The graph $G = (V, E, T)$ created by the SOFA software is a dynamic graph over a window of time $T = [0, t]$. Once constructed, the main time window is split into two continuous, disjoint intervals: $t_p = [1, p] \cup [p + 1, t]$, where $p \in \{10\%, 30\%, 50\%, 70\%, 90\%\}$. The training data consists of all observed social interactions during the first time interval, $[1, p]$, and is aggregated into matrix $A = [a_{ij}^p]$ where $a_{ij}^p = w$ describes the amount of communication observed between user i and user j during time $[1, p]$. In the binary models, $w = 1$ if any communication was observed during the time window of interest. In weighted models, w is a summation of the weight of all observed communication between user i and user j during time window $[1, p]$. In all models, $a_{ij}^p = 0$ if user i and user j did not communicate during the time window of interest. Matrix A is referred to as the community matrix of the users of study.

The testing data consists of all observed social interactions during the later time interval $[p+1, t]$ and is aggregated into a set of feature vectors $V^p = \{\mathbf{v}_i^p\}$ to describe each individual user i within the network. In $\mathbf{v}_i^p$, $[v_j^p] = w$ describes the amount of communication observed between user i and user j during the time window$[p + 1, t]$. The weight w in $\mathbf{v}_i^p$ follows the same procedure as applied in the construction of A: $w = 1$ in the binary models if any communication was observed or w is a summation of the weight of all observed

communication between user i and user j in weighted models. As before, $w = 0$ if user i and user j did not communicate during the time window of interest. With this construction, the patterns in community separability observed during time $[1, p]$ are used to predict the identity of each social network user in time $[p + 1, t]$.

4.3.2 Semidiscrete Decomposition

Following the construction of A, we applied semidiscrete decomposition (SDD) to matrix A to obtain a low rank approximation of A O'Leary and Peleg (1983). The choice of SDD over other matrix factorizations was based on the savings of both storage and query time, two elements of high priority when analyzing large social networks Kolda and O'Leary (1998). Formally, semidiscrete decomposition is a matrix factorization technique that yields approximations (A_k) to the original matrix A such that

$$A_k = X_k D_k Y_K^T, \tag{4.1}$$

where each $m-$vector x_i and each $n-$vector y_i are constrained to have entries from the set $\{-1, 0, 1\}$ and $D = diag\{d_1, \ldots, d_k\}$. This decomposition was originally introduced by O'Leary and Peleg in O'Leary and Peleg (1983) and has been shown to save both storage and query time in large-scale information retrieval applications Kolda and O'Leary (1998). Further, as defined in Kolda and O'Leary (2000), we are also interested in the error produced by this approximation given by

$$||A_k - A||_F^2, \tag{4.2}$$

where F^2 is the Frobenius norm. We refer to this measure as the relative residual norm of the SDD.

4.3.3 Translation of Query Vectors

Given the SDD of matrix A, the set of query vectors $Q = \{\mathbf{q}_i\}$ are defined by

$$\mathbf{q}_i = \mathbf{v}_i^T X_k D_k^{-1}, \tag{4.3}$$

where $\mathbf{v}_i$ is the feature vector for user i during time $[p+1, t]$.

With the above transformations, the cosine similarity between each query vector $\mathbf{q}_i$ and each column of matrix Y is calculated. This technique successfully identifies a social network user when the j^{th} column of Y is determined to be the closest in similarity to the query vector q_j. This technique fails to correctly identify a social network user when the j^{th} column of Y measures to be closest to any other query vector q_k, $k \neq j$.

4.4 Results

4.4.1 Binary and Weighted Models

Our first set of results demonstrate the quantifiable differences between constructing binary or weighted matrices. We examined 84,000 samples of varying windows of time and different sizes of social network communities. Given the size of the community, the SOFA software created a dynamic social network model that was analyzed using both a binary and weighted representation of the observed social communications. Each analysis divided the training and testing data into five different subsets and tested the accuracy of each division: 10% training data with 90% in test, 30% training data with 70% in test, 50% training data with 50% in test, 70% training data with 30% in test, 90% training data with 10% in test. All results not shown in this section are provided in Appendix B.

Figure 4.1 displays the overall accuracy for the social fingerprinting pipeline for 100, 500 and 1000 users. Figure 4.2 displays the final relative residual norm for each of the 84,000 matrix reductions observed in Figure 4.1.

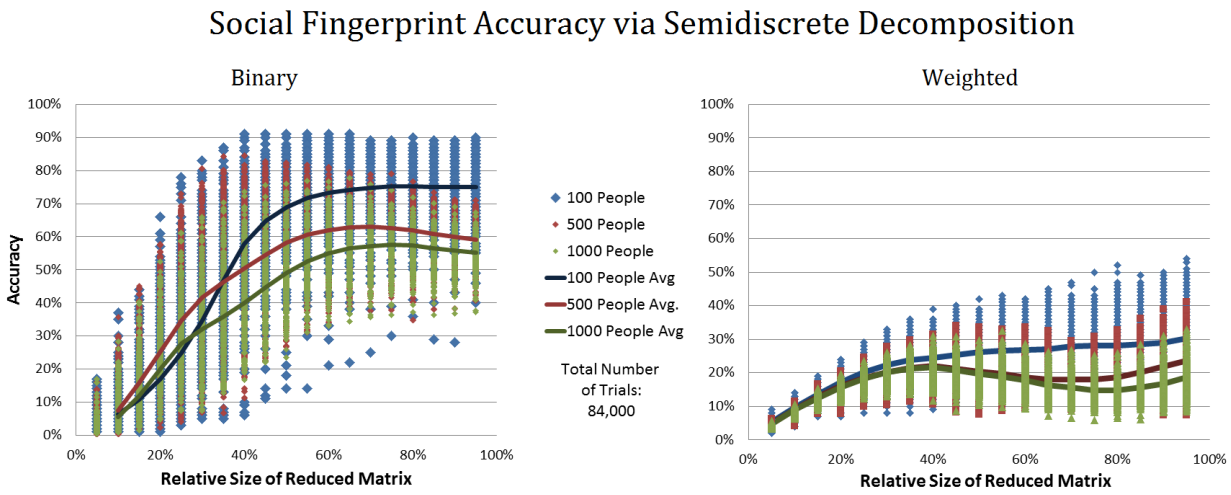


Figure 4.1: The difference in social fingerprinting accuracy after 84,000 trials. The graph on the left illustrates the overall accuracy for binary matrices and query vectors as the relative size of k approaches N. The graph on the right illustrates the overall accuracy for weighted matrices and query vectors as the relative size of k approaches N.

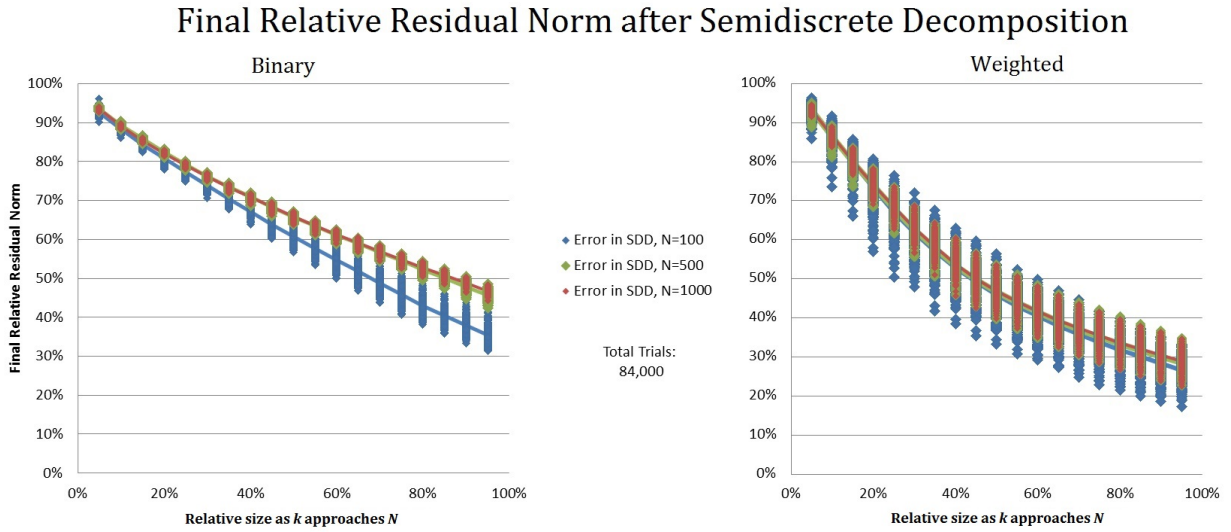


Figure 4.2: The observed and average relative residual norms after 84,000 trials. The graph on the left illustrates the final relative residual norm after the semidiscrete decomposition of a binary community matrix as the relative size of reduction k approaches the original size of the matrix N. The graph on the right illustrates the final relative residual norm after the semidiscrete decomposition of a weighted matrix as the relative size of reduction k approaches the original size of the matrix N

Together, Figures 4.1 and 4.2 confirm that a binary adjacency matrix provides more accurate results for the social fingerprinting pipeline when compared to the weighted models. The highest fingerprinting accuracies of slightly over 91% are achieved via binary matrices for communities of 100 people with relative k values between 40% and 60% of the original matrix size. Additionally, the best average accuracy of approximately 76% is observed via binary models of A when the decomposition approximates the original matrix with a k value of approximately 75% of the original matrix size. For weighted matrices, the best accuracies are achieved when $k \approx 95\%$ of its original size, yet the procedure is only about 54% accurate. For the various models, the optimal reduction size k represents the point at which the condensed matrix contains a balance between noise and separability.

Lastly, as observed in Figure 4.2, the weighted matrices yield a lower overall and average final relative residual norm compared to their counterparts for the binary representations of the same communities. From a mathematical perspective, this result is expected because it is more constraining to fit a binary matrix than a matrix of real numbers. Additional observations are discussed in Section 4.5.

4.4.2 Varying Training Windows

Next, Figure 4.3 displays the varying accuracy for $5,000$ samples of a 100-person community at varying measures of training and testing levels. For the social fingerprinting application, Figure 4.3 demonstrates that the either a 30/70 split or 50/50 split in training and testing data yields the highest overall accuracy. Similar results were observed for both binary and weighted matrices across all sizes of communities. We conjecture that the models ran with 10% training data had lower accuracy because they created an under-fit representation of the community A. On the other hand, we note that the models ran with 90% data in training created an over-fit representation of A. Full results can be found in Appendix B.

Lastly, this research aimed to determine the smallest window of time necessary to create the highest level of accuracy for the social fingerprinting application. Our final set of results applied the optimal settings observed during the first $84,000$ trials runs: a binary matrix

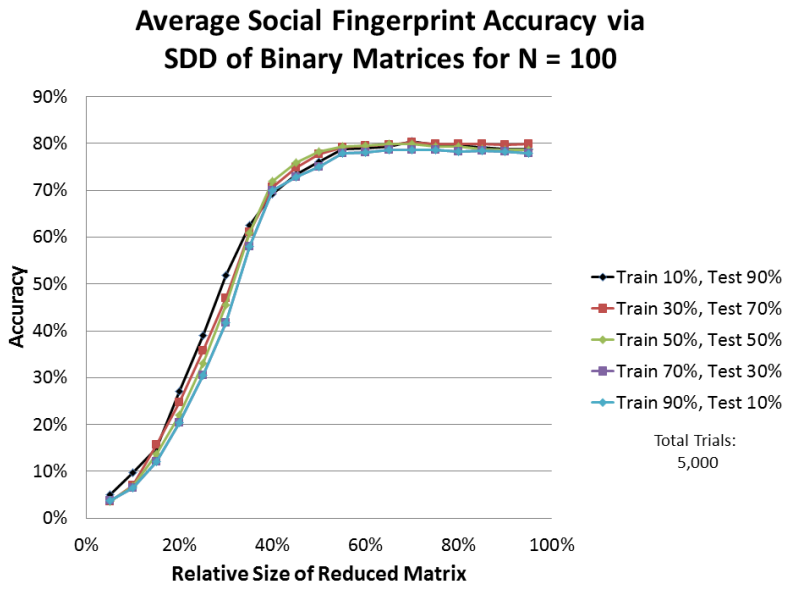


Figure 4.3: The accuracy of the social fingerprinting procedure for 5,000 trials with varying partitions of training and testing data. All trials were run with 100 people using a binary community matrix model.

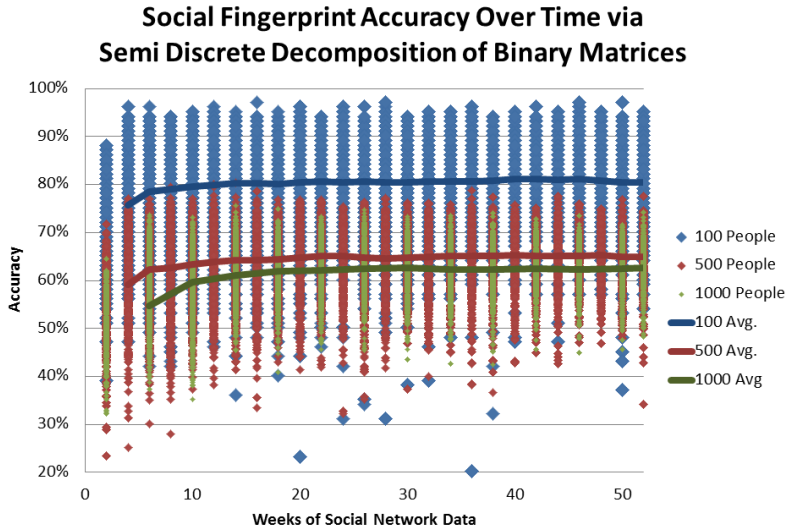


Figure 4.4: The resulting accuracies of the social fingerprinting procedure as the amount of data observations collected for analysis ranged from 2 simulated weeks to 52 simulated weeks. For each community size, $50,000$ trials were run for a total of $150,000$ data samples.

model, a 50/50 split in training and testing data over time and $k = 75\%$ of the original size of the community matrix A. Figure 4.4 shows the resulting accuracies of the social fingerprinting procedure as the amount of data observations collected for analysis ranged from 2 simulated weeks to 52 simulated weeks. For each size of community, $50,000$ trials were run to create a total of $150,000$ models.

4.4.3 A Case Study

The results from Sections 4.4.1 motivated us to explore the underlying discrepancies between the different graph models. To do so, we implemented the social fingerprinting procedure outlined in Algorithm 3 with both a binary and weighted model of the same graph G. We used the optimal training and time windows presented in Section 4.4.2 and set $k = 75\%$ with 50% training data. Further, we set $t = 12$, $\omega = 37\%$, $N = 100$ and relied on the remaining default settings of the SOFA software for all additional parameters.

As expected, the binary model had a higher accuracy over the weighted model and correctly identified 81% of the vertices of A, compared to 27% accuracy with a weighted model of the same graph. Table 4.1 contains the confusion matrix between the binary and weighted models. We see in from the confusion matrix that the binary and weighted models matched on 32 classifications, whereas the binary model correctly identified 61 nodes which the weighted model missed. Further, the weighted model correctly identified 7 nodes that were incorrectly classified by the binary model. A full table of results by node ID is provided in Appendix B.

Table 4.1: The confusion matrix of the case study results. The columns pertain to the results of the weighted model whereas the rows represent the results for the binary model.

	Weighted Model: Incorrect	Weighted Model: Correct
Binary Model: Incorrect	12	7
Binary Model: Correct	61	20

To further explore the difference in accuracy for these two models, we observed the separability between the cosine similarity scores from the binary model against those of

the weighted model. To do so, we created a similarity matrix according to the cosine similarity scores between each query vector and the SDD of A. Then, we constructed a Pearson correlation matrix of the cosine similarity scores to observe the statistical linearity and dependence of the results of this case study Lawrence and Lin (1989). We applied hierarchical clustering with average linkage to both matrices to observe the separability of clusters according to each respective model. Figure 4.5 illustrates the hierarchical clustering of the cosine similarity and Pearson correlation matrices for the binary model of this case study. Figure 4.6 illustrates the hierarchical clustering of the cosine similarity and Pearson correlation matrices for the weighted model of this case study. Figures 4.5 and 4.6 were created using the GAP Software Chen (2002).

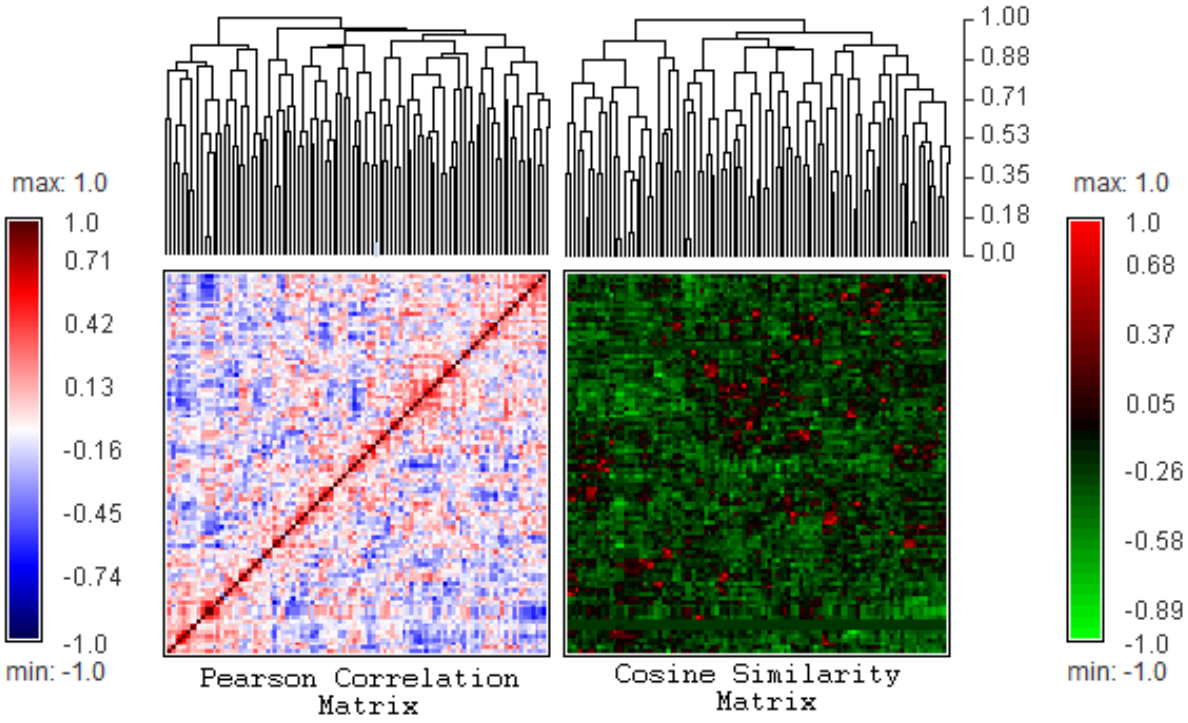


Figure 4.5: The hierarchical clustering of the cosine similarity and Pearson correlation matrices for the binary model of this case study. The red and blue heat map on the left displays the range of scores calculated with the Pearson coefficient of the cosine similarity matrix, which is shown on the right. The hierarchical clustering for each respective set of scores is shown atop each heat map.

We observe much fatter clusters with fewer levels in Figure 4.6 than the clustering observed in Figure 4.5. As such, the weighted model of the SDD social fingerprinting procedure is unable to accurately separate the relevant vertices of G to most accurately match the query vector. That is to say, the SDD of a weighted matrix A produces a less granular clustering of the social communities from G. This results in everyone "looking like" most everyone else and accurate individual identification becomes much more difficult.

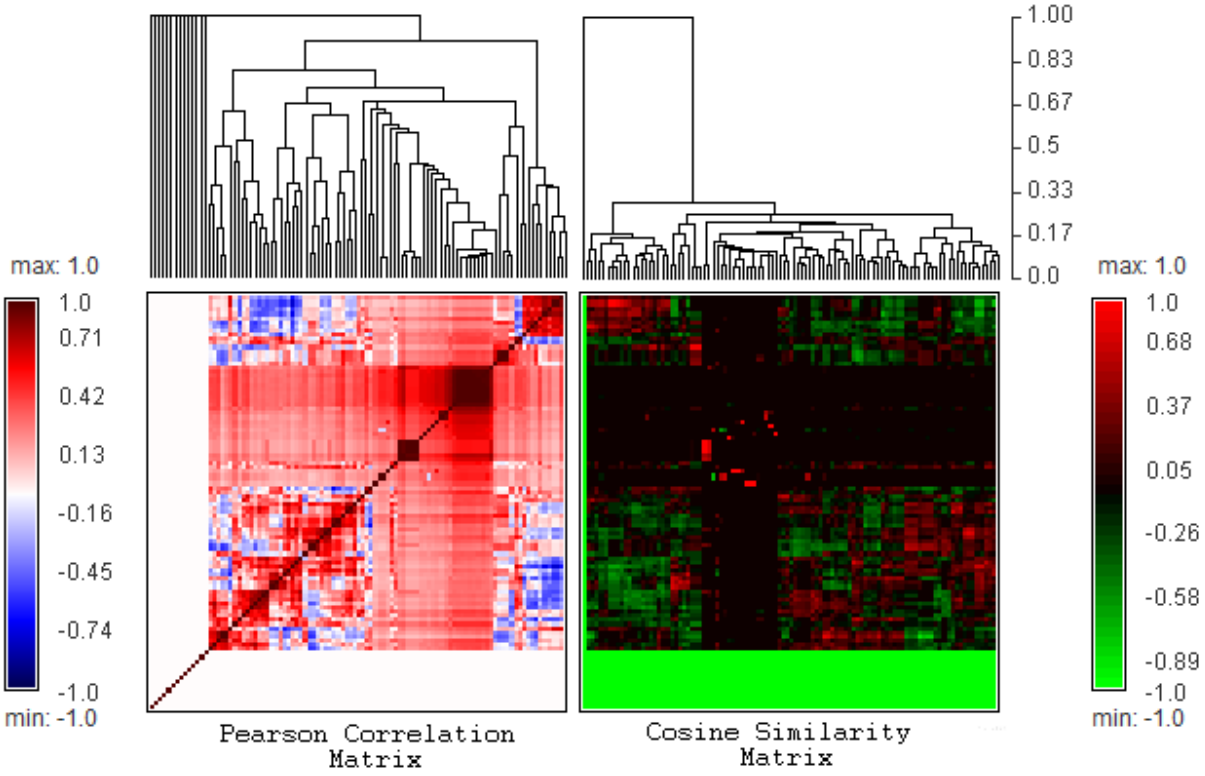


Figure 4.6: The hierarchical clustering of the cosine similarity and Pearson correlation matrices for the weighted model of this case study. The red and blue heat map on the left displays the range of scores calculated with the Pearson coefficient of the cosine similarity matrix, which is shown on the right. The hierarchical clustering for each respective set of scores is shown atop each heat map.

The conjecture that the weighted models produce a less granular view of the social communities is confirmed in the Pearson correlation matrices shown in Figures 4.5 and 4.6. The Pearson correlation values observed from the binary model show a dispersed clustering of communities with high correlation, shown in red and black clusters in Figure 4.5. This

contrasts with the large area of red and black highly correlated communities in Figure 4.6, thereby showing a larger proportion of vertices are highly correlated. The non-symmetric cosine similarities in Figure 4.6 are a result of the large number of rows of zeros after the SDD of A, shown in green.

These results are further confirmed with the *bump-hunting* analogy of SDD described in McConnell and Skillicorn (2001, 2002). McConnell and Skillicorn (2001) demonstrate that the SDD procedure "finds regions of a data set with anomalously large values and extracts them. [Then,] SDD removes bumps with the largest 'volume', a quantity that is based both on the magnitude of values and the number of positions in the data set where they occur." As such, the use of a weighted graph in our application introduces much higher variability amongst the entries of A and introduces a regional topology with large 'bumps' into the N dimensional space represented of A. Therefore, the accuracy of the social fingerprinting pipeline is skewed as the SDD procedure iterates through the regions of A with high quantity. Resultantly, the SDD of A creates large clusters of social communities and provides little separability for individual vector identification.

4.5 Conclusion

The proposed research to collect, create, and identify a network user's social fingerprint is both a novel and difficult research problem. The research presented by Cortes et al. (2003) highlights the complexity of mining social network data due to the attrition of nodes and edges over time. The results and figures presented in this study agree with Cortes et al. (2003) and showcase the difficult components of the data set of study.

This chapter designed an approach to the social fingerprinting problem by creating and testing separability of individual social network users amongst the larger community of observed data. As observed in Section 4.4, the community-based separability approach can be, at best, 76% accurate. These results are achieved with a binary model with a 75% reduction of $N = 100$ and an even split of observed data into training and testing. As such, the results herein support the use of a binary matrix model for this social fingerprinting

application due to the lower accuracy obtained in the weighted matrix model. The lower accuracy of the weighted models is attributed to the "bump hunting" technique of the SDD factorization which produces larger clusters of user identities due to the larger variability of edge weights.

However, it is interesting to note that final relative residual norms in Figure 4.2 are higher for the binary matrix models. That is, we observed that the semidiscrete decomposition more accurately estimates the weighted models of A than the binary models of A. From a mathematical perspective, this result is expected because it is more constraining to fit a binary matrix than a matrix of real numbers. For further investigation, we examined the differences between the estimation of binary and weighted models. We found that the higher error in the weighted models was due to a higher number of rows of zeros in A. Row i of zeros in A indicates that the corresponding user i did not contribute distinguishing social interaction to the community of study. Future work would examine the degree distribution of these corresponding users in addition to surrounding graph properties such as clique membership and local density.

Future work on the approach outlined herein would examine the accuracy and distribution of different similarity measures. Additionally, future work to improve the model outlined in Section 4.3 would challenge the computational limits of this technique by defining a manner to which accurately fingerprint users within communities of 100,000 - 1,000,000 nodes. One approach may be to divide a larger community into smaller sub-communities where Algorithm 3 has proven to accurately identify users. Lastly, another future step for this technique would be to create an real time approximation for the community of study to permit a live identification system. Additional comments on these open problems are revisited in Chapter 6.

To contrast the community-based approach presented herein, the next chapter presents a novel methodology in social fingerprinting to examine how an individual user's data over time creates a social fingerprint for accurate identification. Unlike the collection of an entire community of data, the methodology presented in Chapter 5 solely relies on the node of study and its immediate neighborhood throughout time.

Chapter 5

Social Fingerprinting with Graph Construction and Ranking Functions

"Tell me who your friends are and I'll tell you who you are." -Proverb

5.1 Introduction

This chapter introduces and describes a user-based approach for quantifying a social fingerprint. Given a social network user of interest v_i, we outline a procedure for individual data collection and identification. We define distance functions to quantify the closeness of two users throughout time based on the subgraph which connects them. These functions highlight both the quantity and quality of the shared nodes. Our approach was inspired by the construction of topological neighborhoods to test inference in wireless mesh networks from Xing et al. (2009).

To illustrate our approach, recall the illustration of two anonymous data trails in Figure 1.2. We interpret and visualize the data trails of social network users as network pulses and aim to detect the unique features between any two pulses. One of the most inherent aspects of social networking is the communication with other friends. As such, we are able to break down the network pulse from Figure 1.2 into data trails which describe

each user's relationship on the network of interest. This decomposition of a user's network data into friend specific interactions is visualized in Figure 5.1.

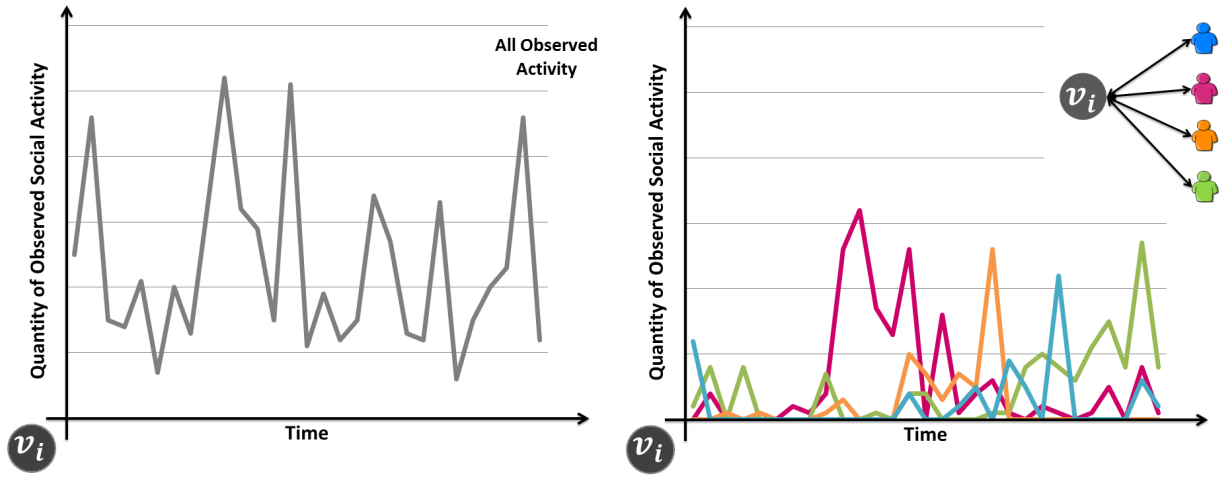


Figure 5.1: The decomposition of a social network user's digital data trail into each unique friendship component.

The decomposition of a user's data trail into friendship components paired with the topology presented in Xing et al. (2009) inspired the approach presented in this chapter. We aim to construct friendship topologies in a manner which is traceable throughout time on a social network. The remaining sections of this chapter are organized as follows. Section 5.2 describes the data generated for this work. Section 5.3 details the graph construction process and the ranking functions implemented across the topologies of study. Section 5.4 contains all of the results and Section 5.5 provides our insights into the results. Our concluding thoughts are in Section 5.6. All numerical results of these simulations are contained in Appendix C.

5.2 Data

We generated multi-edge dynamic graphs with the SOcial Fingerprint Analysis (SOFA) software. Given a range of input parameters, the SOFA software creates a customizable model of a dynamic social network which can then be translated into a variety of social network analysis applications. For a full description of the SOFA software, its capabilities and performance, see Chapter 3.

We rely on the statistics of the AT&T Mobile network published in Cortes et al. (2003) to most accurately design dynamic network models. For this research, we generated dynamic graphs $G = (V, E, T)$ with the SOFA software and varied the input parameters N, ω and w. The number of nodes $N = |V(G)|$ in a model was set to be 1,000, 10,000 or 100,000. We varied the amount attrition of a relationship from $\omega \in \{5, 10, \ldots 95\}$ to simulate all possible relationship intensities. For w, half of the models had binary edge weights whereas we modeled weighted social interactions for the other half with Algorithm 1. With this range of input parameters, we established a constant set of parameters which would enable us to simulate a large range of social networking graphs. We set $t = 6$ for all simulations, created a multi-edge relationship model with $m = 3$, set $\delta = 2$ and $\Delta = 15$. All other parameters were held to their default values, as discussed in Chapter 3.

5.3 Methods

We aim to construct a topology which represents the dynamic nature of a user's social interactions throughout time. Section 5.3.1 steps through the construction process of a graphical social fingerprint. With this construction, we present ten ranking functions to evaluate the importance of a myriad of features present in social data. Then, Section 5.3.6 outlines over 150,000 different tests which are designed to evaluate our approach. Consider Table 5.1 which lists and defines the notation used throughout the remaining sections of this chapter.

5.3.1 Graph Construction

Given the set of input parameters outlined in Section 5.2, we construct a multi-edge dynamic social network using the SOFA software. Then, for each $v_i \in V(G^0)$, we construct two topological representations of the nodes's social neighborhood from different windows of time, as detailed later in Table 5.2. Together, the two topologies create an avenue for quantifying and ranking the importance of various social features for social fingerprint identification.

Table 5.1: A listing of all symbols and terminology used throughout the remaining sections and subsequent chapters of this dissertation.

Symbol	Usage
d	Time point which divides the graph into training and testing
R	The interval of time for training: $[1, d]$
S	The interval of time for testing: $[d+1, t]$
G_i^R	The graph of social interactions for training
G_i^S	The graph of social interactions for testing
$\{P_i\}$	The set of candidate prints associated with v_i; $\{P_i\} \subseteq V(G_i^S)$
p_A	A candidate print in P_i associated with vertex v_A
$E(G^{[a,b]})$	The set of all edges observed during the time window $[a, b]$
$\langle e(v_i, v_j)^t \rangle$	A feature vector of edge weights during time t between v_i and v_j

The first social neighborhood is labeled G_i^R and is visualized on the left of Figure 5.2. In G_i^R, we aggregate all observed social interactions of v_i during the time window $R = [1, d]$ into one graph. Equation 5.1 details the creation of the training graph:

$$G_i^R = N(v_i)^R = \{v_j \in V(G) | e(i, j) \in E(G^{[1,d]})\} \tag{5.1}$$

The second social neighborhood is labeled G_i^S and is illustrated on the right of Figure 5.2. In G_i^S, we aggregate all observed social interactions of $v_j \in G_i^R$ during the time window $S = [d+1, t]$ into one graph. Equation 5.2 details the creation of the testing graph:

$$G_i^S = N(G_i^R)^{[d+1,t]} = \{v_m \in V(G) | e(j, m) \in E(G^{[d+1,t]})\} \tag{5.2}$$

This construction enables us to view all of the vertices $v_m \in G_i^S$ as a set of candidate prints $\{P_i\}$ for vertex v_i. That is, this approach exploits the observed social interactions throughout time as a feature space for user identification. This idea hinges on the hypothesis that there exists a core set of friends with which a social network user interacts over time. If this is true, then we expect that some $v_m \in \{P_i\} = v_i$.

With this hypothesis, we are able to filter out some potential prints after construction. Specifically, a vertex $v_j \in G^R$ is an immediate social connection of v_i and, as such, cannot

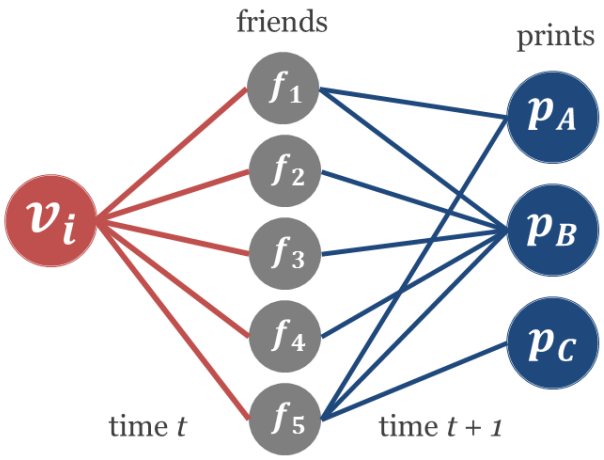


Figure 5.2: An example of the topology of study for user v_i. The information in red on the left and the corresponding friends in gray represents graph G_i^R. The data on the right in blue along with the initiating friends in gray represent graph G_i^S and the candidate social fingerprints are $\{P_i\} = \{p_A, p_B, p_c\}$. The goal of this research is to implement a qualitative ranking algorithm which sorts the list of prints $\{P_i\}$. The algorithm is correct if the top ranked print has the same identifier as the subscriber v_i.

also be a candidate print for v_i. That is, we eliminate all vertices from the testing graph which are also in the training graph: $\{P_i\} = G_i^S - G_i^R$. Figure 5.2 illustrates the construction process. An example of a training graph is shown on the left in red in Figure 5.2 and an example of a testing graph is shown on the right in blue. Any friend in gray from Figure 5.2 is not eligible to be a candidate print in blue.

The goals of this construction are two fold. First, we aim to traverse a social network users's friends throughout time to re-identify the original identity of interest. We assume that a social network user has at least one consistent social connection throughout an arbitrary window of time. That is, we assume the original user of study will show up in the data as a friend of his or her friends as is visualized in Figure 5.2. The second goal of this data construction is to rank all vertices $\{P_i\}$ in a manner which percolates v_i to the top of the list. The algorithm is correct if the top ranked print has the same identifier as v_i.

The following sections detail the various ranking algorithms implemented on the test graphs to rank the vertices $p_A \in \{P_i\}$. These methods examine the quantifiable differences at both the node and relationship levels between v_i and each p_A. Additionally, the

methodologies explored subgraph comparisons and ensemble style ranking algorithms. Section 5.3.2 examines the various ways to calculate the distance between the edges represented in G^R and G^S. Section 5.3.3 details the node-based ranking algorithms.Finally, Section 5.3.4 explores the use of ensemble scoring methods inspired by popular sports ranking algorithms Langville and Meyer (2012).

5.3.2 Edge-based Rankings

Recalling Figure 5.2, the goal of this research is to implement a qualitative ranking algorithm which sorts the list of prints in $\{P_i\}$. The algorithm is correct if the top ranked candidate has the same identifier as v_i. We implement five ranking functions to examine the qualitative difference between the relationship vectors of v_i and each print $p_A \in \{P_i\}$. That is, we seek to find the print with the smallest change in quantity of behavior from G^R to G^S. Therefore, the edge-based ranking algorithms sort the candidates according to the smallest measured change between the edge vectors $\langle e(v_i, v_j)\rangle$ and $\langle e(v_j, p_A)\rangle$ where $v_j \in G^R$ and $p_A \in \{P_i\}$. We rank the print p_A with the smallest change in behavior as the most likely candidate for user v_i. All approaches are normalized by the number of friends shared between v_i and p_A to account for the inherent unfairness of a larger set of friends. Equation 5.3 defines the normalization process:

$$L = N(v_i) \bigcap N(p_A), \tag{5.3}$$

where all edge-based rankings are normalized by $|L|$.

First, we calculate the exact difference in edge weight observed between training and testing. This approach provides a base line for how different the dynamic relationships are over time by measuring the change between the vectors $\langle e(v_i, v_j)\rangle$ and $\langle e(v_j, p_A)\rangle$. Recall from Table 5.1 that $\langle e(v_i, v_j)^t\rangle = \langle b_1^t, b_2^t, \ldots, b_m^t\rangle$ and b_i^t represents the aggregated quantity of each edge type b_i during time window t. Equation 5.4 calculates the Absolute Difference

Score for (p_A):

$$AbsoluteDifference(p_A) = \sum_{v_j \in L} |\left\langle e(v_i, v_j)^R \right\rangle - \left\langle e(v_j, p_A)^S \right\rangle|, \tag{5.4}$$

where the final score is calculated as

$$AbsoluteDifferenceScore(p_A) = \frac{AbsoluteDifference(p_A)}{|L|}. \tag{5.5}$$

The print p_A with the overall lowest difference in behavior is ranked highest and the unique identifier for p_A is checked against that of v_i for correctness. This approach is correct when the identifiers match, otherwise it is incorrect. Note that Equation 5.4 accounts for the unfairness of this approach for prints with high degree by normalizing each score by the total number of common friends.

Next, we calculate the percent difference in edge weight observed between training and testing. Equation 5.6 calculates the Percent Difference Score for (p_A):

$$PercentDifference(p_A) = \sum_{v_j \in L} \frac{|Difference|}{Average}, \tag{5.6}$$

where

$$Difference = \left\langle e(v_i, v_j)^R \right\rangle - \left\langle e(v_j, p_A)^S \right\rangle, \tag{5.7}$$

$$Average = \frac{\left\langle e(v_i, v_j)^R \right\rangle + \left\langle e(v_j, p_A)^S \right\rangle}{2}, \tag{5.8}$$

and the final score is calculated as

$$PercentDifferenceScore(p_A) = \frac{PercentDifference(p_A)}{|L|}. \tag{5.9}$$

The print p_A with the overall lowest difference in behavior is ranked highest and the unique identifier for p_A is checked against that of v_i for correctness. This approach is correct when the identifiers match, otherwise it is incorrect. Note that, Equation 5.6 accounts for the

unfairness of this approach for prints with high degree by normalizing each print's final score by the total number of common friends.

A final three quantitative approaches to comparing the closeness of v_i with each print p_A uses the Euclidean distance between the respective relationship vectors. In theory, a user follows a core pattern of activity over time for its respective friends. As a result, the use of Euclidean distance aims to detect the minimum change between the relationships of v_i and those of each print p_A. Recall the definition for Euclidean Distance:

$$EuclideanDistance(\vec{q}, \vec{p}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \tag{5.10}$$

The third ranking function accumulates the overall Euclidean Distance between the relationships in coming with v_i and p_A. The Euclidean Difference Score for (p_A) is given by:

$$EuclideanDistanceScore(p_A) = \frac{\sum_{v_j \in L} EuclideanDistance(\langle e(v_i, v_j) \rangle, \langle e(v_j, p_A) \rangle)}{|L|}. \tag{5.11}$$

The print p_A with the overall lowest difference in behavior is ranked highest and the unique identifier for p_A is checked against that of v_i for correctness. This approach is correct when the identifiers match, otherwise it is incorrect. Note that Equation 5.11 accounts for the unfairness of this approach for prints with high degree by normalizing each print's final score by the total number of common friends.

Lastly, two variations on the Euclidean Distance score were calculated: a threshold based measure and an exponential sum. The Euclidean Distance Threshold Score is calculated as:

$$EuclideanThresholdScore(p_A) = \frac{\sum_{v \in L} \begin{cases} 1, & if\ \gamma \leq T \\ 0, & otherwise \end{cases}}{|L|}, \tag{5.12}$$

where

$$\gamma = EuclideanDistance(\langle e(v_i, v_j)\rangle, \langle e(v_j, p_A)\rangle), \tag{5.13}$$

and T is a threshold. A perfect score for p_A would be 1, thereby indicating that each $v \in L$ had similar edge weight in G^R as in G^S. Therefore, a print with an Euclidean Distance Threshold Score closer to 1 indicates very similar social activity during the two time windows of study. As such, the print p_A with the overall largest accumulation is ranked highest and the unique identifier for p_A is checked against that of v_i for correctness. This approach is correct when the identifiers match, otherwise it is incorrect. Note that Equation 5.12 accounts for the unfairness of this approach for prints with high degree by normalizing by the total number of common friends.

The final way we examine the quantitative value of Euclidean distance between the training and testing graphs is by accumulating an overall score before normalizing by the number of friends in common:

$$EuclideanSummationScore(p_A) = \frac{\sum_{v \in L} e^{-\gamma}}{|L|}. \tag{5.14}$$

A perfect score for p_A would be $e^0 = 1$, thereby indicating that each $v \in L$ had the exact same edge weight in G^R as in G^S. Therefore, a print with an Euclidean Distance Summation Score closer to 1 indicates very similar social activity during the two time windows of study. As such, the print p_A with the overall largest accumulation is ranked highest and the unique identifier for p_A is checked against that of v_i for correctness. This approach is correct when the identifiers match, otherwise it is incorrect. Note that Equation 5.12 accounts for the unfairness of this approach for prints with high degree by normalizing by the total number of common friends.

The only difference between the final two metrics is that Equation 5.12 permits a range of closeness to affect the candidate's score whereas every single distance score in Equation 5.14 is accumulated into a global score. Intuitively, all five of these metrics create measurable similarity of the edge weights between the print p_A, the user v_i and each respective common

friend. It is necessary to normalize each score by the number of friends in common to prevent a candidate print from being penalized for having more friends in common with v_i.

5.3.3 Node-based Rankings

We implement three variations of node-based ranking functions to test the importance of friendships over time as an integral feature for social fingerprinting. Recalling Figure 5.2, the goal of this research is to implement a qualitative ranking algorithm which sorts the list of prints in $\{P_i\}$. The algorithm is correct if the top ranked candidate has the same identifier as v_i. The simplest approach explored in this work ranks the prints according to the number of friends they have in common with v_i. That is,

$$IntersectionScore(p_A) = |N(v_i) \bigcap N(p_A)|, \tag{5.15}$$

where prints with more common friends are ranked higher than a candidate with fewer friends. Then, the identifier of the top ranked print is checked against that of v_i for correctness. This basic ranking hinges on the natural assumption that people interact with a core set of the same people over time on a social network. Figure 5.3 illustrates an example of Equation 5.15.

The next two node-based ranking functions present different ways to quantify the difference between two subgraphs. An intuitive way to compare the subgraph shared by v_i and each vertex in $\{P_i\}$ is to examine the subgraph matching cost. First, we implement a variation of Hamming Distance to measure the observed difference between the multi-edge relationships in G^R and G^S. As demonstrated in Figure 5.4, a candidate print p_A is awarded for each matching behavior between graphs G^R and G^S, for the observed activity only. Each relationship in Figure 5.4 had three possible edge types. As such, p_A is ranked over p_B for correctly matching more behaviors than p_B. For validation, the unique identifier for p_A is checked against that of v_i for correctness.

The final node-based distance calculates the minimum number of alterations required to adjust one subgraph to another. Therefore, a third series of analytics are collected which take

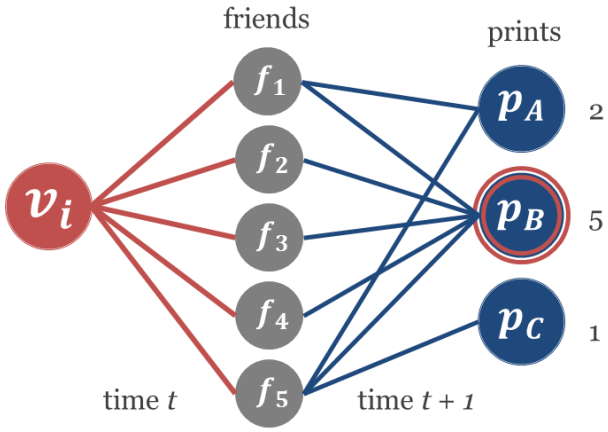


Figure 5.3: An example of the Intersection Score. Prints with more common friends are ranked higher than a candidate with fewer friends. This ranking hinges on the natural assumption that people interact with a core set of the same people over time. Print p_B has a total of five friends in common with v_i and is ranked highest of all candidate prints. In the case that the identifier of p_B matches that of v_i, we label this test case as correct. Otherwise, this test case is labeled as incorrect.

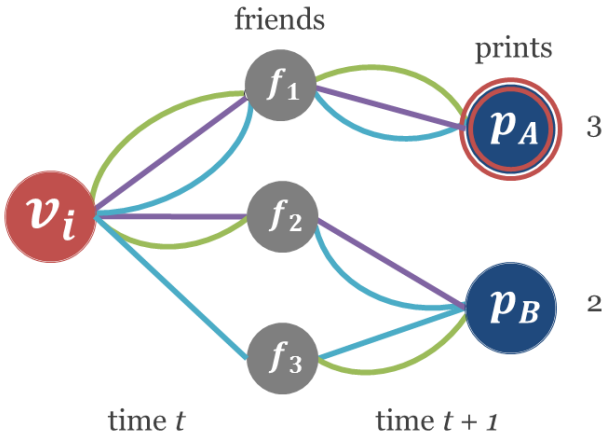


Figure 5.4: An example of a graph based approach to observed Hamming Distance. Prints with more matching behaviors are ranked higher than a candidate with fewer matching behaviors. This basic ranking hinges on the natural assumption that people interact with the same people in a similar manner over time. Print p_A has a total of three matching behaviors in common with v_i and is ranked highest. In the case that the identifier of p_A matches that of v_i, we label this test case as correct. Otherwise, this test case is labeled as incorrect.

into account the absences of social interaction. Consider the example shown in Figure 5.5. In this figure, we observe that v_i and f_3 did not engage in a one of the potential edge types in graph G^R, and f_3 and p_B also did not engage in that particular social activity in graph G^S. As such, p_B is awarded for matching the absence of this particular social interaction. Accordingly, the third round of analytics performed on the simulated data awards a print for matching v_i exact behavior. The print with the most points is listed as the top ranked print and the vertex's identifier is checked against that of v_i for correctness. The tie-breaking function noted in Figure 5.5 is detailed in Section 5.3.5.

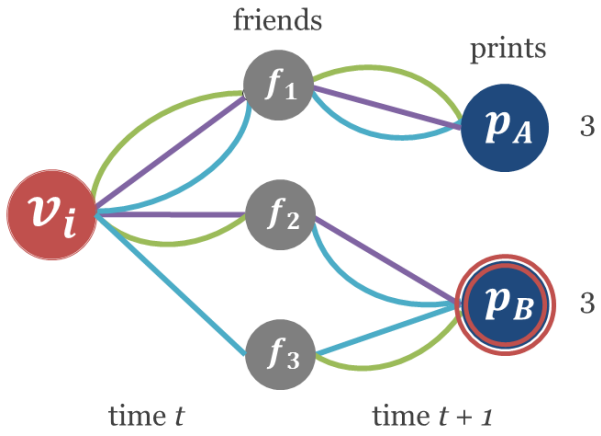


Figure 5.5: An example of the Graph Matching Score. A candidate print p_A that requires fewer alterations to match the subgraph in G^R is ranked higher than a print requiring more alterations. This basic ranking hinges on the natural assumption that people interact with the same people in a similar manner over time. We observe that v_i and f_3 did not engage in a one of the potential edge types in graph G^R, and f_3 and p_B also did not engage in that particular social activity in graph G^S. As such, p_B is awarded for matching the absence of this particular social interaction. Print p_B has a total of three matching behaviors in common with v_i and is ranked highest after implementing the tie-breaking function from Section 5.3.5. In the case that the identifier of p_B matches that of v_i, we label this test case as correct. Otherwise, this test case is labeled as incorrect.

5.3.4 Sports Inspired Ensemble Voting Measures

After implementing a variety of ranking methodologies described in the previous section, we naturally arrive at the need to explore voting methods. After studying some of the sports

ranking systems in Langville and Meyer (2012) and Govan et al. (2009), the author chose to implement two basic voting strategies. The first strategy, herein after referred to as the "BCS", scores each candidate by the total number of first place votes received for each of the aforementioned 8 ranking algorithms in Sections 5.3.2 and 5.3.3. A second strategy, referred to as the"Olympic Score", awards each candidate a predetermined number of points according to its ranking for each of the aforementioned methodologies. A first place ranking is awarded 20 points, second place is awarded 18, third gets 17, fourth receives 16 and so on.

5.3.5 Tiebreakers

In the case of a tie, as occurs in Figure 5.5, two levels of tiebreakers are implemented. If the ranking function in question produces matching scores, the first tiebreaker defaults to Equation 5.15. If a tie is again achieved, we implement the Most Influential Score:

$$Influence(p_A) = \sum_{v_j \in N(p_A)} deg(v_j) \tag{5.16}$$

Equation 5.16 allows us to prioritize linking friendships of low degree by rewarding the print with a lower score. As such, prints with lower influence scores are ranked higher, indicating they are a node with more significant social ties.

5.3.6 Test Design

In this research, we generated multi-edge dynamic graphs with the SOFA software according to the input parameters outlined in Section 5.2. For all simulations, we set $t = 6$. With each graph, we created five different training windows for testing purposes; Table 5.2 details the five different time windows.

Further, to simulate all possible scenarios, we varied the amount attrition (ω) of a relationship in the dynamic graphs from $\omega in\{5, 10, \ldots 95\}$ and $N \in \{1000, 10000, 100000\}$. For each value of ω and N, we created 100 instances of binary graphs and 100 instances

Table 5.2: The training and testing windows used in this research

d	G^R	G^S
1	G^1	$G^2 \cup G^3 \cup \ldots \cup G^6$
2	$G^1 \cup G^2$	$G^3 \cup G^4 \cup \ldots \cup G^6$
3	$G^1 \cup G^2 \cup G^3$	$G^4 \cup G^5 \cup G^6$
4	$G^1 \cup G^2 \cup \ldots \cup G^4$	$G^5 \cup G^6$
5	$G^1 \cup G^2 \cup \ldots \cup G^5$	G^6

of weighted graphs. Further, each graph was analyzed using the five different scenarios described in Table 5.2, yielding a total of 57,000 example graphs using the SOFA software. With each graph G, we created G^R and G^S for every vertex $v_i \in V(G)$ and ranked the set of candidate prints $\{P_i\}$ with each of the ten ranking functions described in Section 5.3. The correctness of each instance of the ranking algorithm was recorded. The results of our test runs are presented in Section 5.4. Appendix C contains complete tables of the results highlighted in Section 5.4. We discuss all results in Section 5.5.

5.4 Results

Before breaking down the performance of our models, let us first examine the validity the construction process designed in Section 5.3.1. Figure 5.6 shows the average percentage of cases when v_i is an print of $\{P_i\}$ as ω ranges from $[5, 95]$. The accuracy of this construction process is shown for all five different training windows from Table 5.2. In Figure 5.6, the results shown for "10% Train" correlate to $d = 1$ from Table 5.2, "30% Train" correlates to $d = 2$, "50% Train" correlate to $d = 3$, and so on. The results were averaged over all models for any size N. These results indicate that for values of $\omega \leq 50$, we are 99.9% likely to re-identify v_i in $\{P_i\}$ when 50% of the data is collected for training.

5.4.1 Training Windows

Figure 5.6 confirms that our construction hypotheses accurately trace v_i over time. Next, let us investigate the affect the different training windows had on the overall average accuracy

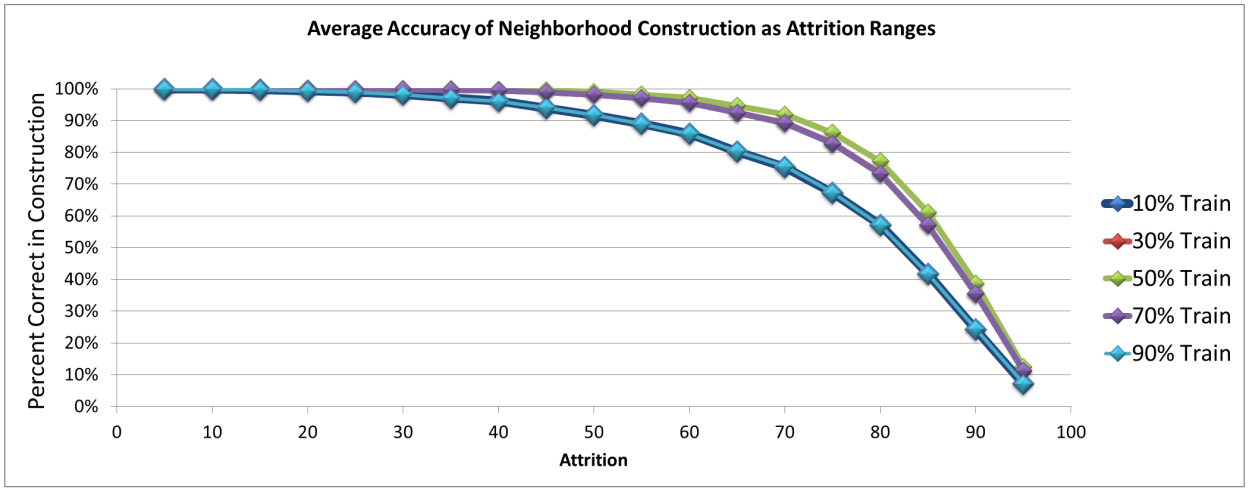


Figure 5.6: The total percentage of cases when $v_i \in \{P_i\}$ as ω ranges from $[5, 95]$ for the five different training windows from Table 5.2.

of the model. Figure 5.7 depicts the average results obtained across all ranking functions whereas Figure 5.8 contains only the average results for when $N = 100,000$. Similar results were observed when $N = 1,000$ and $N = 10,000$ and are displayed in Appendix C.

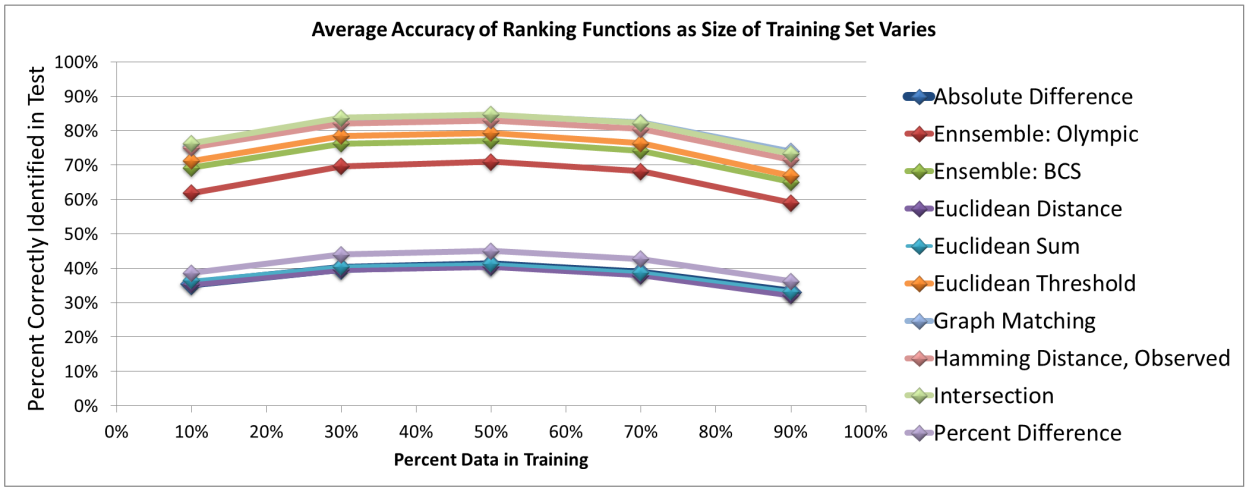


Figure 5.7: The average accuracy across all ranking functions and all values of attrition as the training window ranges from 10% to 90%.

Figures 5.7 and 5.8 show that a user's social fingerprint is more accurately identified with equal amounts of data collected for training and testing. As seen in the results for 10% and

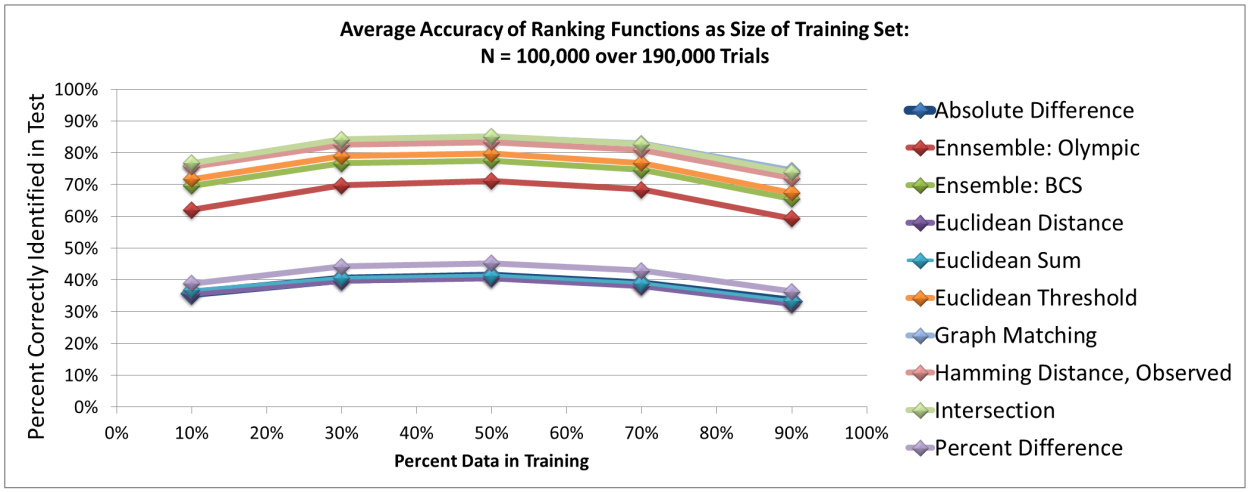


Figure 5.8: The average accuracy across all ranking functions and all values of attrition as the training window ranges for $N = 100,000$.

90%, the overall average accuracy of our models slightly decreases with an imbalance of data aggregated into the training and testing graphs.

5.4.2 Attrition Results by Function

Next, we consider the average accuracy of each ranking function across all models. Figure 5.9 showcases the average results of each ranking function as ω ranges from 5% to 95% for all values of N and d. Figure 5.10 displays the average results of each ranking function as attrition ranges from $[5, 95]$ for only those models with $N = 100,000$. Similar results were observed when $N = 1,000$ and $N = 10,000$ and are displayed in Appendix C. Further, all averages are provided tables in Appendix C.

With attrition rates lower than 65%, the tables in Appendix C confirm that the Graph Matching ranking function slightly outperforms all ranking functions implemented in Figure 5.9. However, the Intersection ranking function is the best model for social networks with attrition rates higher than 65%. Overall, the edge-based ranking functions performed the worst.

Given that the best results are obtained with $d = 3$ and the Graph Matching ranking function, we observe the difference between the binary and weighted graph simulations with

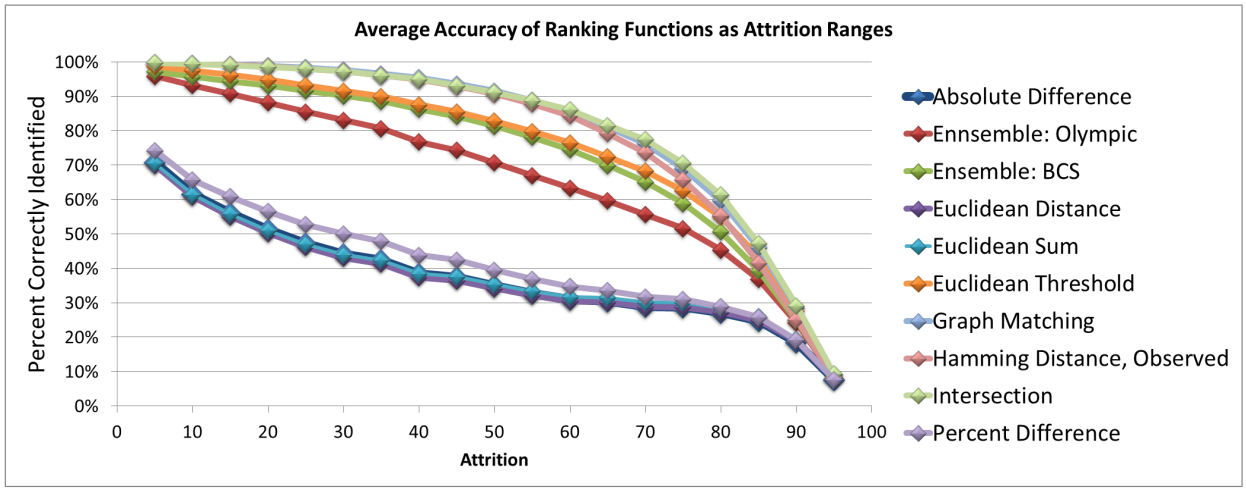


Figure 5.9: The accuracy of each ranking function as the attrition rate ranges from 5 to 95. This figure shows the average results for all values of N for 190,000 tests.

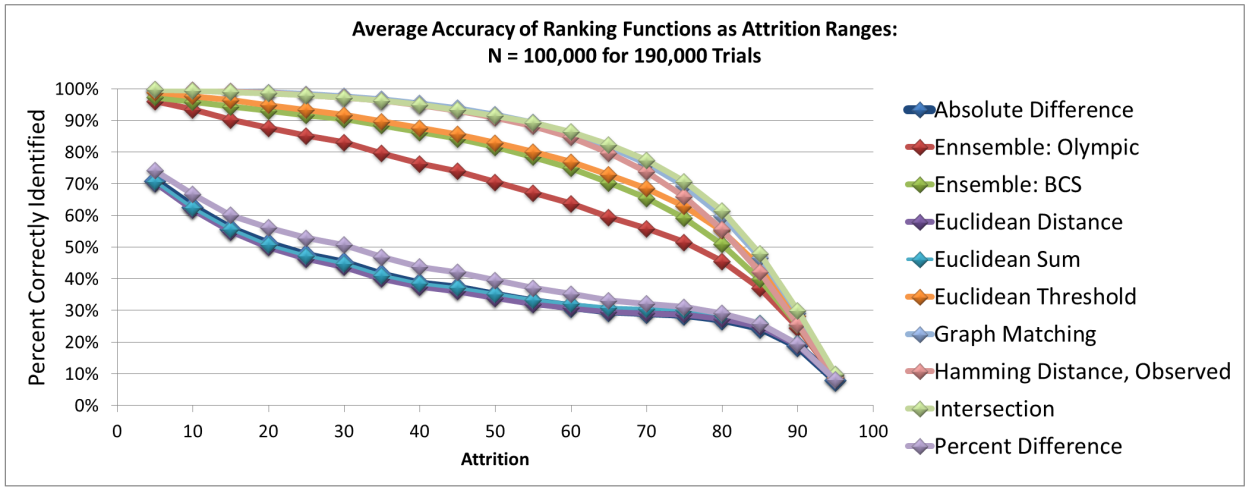


Figure 5.10: The accuracy of each ranking function as the attrition rate ranges from 5 to 95. This figure shows the average results for $N = 100,000$ for 190,000 tests.

those parameters. Figure 5.11 illustrates the accuracy of 570 binary simulations and 570 weighted simulations as the attrition rate ω ranges from $[5, 95]$. In Figure 5.11, the vertical dispersion of the data depicts the range of accuracy observed across the simulations with the respective lines indicated the average performance of each model.

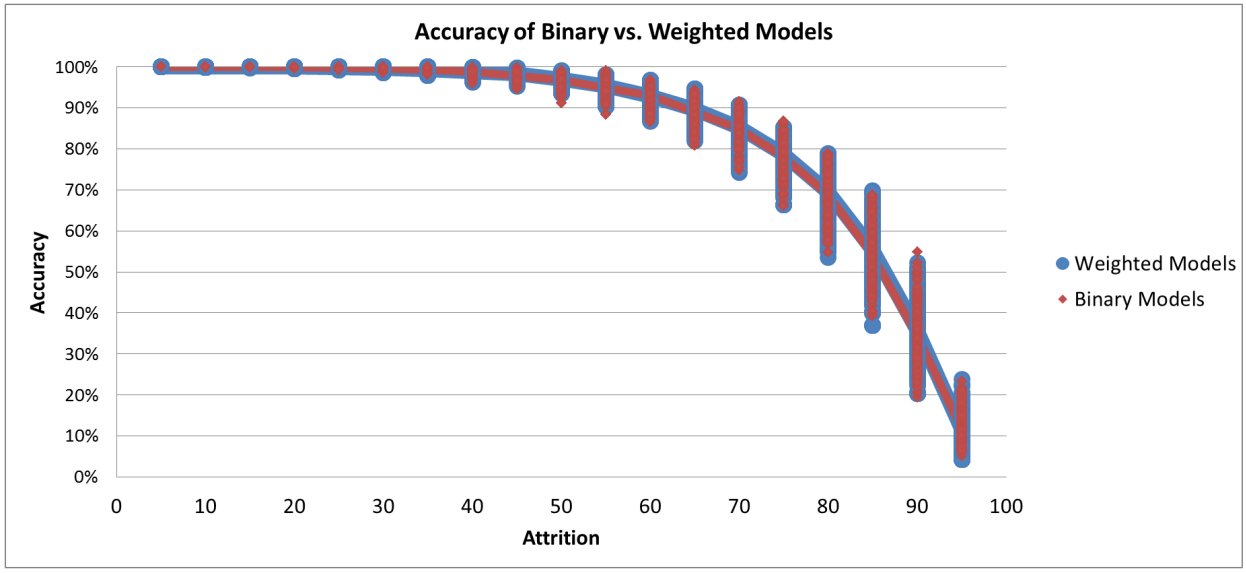


Figure 5.11: The accuracy of 570 binary simulations and 570 weighted simulations as the attrition rate ω ranges from $[5, 95]$. The weighted models were plotted with blue data points and the binary models were plotted with red data points. The vertical dispersion of the data depicts the range of accuracy observed across the simulations. The blue line shows the average accuracy of the weighted models. The red line shows the average accuracy of the binary models.

5.4.3 Observed Friendships

Lastly, we observe the performance of each ranking function as the number of friends of v_i ranges from 2 to 15 in G_i^R. As seen in Figure 5.12, we observe that the node-based ranking functions accurately identify 99.9% of our test cases when $\omega = 35\%$ and $d = 3$. The results for each size of model were similar and are contained in Appendix C.

Lastly, we consider the accuracy of this research for the best set of parameters. We trained on 50% of the data and observed the performance of the Graph Matching algorithm

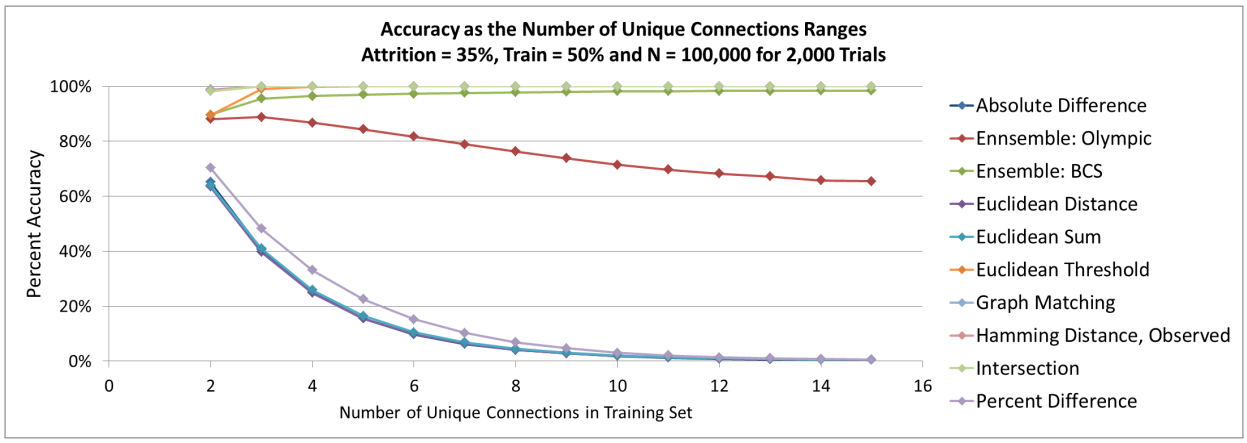


Figure 5.12: The performance of each ranking function as the number of friends of v_i ranges from 2 to 15 in G_i^R for $N = 100,000$, $\omega = 35$, and $d = 3$.

on 4,000 models with $N = 100,000$. We collected the average accuracy of these models as the attrition rate ranged from $[5, 95]$. Figure 5.13 showcases the results of this simulation.

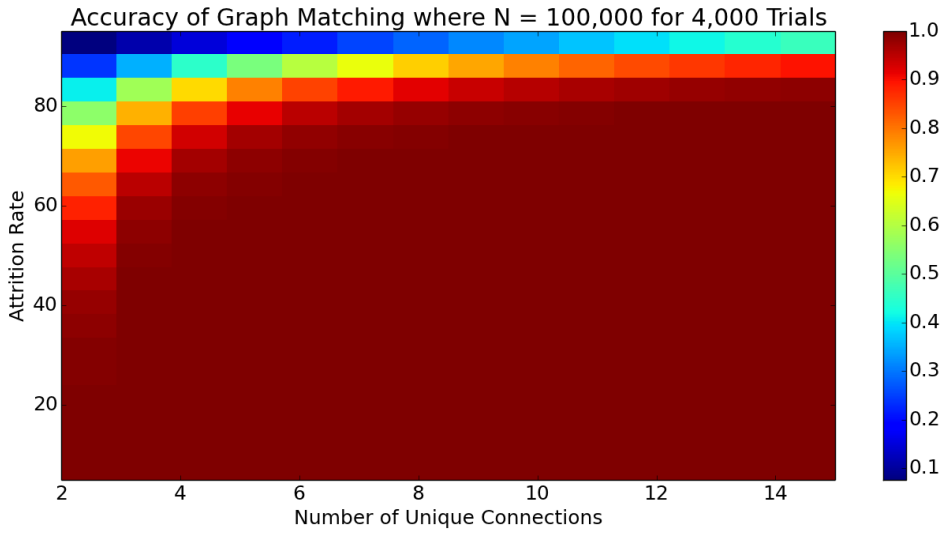


Figure 5.13: The performance of the Graph Matching algorithm on 4,000 models with $N = 100,000$ as the number of friends of v_i ranges from [2,15].

5.5 Discussion

Figure 5.6 contains one of the most promising features of this research. This result confirms the graph construction procedure described in Section 5.3.1 creates a traceable topology throughout time which is an accurate identifier of social network users. Consider the monthly attrition rate of 37% for the AT&T mobile subscribers published in Cortes et al. (2003). The results in Figure 5.6 suggest that over 6 months of time, any mobile subscriber on the AT&T network could be uniquely identified with the graph construction procedure outlined in this research. We hypothesize that a similar result would hold true for any social network with an observed attrition rate of less than 50% over six arbitrary time steps.

5.5.1 Training Windows

Figure 5.7 demonstrates a fundamental principal for future implementations of social fingerprinting. The convexity of the curves indicate that the most accurate social fingerprints are traced with balanced time windows during the collection process. That is, the length of time spent on data aggregation for training is best matched with prints collected over the same quantity of time for testing. The decay in accuracy for models with 10% or 90% of the data collected for training confirm the imbalance of information creates inaccurate depictions of the user, thus leading to incorrect identifications during the testing procedures.

5.5.2 Attrition Results by Function

Figures 5.9 and 5.10 display the resilience of the node-based ranking functions. As social interactions naturally dissipate over time, the comparison of node-based statistics vastly outperforms those metrics emphasizing differences in edge weights. This is confirmed in the tables of Appendix C where we find that the Graph Matching, Intersection and Hamming Distance functions had an average accuracy of 96% or better when $\omega \leq 50$ and $d = 3$ for any size model. We note that other distance measures, such as Mahalanobis distance

De Maesschalck et al. (2000), may be more appropriate for edge-based models if the variances in edge weight are drastically different.

There is one edge-based function which outperforms both ensemble voting methods. Shown in Figure 5.9 in orange, we see that the Euclidean Threshold function described in Equation 5.12 vastly outperforms the other edge comparison functions. The performance is attributed to the hybrid nature of Equation 5.12. This function essentially performs as the intersection function with a preliminary test to measure the difference between the relationship observed during training and testing. If that difference is low enough, the friendship is counted, else it is disregarded. As such, this allows this approach to Euclidean distance to use the edge weight information to perform, essentially, a node-based ranking function.

Knowing that the best results are achieved with $d = 3$ and the Graph Matching ranking, it is interesting to observe the results contained in Figure 5.11. We observe that the binary and weighted models produced by the SOFA software have little to no difference in accuracy for the graph construction approach to social fingerprinting. While this figure showcases the results of the best performing function, similar averages were reported across all ten ranking functions and can be found in Appendix C.

5.5.3 Observed Friendships

The results presented in Section 5.4.3 demonstrate the validity of the design in our graph construction from Figure 5.2. Most supportive is Figure 5.13 which shows that any social network user is traceable with the right amount of information. Consider the range of accuracy in Figure 5.13 for 8 friends. This demonstrates that a social network user with 8 friends or more can be uniquely identified across 6 time steps when the attrition of any relationship reaches over 75%. Further, those social network users with only 2 friends can be accurately traced over 6 time steps when the attrition of a relationship is less than 40%.

5.6 Concluding Remarks

We outlined a procedure for examining the feature space observed in social network interactions as potential identifiers for new approaches in fingerprinting. We hypothesized that a social network user's friendships would provide an accurate base for identifying the same user during another period of time. Using the simulations created by the SOFA software, we demonstrated that this approach provides a valid technique for fingerprinting social network users over time with varying levels of relationship attrition. In over 500,000 simulations, we examined the accuracy of ten different ranking functions across a gamut of input parameters to model the dynamic nature of the AT&T mobile graphs described in Cortes et al. (2003).

With these simulations, we are able to conclude that a social network user's friendships over time provide enough data to accurately identify the user of study. Specifically, the results in Figure 5.13 demonstrate that a social network user can be accurately printed with as few as two connections during a time span on interest. Further, this research demonstrates that the accuracy of the social fingerprint procedure constructed in Section 5.3.1 is more accurate for social network users with larger sets of friends.

The results simulated herein are an improvement over current published methodologies. In March of 2013, de Montjoye et al. (2013) calculated the uniqueness of 95% of mobile subscribers over a 15 month time period with as few as 4 unique mobile tower locations. We improve upon these findings in two manners. First, we present a procedure for social fingerprinting that identifies users from anonymous data trails on a network of study. Second, we present simulated results which demonstrate that two or more observed friendships over six time periods accurately re-identify network users.

Next, we conclude and compare the results presented in this chapter with those obtained in the community-based approach from Chapter 4. Further, we present open work in social fingerprinting as it relates to all methodologies created, tested and analyzed in this dissertation.

Chapter 6

Conclusion

We introduced a novel and difficult problem in social network analytics: the social fingerprint. With the advent of social networks and the modern data explosion, we theorized and demonstrated that the data generated by a social network user leaves a viable trail of data which can serve as a unique identifier, just like the human fingerprint. To frame the advent of the social fingerprint within current research, we discussed current techniques in large-scale social analytics to detect fraud, predict market share and provide accurate social analytics. Further, we established the difficulty of the social fingerprint as a new approach for digital forensics by exploring the dynamic and challenging nature of mining expansive social networks.

Further, since the most sophisticated techniques and datasets for social network analysis are unattainable in most academic settings, the SOcial Fingerprint Analysis (SOFA) software delivers a scalable vehicle for future research in large-scale social analytics and social fingerprinting. With data generated by the SOFA software, we demonstrated that techniques in matrix factorizations and information retrieval can accurately identify 76% of social network users by quantifying the separability of the data trails observed within small communities. Most notably, we demonstrated that, depending on the observed attrition levels of a social network, we can accurately identify 99.9% of social network users when we observe the social topology of their immediate connections over time. In the case of the

AT&T network which has an attrition rate of 37%, the techniques presented in Chapter 5 theoretically could accurately print 98.8% of the mobile network users.

While we defined, explored and analyzed techniques in social fingerprinting, we also uncovered a myriad of open problems for future research. Chapter 3 outlines the initial implementation of the SOcial Fingerprint Analysis software and provides a few suggestions for its next iteration. We note three categories of open research for the next iteration of software: enhanced modeling, additional features and improvements on the max flow algorithm. First, as we noted in Section 3.4, we made a few assumptions regarding the distribution of a variety of social network statistics. These assumptions stemmed from a lack of real data regarding the distributions and likelihoods of different network edge types, user activity levels and quantity of friendships on social networks. As such, a future iteration of the SOFA software would either model observed social network relationship data to create a different distribution of relationship based events or, could implement different stochastic distributions of edge likelihoods and vertex capacities. For example, instead of a stochastic assignment, a different approach to vertex capacity assignment would be to implement different decay functions to model individual change in activity level throughout time. For the edges, we modeled relationship quantity (the edge weights) according to an approximate solution to the maximal diffusion problem. This design choice assumes that the quantity of social interaction between any two users is directly correlated to the overall "level of activeness of both users. With empirical data regarding the quantity and distribution of friendships, future work could improve upon the synthetic edge weights in the SOFA software. A potential change here would be to model a distribution of active and inactive relationships over the existing stochastic node based determinism.

During the initial design of the SOFA software, we made a wishlist of features for the software, some of which were not implemented in its first version. First, we discussed the idea of network bias on social networks in Chapter 2 yet the SOFA software creates graphs from one network perspective. As such, a future improvement on this software would model cross-platform social interactions by adding additional networks of vertices and the connections between nodes of opposing social networks. An example of this is observed on Facebook

when users automatically post their updates from other social networks such as Instagram or Twitter. This type of event would provide a semi-permanent link between the vertices of the two opposing social networks, yet neither network would directly interact with the users of the other. This could be modeled by giving each existing node of in the SOFA software a *team* or by creating entire additional networks and modeling cross-platform profile linkage. These additional networks could also represent sub-graphs that could contain location data, such as mobile towers, or hardware specifications. The integration of additional layers of graphs into the SOFA software would open a whole new avenue for social analytics.

As observed in Chapter 3, the relationships in the graphs are mutual; the next iteration of the SOFA software could take into account relationship direction and varying frequencies of directional communication. Additionally, future work on the SOFA software could implement and explore the effects of different node-based degree distributions on the connectivity of the network. The modularity of the SOFA software design makes it possible to add in additional distributions controlled by a new entry in the settings file. Further, it is essential to note that all parameter bounds in the SOFA software were created with the available hardware in mind; customized software could easily change these restrictions to address different hardware specifications.

Lastly, there are open problems from Chapter 3 as related to the **maximal diffusion problem**. As with any new approach, future research could aim to implement faster solutions with tighter bounds. Further, additional research should aim to classify this problem into its proper category of P or NP. Naturally, due to the observed bottleneck in our implementation, a parallel implementation of our approximation to the maximal diffusion problem would greatly improve overall modeling time.

While there is a large list of features which could be implemented with the SOFA software, there are many open problems which remain in the presented first techniques in social fingerprinting. In Chapter 4, we examined the accuracy of dimensionality reduction and information retrieval as a technique for social fingerprinting. These techniques are well-understood and implemented for various real-time search platforms. As such, future improvements on the SDD and information retrieval approach to social fingerprinting would

build a reduced model of the sparse social adjacency matrix and update the model with newly observed social interactions. This type of model would save overall computation time and enable future researchers to study larger models of social network users.

Additional work from Chapter 4 would also examine the effect of different weighting schemes, such as term frequency or inverse document frequency, on the overall accuracy of the social fingerprinting pipeline. While preliminary tests were created to observe the different accuracies with other weighting schemes, we did not include them here. Further, we also note that future research in these approaches could adopt other measures of accuracy that are more in line with the traditional precision and recall measures. That is to say, imperfect identifications are not that bad when the social fingerprinting algorithm selects one of the closest neighbors of v_i as the candidate identity over v_i. Lastly, we would like to note the open problems derived from Chapter 5. While this methodology presented the best approach for social fingerprinting, real implementations of this methodology require some notion of confidence in the algorithm. As such, future work could either outline the confidence of the techniques presented herein or, use samples of empirical data to create an understanding of each ranking functions performance across time on a real social network. These confidence levels would be required for any real application which utilized unique customer identification for business purposes. With the addition of some features in a future iteration of the SOFA software, the methodology presented in Chapter 5 leaves much room for future ranking games or other learning algorithms.

While each methodology created for this work could launch future research, the advent of the social fingerprint paired with the SOFA software for exploration provides immeasurable potential. Future research could explore additional pipelines for social fingerprint technology. The platform presented in this dissertation provides an easily adaptable vehicle to test the accuracy of a new methodology across a wide range of input parameters. Further, while we presented network and modeling statistics in Chapter 3, the SOFA software provides a great teaching tool for the exploration of basic or advanced techniques in graph mining. Future work could explore the number of connected components, girth, diameter, connectivity, density or a countless set of other topological parameters of the SOFA graphs.

On the other hand, future research may be interested in determining how social network users can evade social fingerprinting techniques. While the results presented herein demonstrate conclusive evidence that any network user can be easily identified, a potential extension of this work could be to exploit how to hide from techniques in digital forensics. Many top companies thrive on the business of user privacy and the protection of ones digital identity is of interest for most internet users. As such, with the knowledge of the ease of this type of user identification, new research in social fingerprint evasion would provide invaluable analytics for any business.

We introduced a new metric for social network analytics, demonstrated its difficulty and provided novel solutions. We outlined accurate techniques for social fingerprinting and discussed assumptions, drawbacks and advances for the methodologies presented in this dissertation. Further, we created software for the future academic researcher to explore some of the open research problems noted herein or to create new approaches to social fingerprinting.

Bibliography

Aggarwal, C. C. (2011). *An introduction to social network data analytics*. Springer. 16

Alteryx (2014). Intuitive workflow for data blending and advanced analytics. 14

Bachrach, Y., Kosinski, M., Graepel, T., Kohli, P., and Stillwell, D. (2012). Personality and patterns of facebook usage. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 24–32. ACM. 14

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509–512. 12

Becker, R. A., Cáceres, R., Hanson, K., Loh, J. M., Urbanek, S., Varshavsky, A., and Volinsky, C. (2011a). Clustering anonymized mobile call detail records to find usage groups. In *Ist Workshop on Pervasive Urban Applications*. 9

Becker, R. A., Caceres, R., Hanson, K., Loh, J. M., Urbanek, S., Varshavsky, A., and Volinsky, C. (2011b). A tale of one city: Using cellular network data for urban planning. *Pervasive Computing, IEEE*, 10(4):18–26. x, 8, 9, 10

Becker, R. A., Volinsky, C., and Wilks, A. R. (2010). Fraud detection in telecommunications: History and lessons learned. *Technometrics*, 52(1):20–33. 8

Berry, D. (2011). The computational turn: Thinking about the digital humanities. *Culture Machine*, 12(0). 2, 7

Blumberg, A. J. and Eckersley, P. (2009). On locational privacy, and how to avoid losing it forever. *Electronic Frontier Foundation*. 10, 11

Bonchi, F., Castillo, C., Gionis, A., and Jaimes, A. (2011). Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):22. 14

Burge, P., Shawe-Taylor, J., Cooke, C., Moreau, Y., Preneel, B., and Stoermann, C. (1997). Fraud detection and management in mobile telecommunications networks. In *Security and Detection, 1997. ECOS 97., European Conference on*, pages 91–96. IET. 8

Chakrabarti, S., Dom, B. E., Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D., and Kleinberg, J. (1999). Mining the web's link structure. *Computer*, 32(8):60–67. 12

Champod, C., Lennard, C. J., Margot, P., and Stoilovic, M. (2004). *Fingerprints and other ridge skin impressions*. CRC press. 1

Chen, C.-H. (2002). Generalized association plots: Information visualization via iteratively generated correlation matrices. *Statistica Sinica*, 12(1):7–30. 54

Clauset, A., Shalizi, C. R., and Newman, M. E. (2009). Power-law distributions in empirical data. *SIAM review*, 51(4):661–703. 19

Cortes, C., Pregibon, D., and Volinsky, C. (2003). Computational methods for dynamic graphs. *Journal of Computational and Graphical Statistics*, 12(4). ix, 4, 5, 11, 13, 14, 15, 21, 46, 56, 60, 77, 79

de C Gatti, M. A., Appel, A. P., dos Santos, C. N., Pinhanez, C. S., Cavalin, P. R., and Neto, S. B. (2013). A simulation-based approach to analyze the information diffusion in microblogging online social network. *Proceedings of the 2013 Winter Simulation Conference*. 14, 15

De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18. 78

de Montjoye, Y.-A., Hidalgo, C. A., Verleysen, M., and Blondel, V. D. (2013). Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3. 10, 11, 14, 15, 79

Delre, S. A., Jager, W., and Janssen, M. A. (2007). Diffusion dynamics in small-world networks with heterogeneous consumers. *Computational and Mathematical Organization Theory*, 13(2):185–202. 13

Detectives, D. (2014). The latest and most innovative use of analytics and big data to learn about–and drive–your business. 14

Doran, D., Mendiratta, V., Phadke, C., and Uzunalioglu, H. (2012). The importance of outlier relationships in mobile call graphs. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 24–29. IEEE. 19

Draper, N. R. and Smith, H. (1981). *Applied regression analysis 2nd ed.* New York New York John Wiley and Sons. 28, 29, 36

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2):229–236. 46

Easley, D. and Kleinberg, J. (2010). Networks, crowds, and markets. *Cambridge Univ Press*, 6(1):6–1. 16

Gallagher, R. (2013). How anonymous cellphone location data leave "fingerprints" that could identify you. *Slate*. 1, 10

Gatti, M., Appel, A. P., Pinhanez, C., dos Santos, C., Gribel, D., Cavalin, P., and Neto, S. B. (2013). Large-scale multi-agent-based modeling and simulation of microblogging-based online social network. *Proc. of Int. Work. on Multi-Agent-based Simul.* 14

Gomez Rodriguez, M., Leskovec, J., and Schölkopf, B. (2013). Structure and dynamics of information pathways in online media. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 23–32. ACM. 13

Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, 453(7196):779–782. 14

Govan, A. Y., Langville, A. N., and Meyer, C. D. (2009). Offense-defense approach to ranking team sports. *Journal of Quantitative Analysis in Sports*, 5(1):1151. 70

Gross, D. (2013). How your movements create a gps fingerprint. *CNN*. 1, 10

Grosser, H., Britos, P., and García-Martínez, R. (2005). Detecting fraud in mobile telephony using neural networks. In *Innovations in Applied Artificial Intelligence*, pages 613–615. Springer. 8

Gruhl, D., Guha, R., Liben-Nowell, D., and Tomkins, A. (2004). Information diffusion through blogspace. In *Proceedings of the 13th international conference on World Wide Web*, pages 491–501. ACM. 13

Isaacman, S., Becker, R., Cáceres, R., Kobourov, S., Martonosi, M., Rowland, J., and Varshavsky, A. (2011). Identifying important places in peoples lives from cellular network data. In *Pervasive Computing*, pages 133–151. Springer. 9

Janssen, M. A. and Jager, W. (2003). Simulating market dynamics: Interactions between consumer psychology and social networks. *Artificial Life*, 9(4):343–356. 13, 15

Kempe, D., Kleinberg, J., and Tardos, É. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM. 13

Keppel, D. (2000). Time(1). 30, 32

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The web as a graph: Measurements, models, and methods. In *Computing and combinatorics*, pages 1–17. Springer. 12

Knight, W. (2013). How access to location data could trample your privacy. *MIT Technology Review*. 1, 10

Knoke, D. and Yang, S. (2008). *Social network analysis*, volume 154. Sage. 16

Kolda, T. G. and O'Leary, D. P. (1998). A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems (TOIS)*, 16(4):322–346. 48

Kolda, T. G. and O'Leary, D. P. (2000). Computation and uses of the semidiscrete matrix decomposition. *ACM Transactions on Mathematical Software*, 26. 48

Kossinets, G. and Watts, D. J. (2006). Empirical analysis of an evolving social network. *Science*, 311(5757):88–90. 1, 7

Langville, A. N. and Meyer, C. C. D. (2012). *Who's# 1?: The Science of Rating and Ranking*. Princeton University Press. 63, 70

Lawrence, I. and Lin, K. (1989). A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, pages 255–268. 54

Laya, P. (2011). Do you pay enough for advertising? one big corporation spent a jaw-dropping $4.2 billion last year. *Business Insider*. 2, 14

Leskovec, J. (2014). Snap: Stanford network analysis project. 13, 15, 24, 42

Leskovec, J., Chakrabarti, D., Kleinberg, J., and Faloutsos, C. (2005a). Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *Knowledge Discovery in Databases: PKDD 2005*, pages 133–145. Springer. 12

Leskovec, J. and Faloutsos, C. (2007). Mining large graphs. *Tutorial ECML/PKDD*. 11

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005b). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM. 12

Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005c). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM. 12

Liu, D. and Chen, X. (2011). Rumor propagation in online social networks like twitter–a simulation study. In *Multimedia Information Networking and Security (MINES), 2011 Third International Conference on*, pages 278–282. IEEE. 13, 15

McConnell, S. and Skillicorn, D. (2001). Outlier detection using semi-discrete decomposition. Technical report, Citeseer. 56

McConnell, S. and Skillicorn, D. (2002). Semidiscrete decomposition: A bump hunting technique. In *Australasian Data Mining Workshop*, pages 75–82. Citeseer. 56

Nanavati, A. A., Singh, R., Chakraborty, D., Dasgupta, K., Mukherjea, S., Das, G., Gurumurthy, S., and Joshi, A. (2008). Analyzing the structure and evolution of massive telecom graphs. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):703–718. 19

Newman, M. E., Watts, D. J., and Strogatz, S. H. (2002). Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572. 11

O'Leary, D. P. and Peleg, S. (1983). Digital image compression by outer product expansion. *IEEE Transactions on Communications*, 31(3):441. 48

Palmer, C. R. and Steffan, J. G. (2000). Generating network topologies that obey power laws. In *Global Telecommunications Conference, 2000. GLOBECOM'00. IEEE*, volume 1, pages 434–438. IEEE. 12, 13

Ragghianti, G. (2014). Newton program documentation. viii, 6, 27

Robins, G., Pattison, P., Kalish, Y., and Lusher, D. (2007). An introduction to exponential random graph models for social networks. *Social networks*, 29(2):173–191. 11

Rosset, S., Murad, U., Neumann, E., Idan, Y., and Pinkas, G. (1999). Discovery of fraud rules for telecommunications challenges and solutions. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 409–413. ACM. 8

Schommer, C. (2009). Discovering fraud behaviour in call detailed records. 8

Seshadri, M., Machiraju, S., Sridharan, A., Bolot, J., Faloutsos, C., and Leskove, J. (2008). Mobile call graphs: beyond power-law and lognormal distributions. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 596–604. ACM. 14, 43

Siek, J. G., Lee, L.-Q., and Lumsdaine, A. (2001). *Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education. 21, 24

Smith, A. (2012). Americans and their gadgets. *Pew Research Center*. 1

Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968):1018–1021. 9

Uzzi, B. (1997). Social structure and competition in interfirm networks: The paradox of embeddedness. *Administrative science quarterly*, pages 35–67. 1, 7

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442. 11

Weiss, G. M. (2005). Data mining in telecommunications. In *Data Mining and Knowledge Discovery Handbook*, pages 1189–1201. Springer. 7, 8

Weisstein, E. W. (2011). Least squares fitting–exponential. *MathWorld-A Wolfram Web Resource*. 29

Wuchty, S. and Uzzi, B. (2011). Human communication dynamics in digital footsteps: a study of the agreement between self-reported ties and email networks. *PloS one*, 6(11):e26972. 1, 7

Xing, D. and Girolami, M. (2007). Employing latent dirichlet allocation for fraud detection in telecommunications. *Pattern Recognition Letters*, 28(13):1727–1734. 8

Xing, K., Cheng, X., Chen, D., and Du, D. H.-C. (2009). Topology inference in wireless mesh networks. *Springer*, pages 159–168. 58, 59

Zyga, L. (2013). Study shows how easy it is to determine someone's identity with cell phone data. *Physics.org*. 1, 10

Appendices

Appendix A: All Results from Chapter 3

All Time and Memory Results

Average user minutes for weighted graph construction.

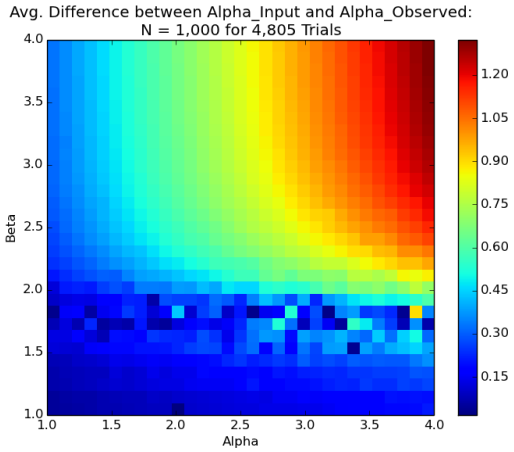
$\|V(G)\|$	Trials	Avg. Time	$\sigma(Time)$	Avg. GB	$\sigma(GB)$
10,000	200	0.0135	0.0115	0.0431	0.0000
100,000	200	5.4639	4.6327	0.3782	0.0002
200,000	200	23.0147	15.3446	0.7523	0.0011
300,000	200	52.8967	31.5479	1.1348	0.0027
400,000	200	89.2375	48.4665	1.5009	0.0015
500,000	200	129.2849	69.9264	1.8678	0.0001
600,000	200	181.6926	89.4631	2.2533	0.0022
700,000	200	268.3056	143.7936	2.6240	0.0024
800,000	200	386.8604	201.6467	2.9939	0.0024
900,000	200	467.2465	249.5532	3.3601	0.0012
1,000,000	200	590.2611	282.9561	3.7285	0.0004

Average user minutes for non-weighted graph construction.

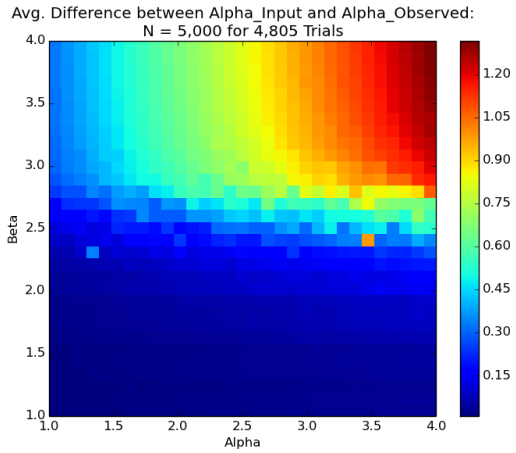
$\|V(G)\|$	Trials	Avg. $\|E(G)\|$	$\sigma(\|E(G)\|)$	Avg. GB	$\sigma(GB)$	Avg. Time	$\sigma(Time)$
10,000	500	19,795	2.1072	0.0441	0.0000	0.0010	0.0004
100,000	500	198,349	1.5685	0.3870	0.0000	0.0315	0.0083
200,000	500	396,732	1.6156	0.7690	0.0010	0.1105	0.0237
250,000	500	495,926	1.4510	0.9588	0.0002	0.1734	0.0407
300,000	500	595,112	1.4550	1.1587	0.0027	0.2475	0.0595
400,000	500	793,498	1.5662	1.5351	0.0013	0.4343	0.1198
500,000	500	991,881	1.7838	1.9121	0.0002	0.6854	0.2321
600,000	500	1,190,255	1.5689	2.3035	0.0035	1.1567	0.7902
700,000	500	1,388,640	1.4309	2.6837	0.0032	1.5886	0.9566
750,000	500	1,487,845	1.7132	2.8737	0.0032	1.9016	1.7001
800,000	500	1,587,024	1.4135	3.0629	0.0031	2.1723	1.4348
900,000	500	1,785,408	1.4819	3.4392	0.0016	2.8515	1.6579
1,000,000	500	1,983,792	1.3751	3.8173	0.0006	3.7338	2.0408
2,500,000	10	4,959,536	1.0325	9.5480	0.0041	19.7368	5.7860
5,000,000	10	9,919,104	0.9574	19.0768	0.0048	119.5531	63.5054
7,500,000	10	14,878,683	0.7071	28.6036	0.0079	508.8857	39.5479
10,000,000	10	19,838,243	0.4588	38.1321	0.0060	946.1793	76.5589

All Model Fit Results

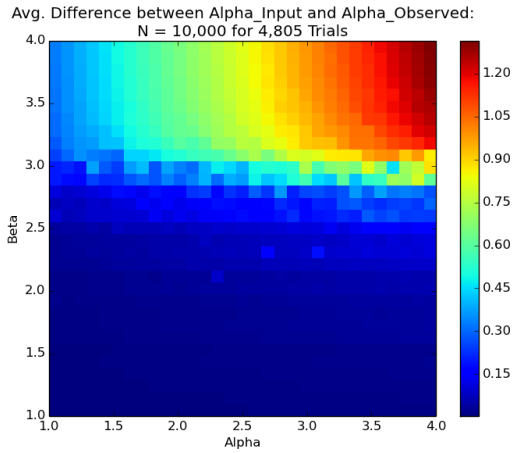
Percent difference between user input and observed data



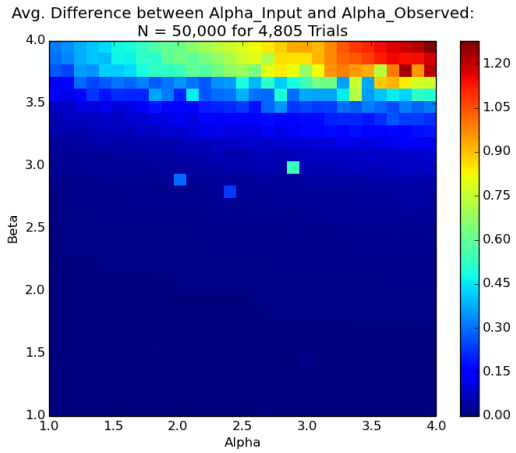
The percent difference between the input α and the observed α across $4,805$ different simulations where $N = 1,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input α and the observed α for 5 trials is recorded and shaded according to the scale on the right.



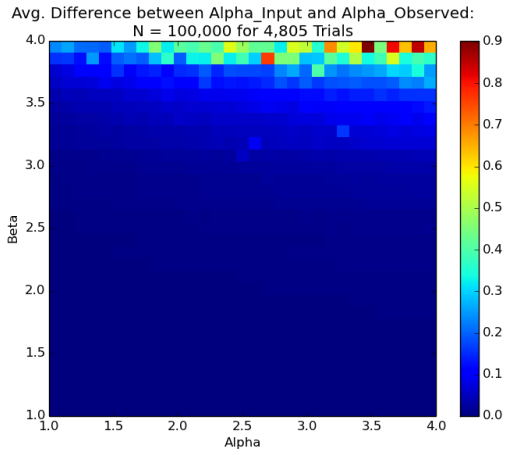
The percent difference between the input α and the observed α across $4,805$ different simulations where $N = 5,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input α and the observed α for 5 trials is recorded and shaded according to the scale on the right.



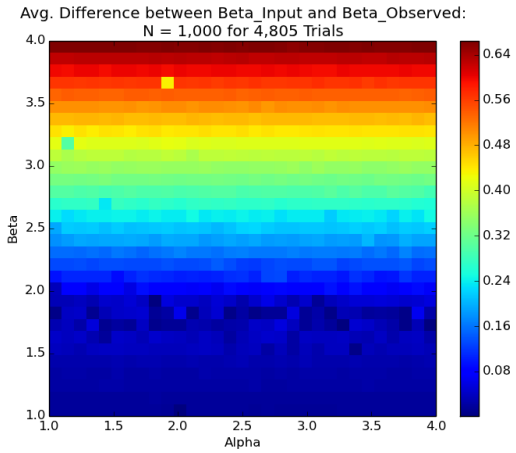
The percent difference between the input α and the observed α across $4,805$ different simulations where $N = 10,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input α and the observed α for 5 trials is recorded and shaded according to the scale on the right.



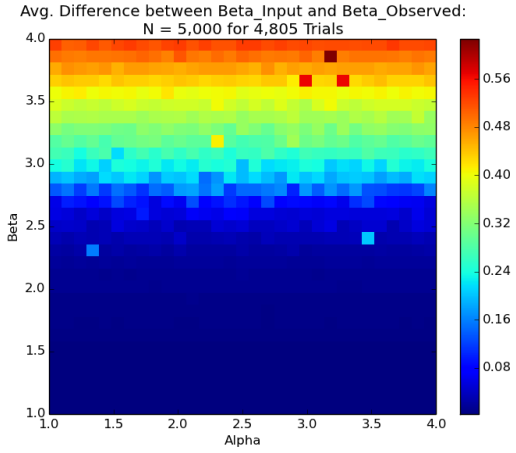
The percent difference between the input α and the observed α across $4,805$ different simulations where $N = 50,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input α and the observed α for 5 trials is recorded and shaded according to the scale on the right.



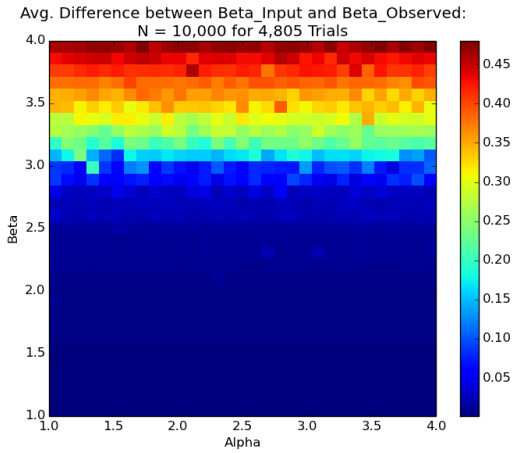
The percent difference between the input α and the observed α across $4,805$ different simulations where $N = 100,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input α and the observed α for 5 trials is recorded and shaded according to the scale on the right.



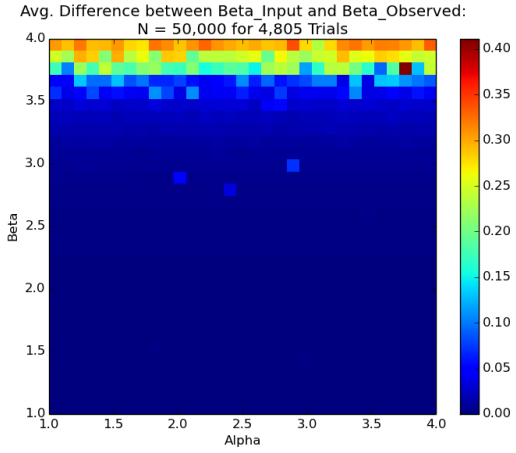
The percent difference between the input β and the observed β across $4,805$ different simulations where $N = 1,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input β and the observed β for 5 trials is recorded and shaded according to the scale on the right.



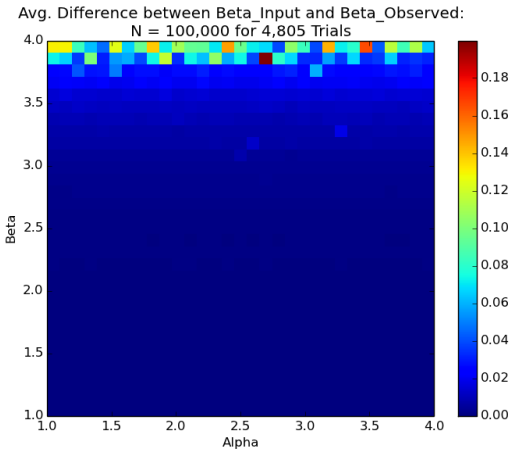
The percent difference between the input β and the observed β across $4,805$ different simulations where $N = 5,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input β and the observed β for 5 trials is recorded and shaded according to the scale on the right.



The percent difference between the input β and the observed β across $4,825$ different simulations where $N = 10,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input β and the observed β for 5 trials is recorded and shaded according to the scale on the right.

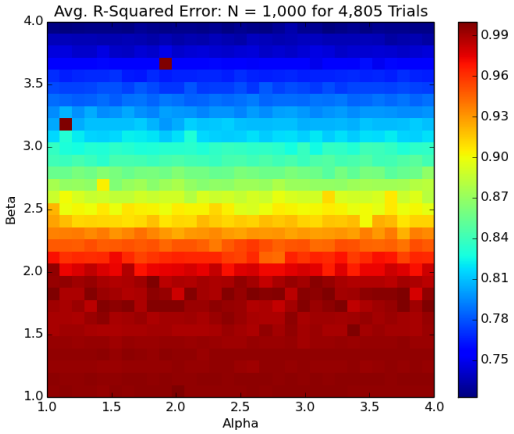


The percent difference between the input β and the observed β across $4,805$ different simulations where $N = 50,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input β and the observed β for 5 trials is recorded and shaded according to the scale on the right.

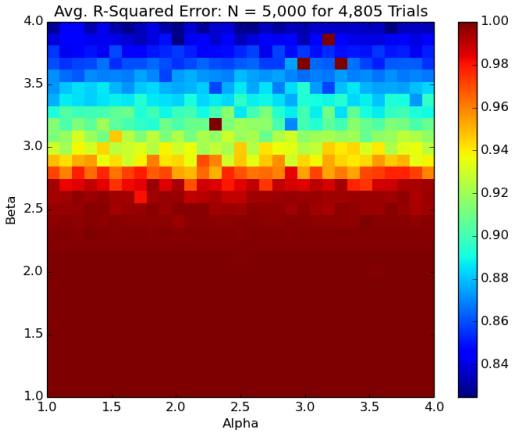


The percent difference between the input β and the observed β across $4,805$ different simulations where $N = 100,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The average percent difference between the input β and the observed β for 5 trials is recorded and shaded according to the scale on the right.

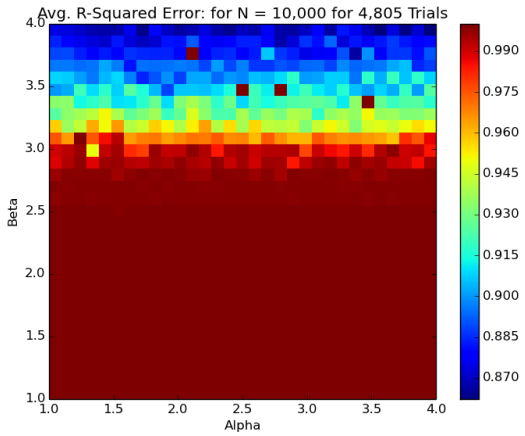
Fit of model to observed data



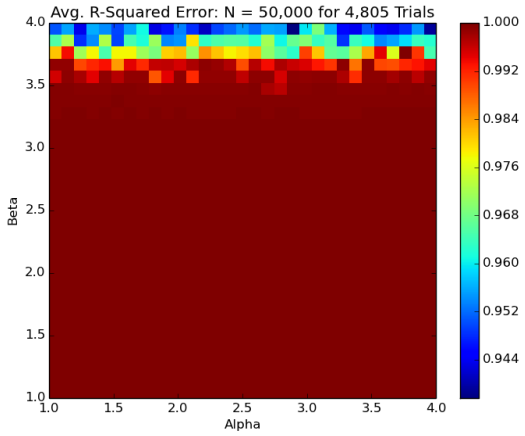
The average R-squared error across $4,805$ different simulations where $N = 1,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The R-squared error of the least-squares regression for 5 trials is recorded and shaded according to the scale on the right.



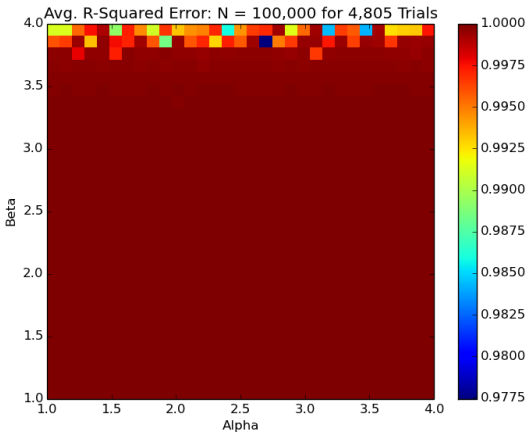
The average R-squared error across $4,805$ different simulations where $N = 5,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The R-squared error of the least-squares regression for 5 trials is recorded and shaded according to the scale on the right.



The average R-squared error across $4,805$ different simulations where $N = 10,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The R-squared error of the least-squares regression for 5 trials is recorded and shaded according to the scale on the right.

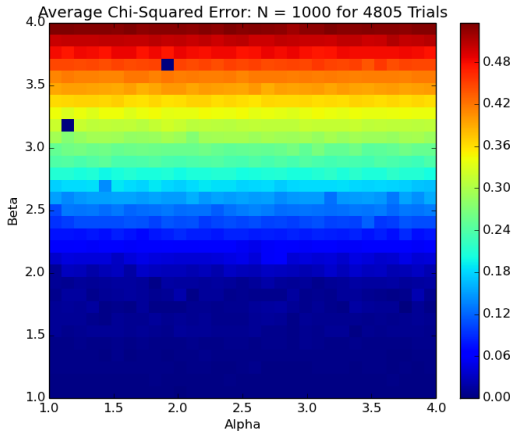


The average R-squared error across $4,805$ different simulations where $N = 50,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The R-squared error of the least-squares regression for 5 trials is recorded and shaded according to the scale on the right.

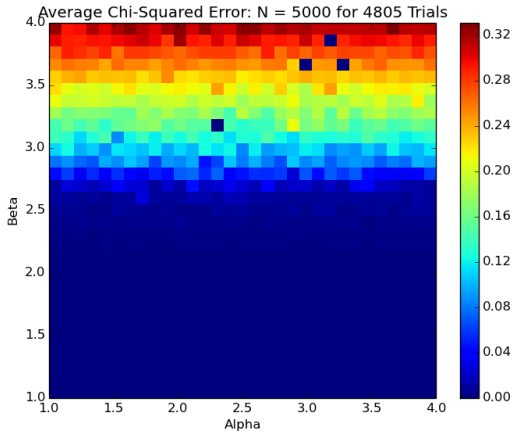


The average R-squared error across $4,805$ different simulations where $N = 100,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The R-squared error of the least-squares regression for 5 trials is recorded and shaded according to the scale on the right.

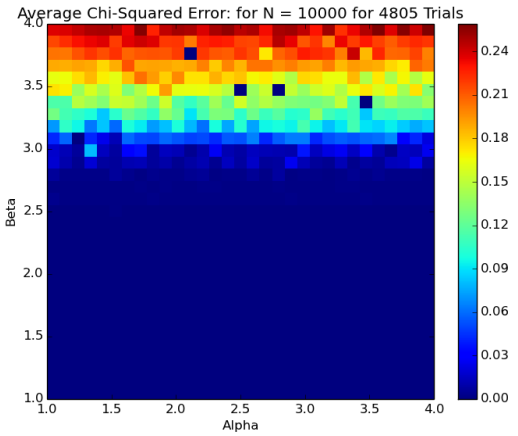
Fit of observed model to user input



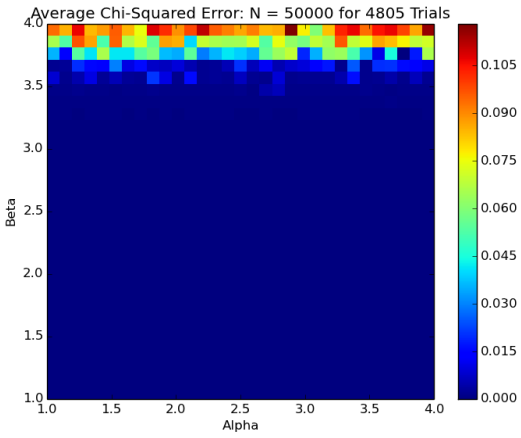
The average χ-squared error across $4,805$ different simulations where $n = 1,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The χ-squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the right.



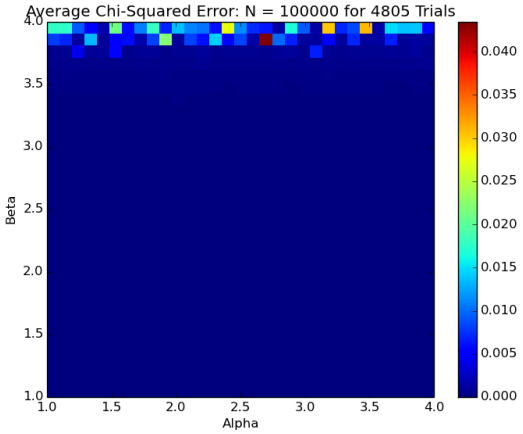
The average χ-squared error across $4,805$ different simulations where $n = 5,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The χ-squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the right.



The average χ-squared error across $4,805$ different simulations where $n = 10,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The χ-squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the right.



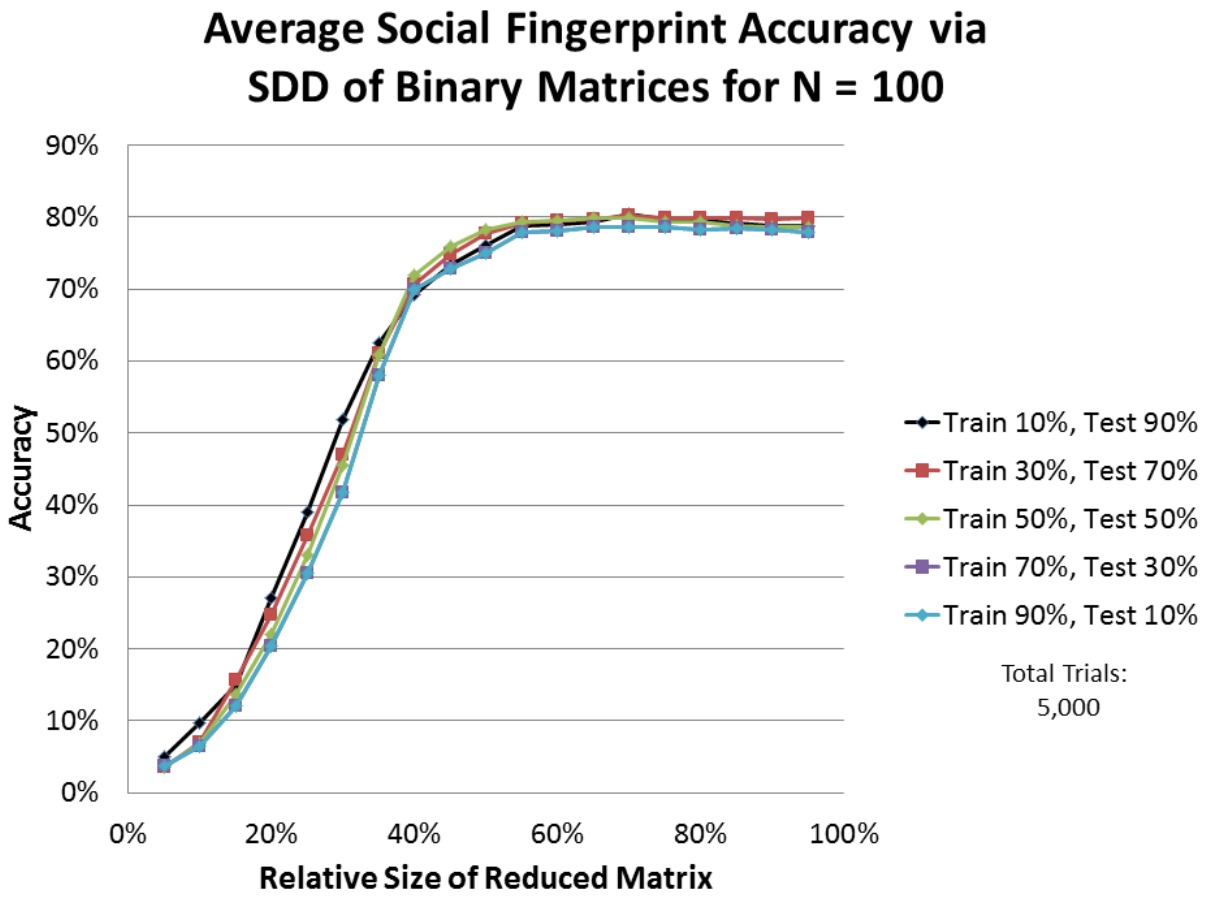
The average χ-squared error across $4,805$ different simulations where $n = 50,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The χ-squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the right.

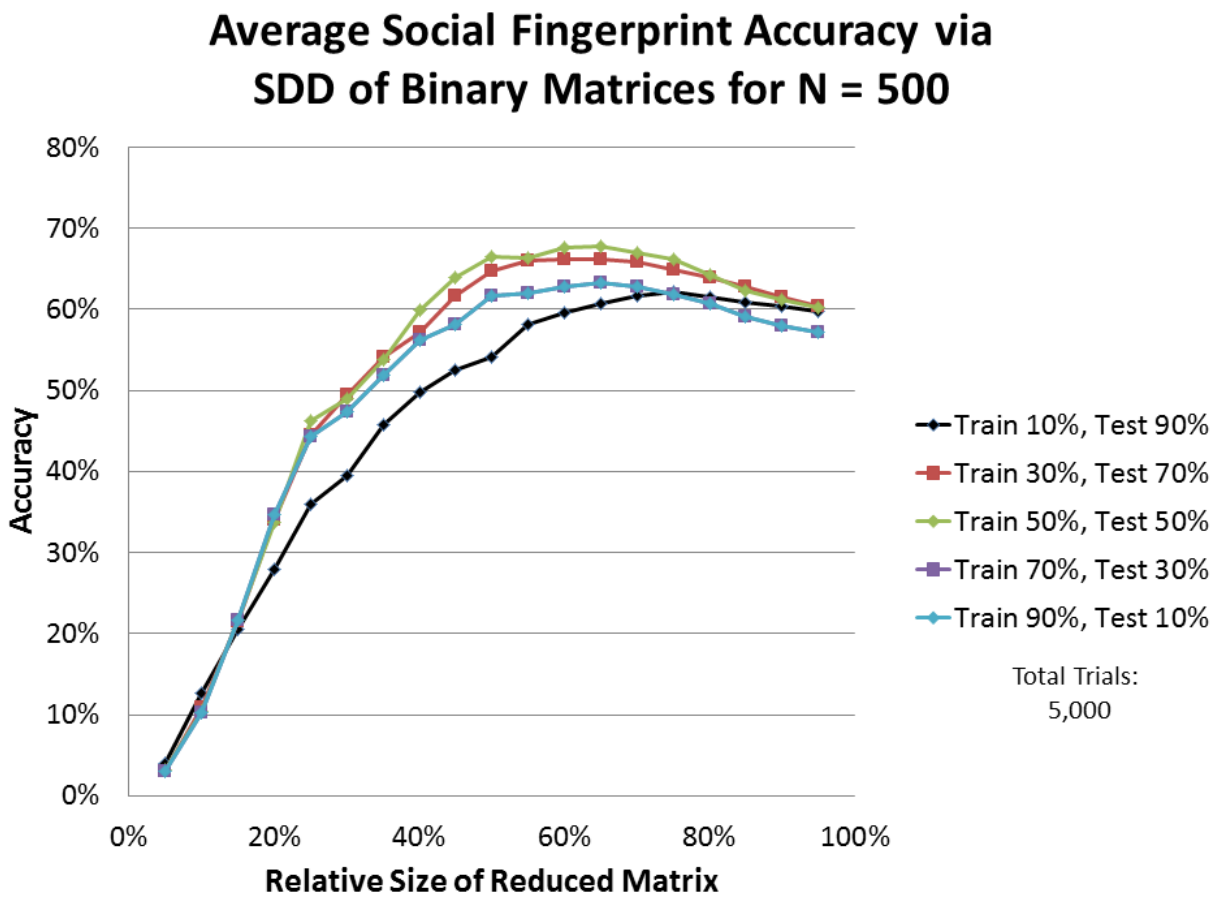


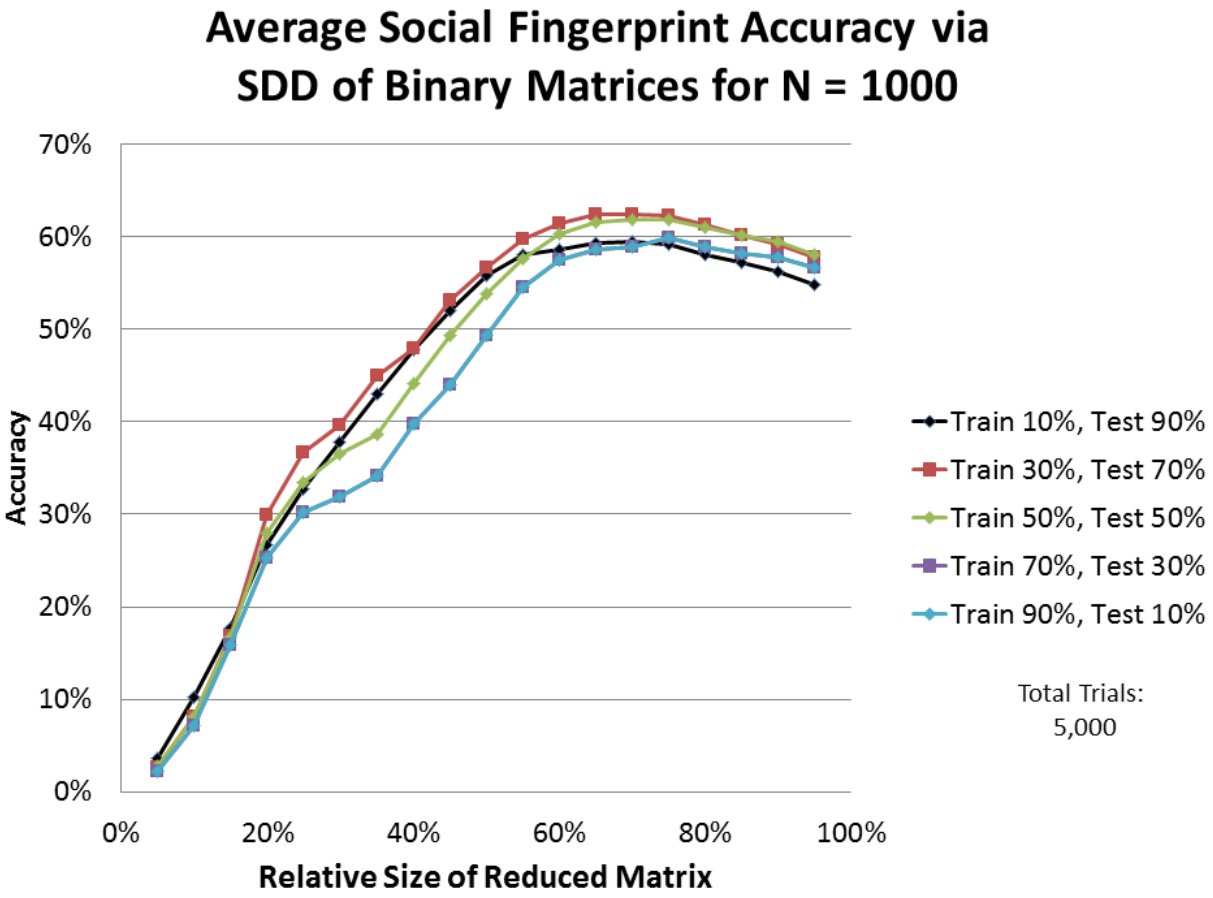
The average χ-squared error across $4,805$ different simulations where $n = 100,000$. The x-axis ranges across 31 different input values for α and the y-axis ranges across various 31 different input values for β. As such, there are a total of 961 different combinations of input parameters represented across this grid. The χ-squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the right.

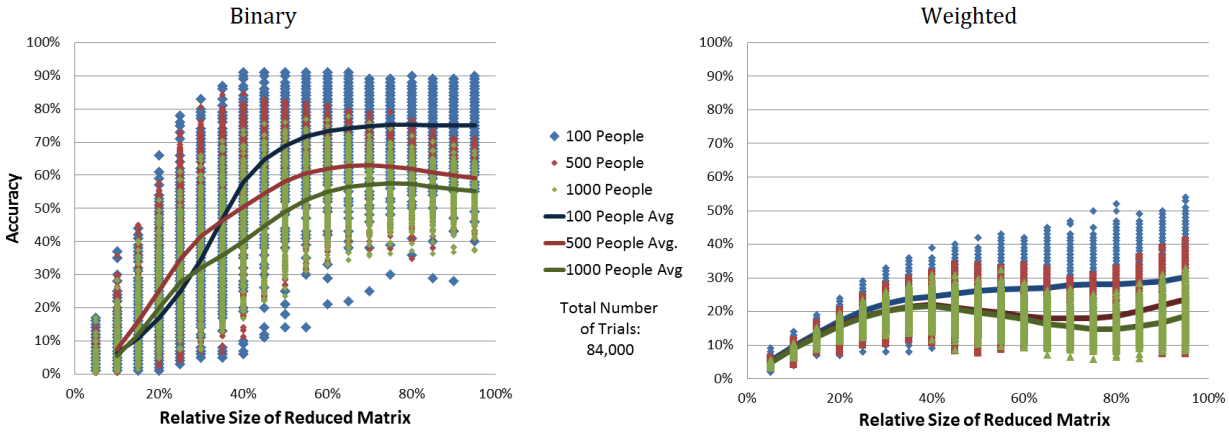
Appendix B: All Results from Chapter 4

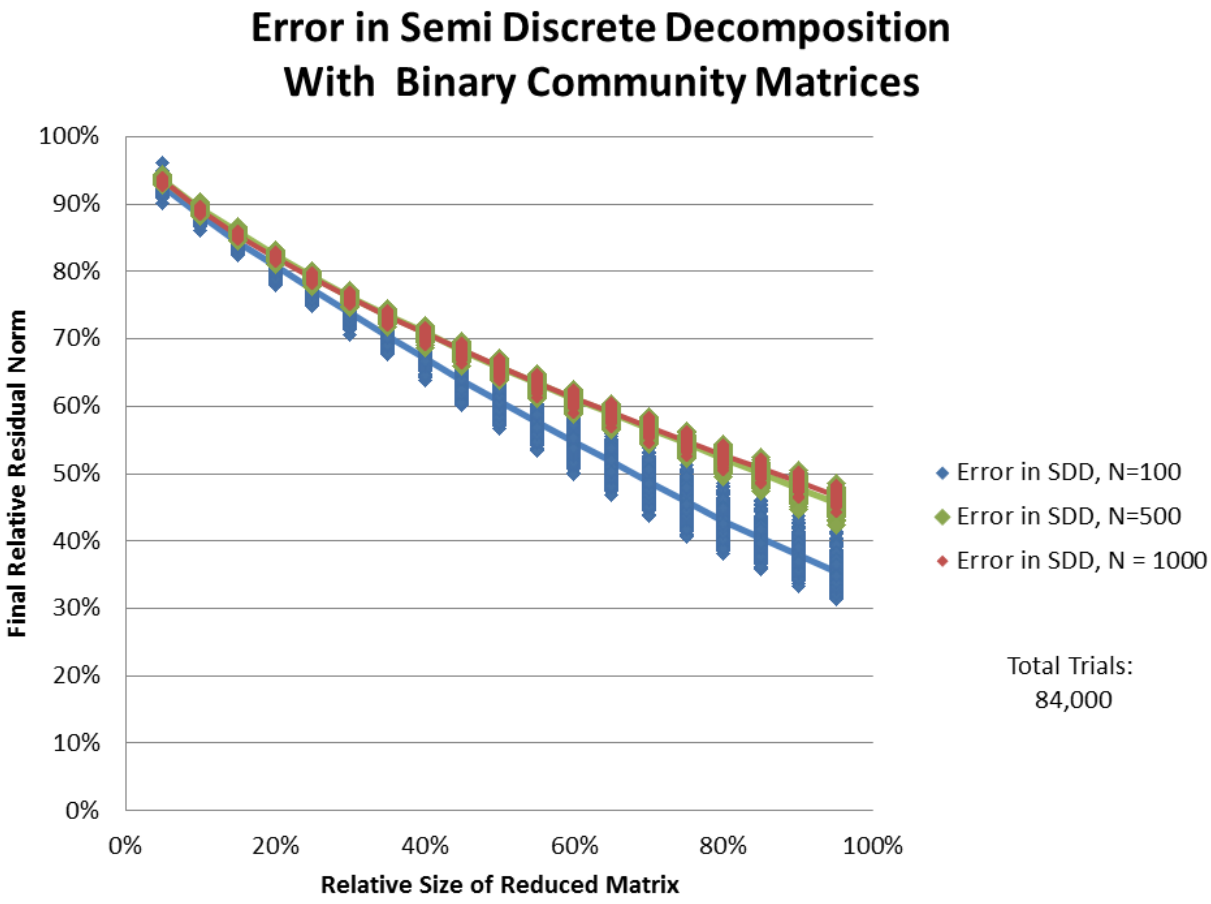
Binary Matrix Results

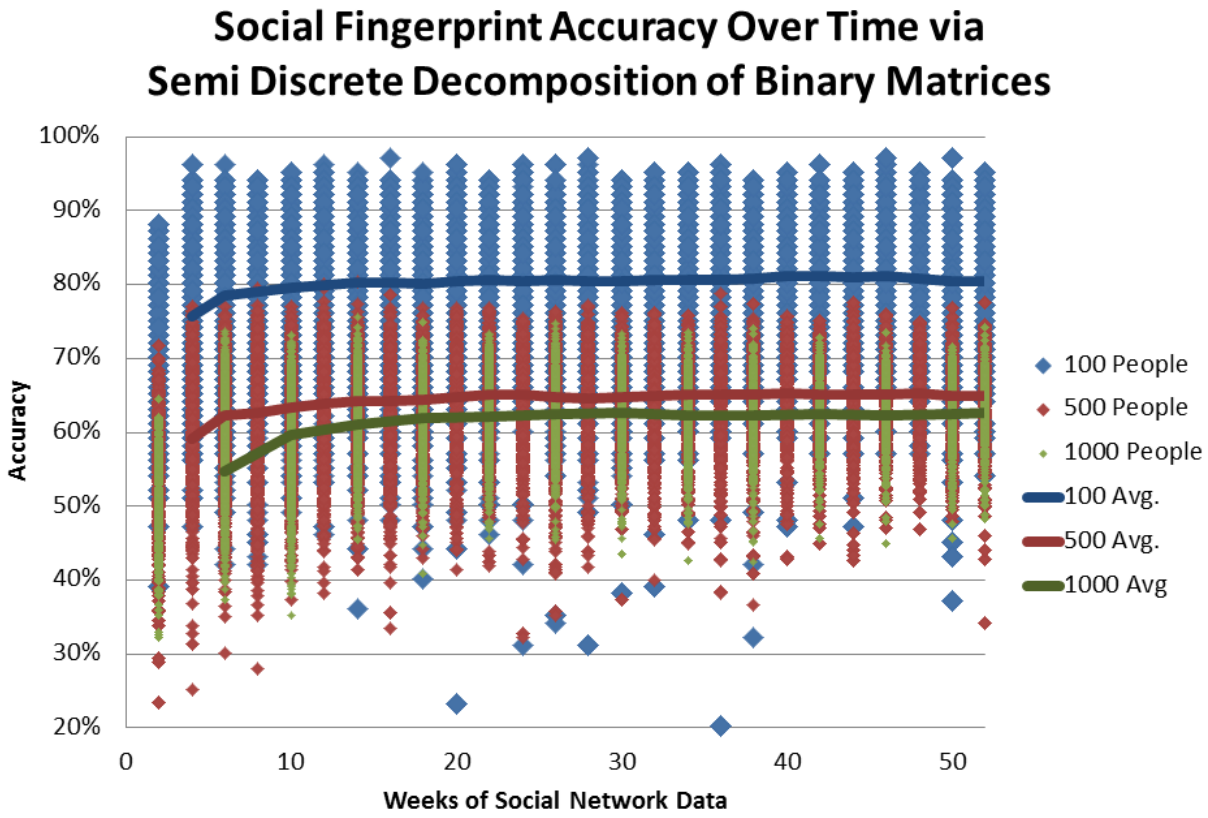



Average Social Fingerprint Accuracy via SDD of Binary Matrices for N = 500
Accuracy
80%
70%
60%
50%
40%
30%
20%
10%
0%
0%
20%
40%
60%
80%
100%
Relative Size of Reduced Matrix
Train 10%, Test 90%
Train 30%, Test 70%
Train 50%, Test 50%
Train 70%, Test 30%
Train 90%, Test 10%
Total Trials: 5,000

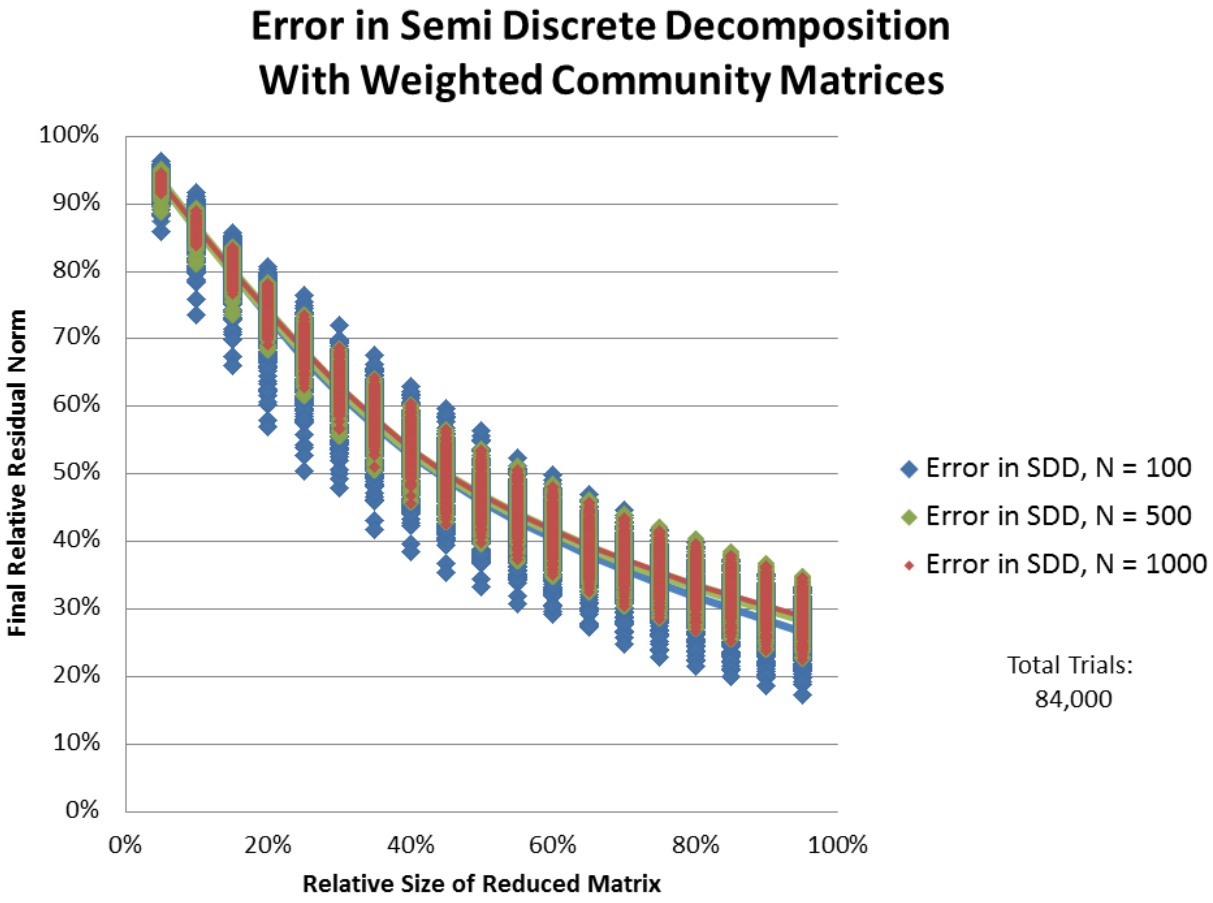

Average Social Fingerprint Accuracy via SDD of Binary Matrices for N = 1000
Accuracy
70%
60%
50%
40%
30%
20%
10%
0%
0%
20%
40%
60%
80%
100%
Relative Size of Reduced Matrix
Train 10%, Test 90%
Train 30%, Test 70%
Train 50%, Test 50%
Train 70%, Test 30%
Train 90%, Test 10%
Total Trials: 5,000

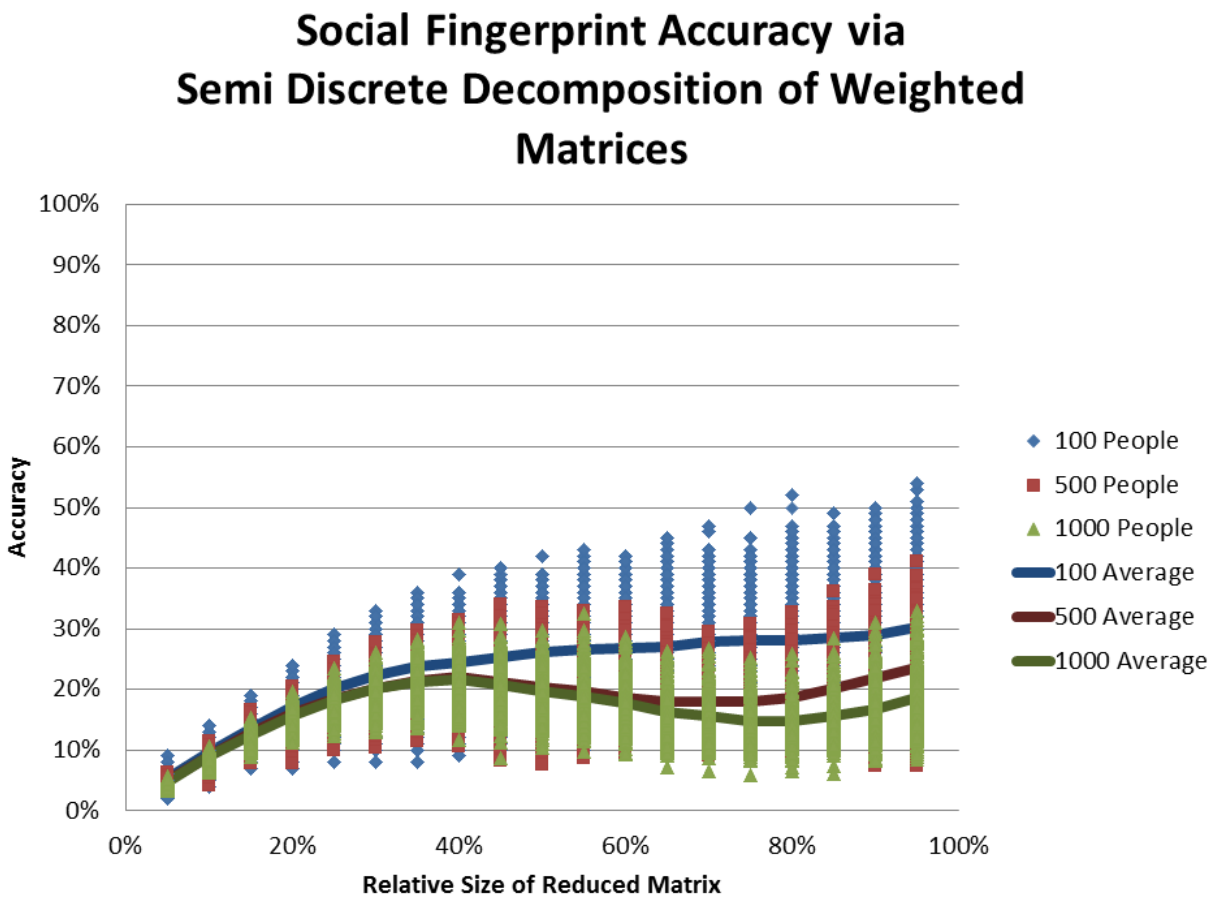

Social Fingerprint Accuracy via Semidiscrete Decomposition
Binary
Weighted
Accuracy
Relative Size of Reduced Matrix
100 People
500 People
1000 People
100 People Avg
500 People Avg.
1000 People Avg
Total Number of Trials: 84,000


Error in Semi Discrete Decomposition With Binary Community Matrices
Final Relative Residual Norm
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
0%
20%
40%
60%
80%
100%
Relative Size of Reduced Matrix
Error in SDD, N=100
Error in SDD, N=500
Error in SDD, N = 1000
Total Trials:
84,000


Social Fingerprint Accuracy Over Time via
Semi Discrete Decomposition of Binary Matrices
100%
90%
80%
70%
60%
50%
40%
30%
20%
Accuracy
0
10
20
30
40
50
Weeks of Social Network Data
100 People
500 People
1000 People
100 Avg.
500 Avg.
1000 Avg

Weighted Matrix Results




Social Fingerprint Accuracy via Semi Discrete Decomposition of Weighted Matrices
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Accuracy
0%
20%
40%
60%
80%
100%
Relative Size of Reduced Matrix
100 People
500 People
1000 People
100 Average
500 Average
1000 Average

Case Study Results by Node ID

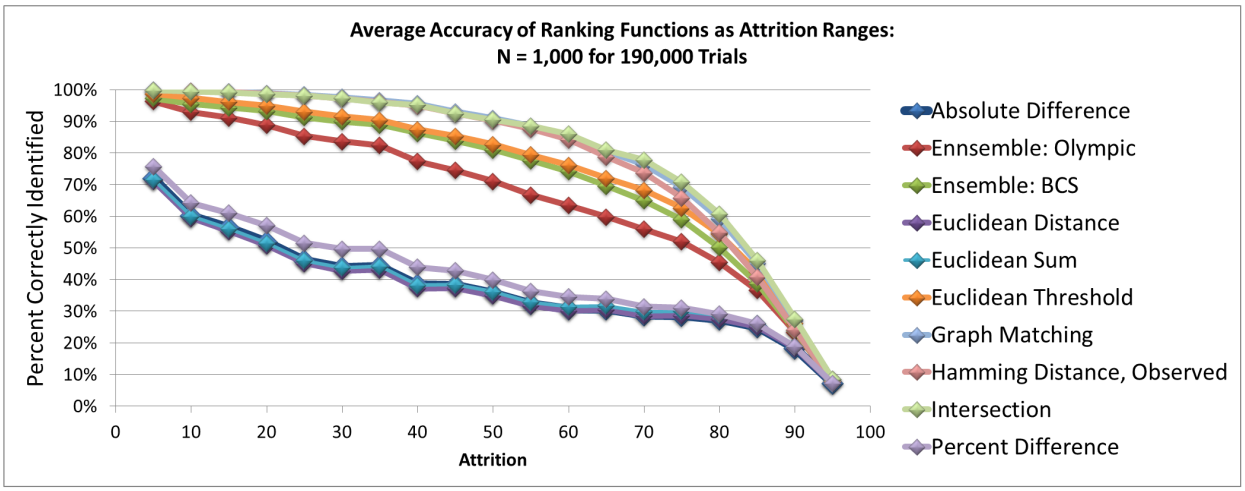
ID	B	W	Deg.	Friend List												
0	1	0	5	2	15	33	51	67								
1	1	0	3	25	37	81										
2	1	0	8	0	5	6	8	11	15	52	69					
3	1	0	6	5	8	15	22	41	65							
4	0	0	8	22	23	25	31	39	42	79	97					
5	0	0	13	32	3	6	8	11	12	14	18	21	23	35	76	89
6	0	0	13	32	5	12	15	17	18	20	23	35	38	60	61	72
7	1	0	4	15	24	39	41									
8	1	0	7	2	3	5	23	25	26	55						
9	0	1	2	54	99											
10	1	0	3	28	50	60										
11	1	0	6	2	5	17	42	69	93							
12	0	0	10	5	6	20	23	29	47	75	88	96	97			
13	1	0	2	23	61											
14	1	0	6	5	17	23	39	58	94							
15	1	1	11	10	2	3	6	7	19	20	23	43	72	89		
16	0	0	2	54	64											
17	1	0	7	6	11	14	18	27	46	61						
18	1	0	5	5	6	17	56	95								
19	0	0	3	15	32	34										
20	0	0	3	6	12	15										
21	1	1	5	5	24	37	38	81								
22	1	0	5	3	4	49	52	91								
23	0	0	12	24	5	6	8	12	13	14	15	44	61	84	94	
24	1	0	7	7	21	25	55	74	75	92						
25	1	0	5	1	4	8	24	33								
26	1	0	3	8	64	80										
27	1	0	3	17	78	86										
28	1	1	3	10	70	78										
29	1	0	3	12	56	67										

ID	B	W	Deg.	Friend List			
30	1	0	2	63	86		
31	1	0	3	4	37	52	
32	0	1	3	19	43	76	
33	1	0	2	0	25		
34	0	1	3	19	40	90	
35	1	1	2	5	6		
36	0	1	4	47	48	66	77
37	0	0	4	1	21	31	45
38	0	1	4	6	21	66	84
39	1	0	3	4	7	14	
40	1	0	2	34	90		
41	1	1	3	3	7	68	
42	1	0	3	4	11	88	
43	1	0	3	15	32	99	
44	1	0	3	23	52	70	
45	1	0	2	37	83		
46	1	0	3	17	59	98	
47	1	0	3	12	36	74	
48	1	0	3	36	59	77	
49	1	1	2	22	82		
50	1	0	2	10	93		
51	1	0	2	0	56		
52	1	0	4	2	22	31	44
53	1	0	2	68	71		
54	1	0	2	9	16		
55	1	0	2	8	24		
56	1	0	4	18	29	51	91
57	0	0	4	73	80	95	96
58	1	0	2	14	62		
59	1	1	2	46	48		
60	1	0	4	6	10	62	71
61	1	1	4	6	13	17	23
62	1	0	2	58	60		
63	1	0	2	30	87		
64	0	1	2	16	26		
65	1	1	2	3	85		
66	1	1	2	36	38		
67	1	1	2	0	29		
68	1	1	2	41	53		
69	1	0	2	2	11		

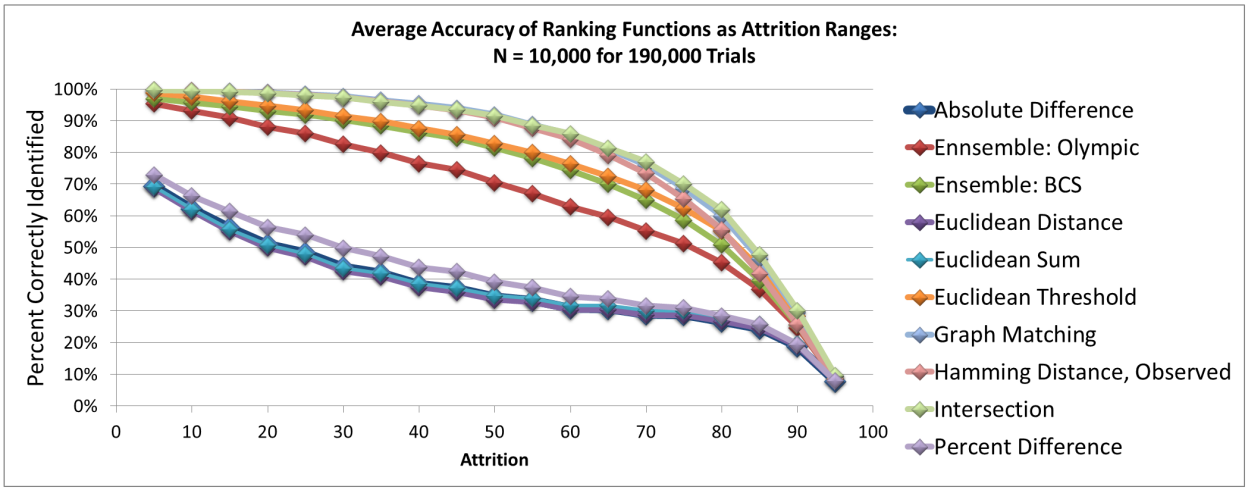
ID	B	W	Deg.	Friend List	
70	1	0	2	28	44
71	1	1	2	53	60
72	1	1	2	6	15
73	1	0	2	57	98
74	1	0	2	24	47
75	1	0	2	12	24
76	1	0	2	5	32
77	1	0	2	36	48
78	1	1	2	27	28
79	1	1	2	4	87
80	1	0	2	26	57
81	1	0	2	1	21
82	1	0	2	49	85
83	1	0	2	45	92
84	1	0	2	23	38
85	1	0	2	65	82
86	1	1	2	27	30
87	0	0	2	63	79
88	1	0	2	12	42
89	0	0	2	5	15
90	1	1	2	34	40
91	1	0	2	22	56
92	0	1	2	24	83
93	1	0	2	11	50
94	1	0	2	14	23
95	1	0	2	18	57
96	1	0	2	12	57
97	1	0	2	4	12
98	1	1	2	46	73
99	1	1	2	9	43

Appendix C: All Results from Chapter 5

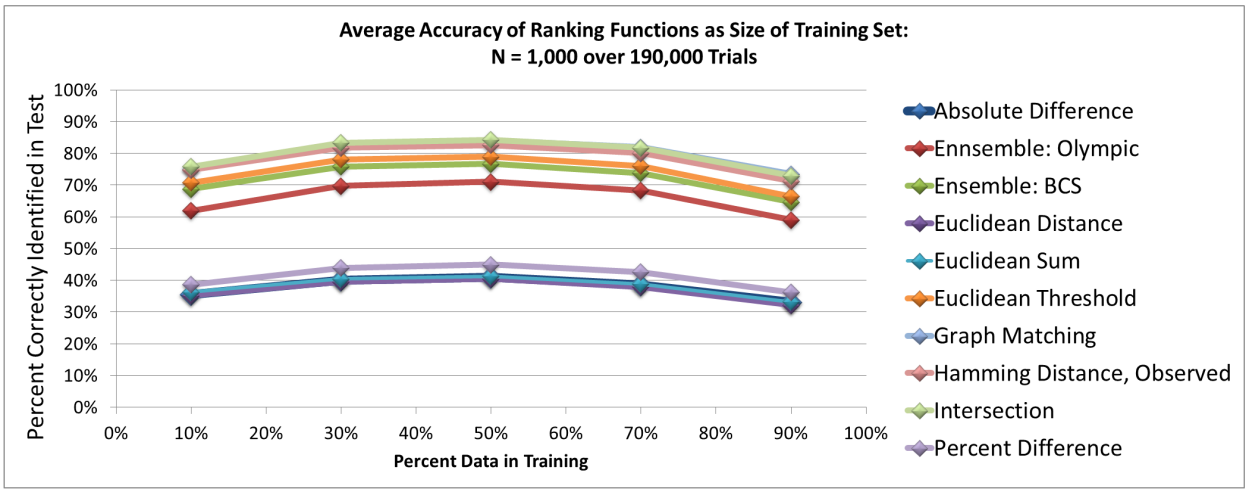
Additional Results Images



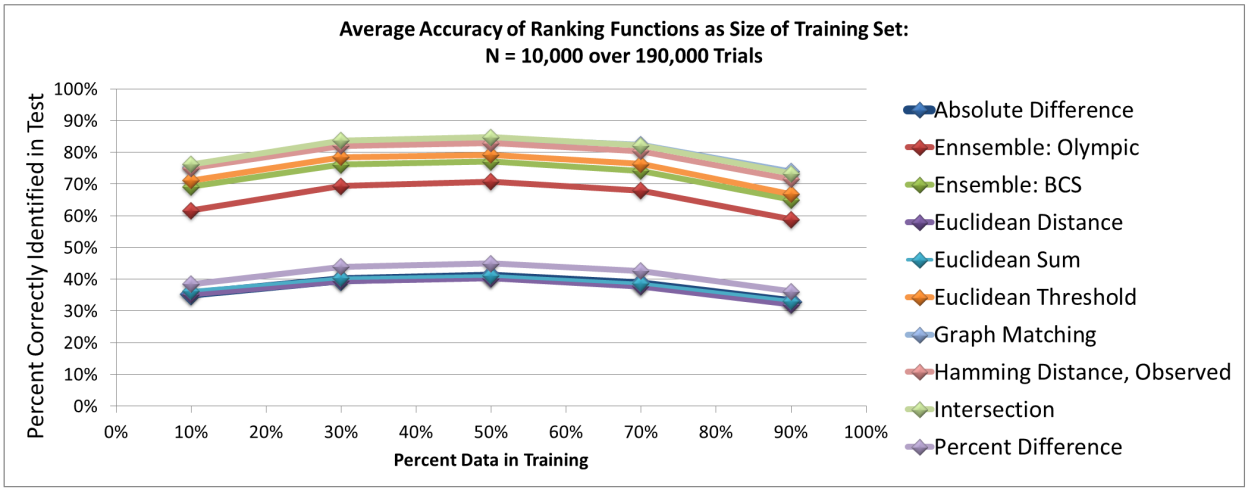
The accuracy of each ranking function as the attrition rate ranges from 5 to 95. This figure shows the average results for $N = 1,000$ for 190,000 tests.



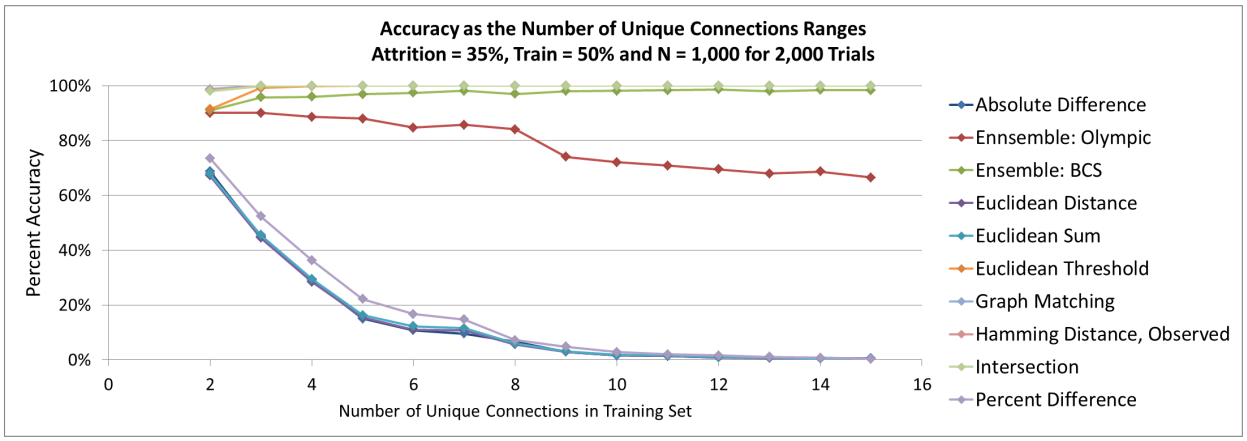
The accuracy of each ranking function as the attrition rate ranges from 5 to 95. This figure shows the average results for $N = 10,000$ for 190,000 tests.



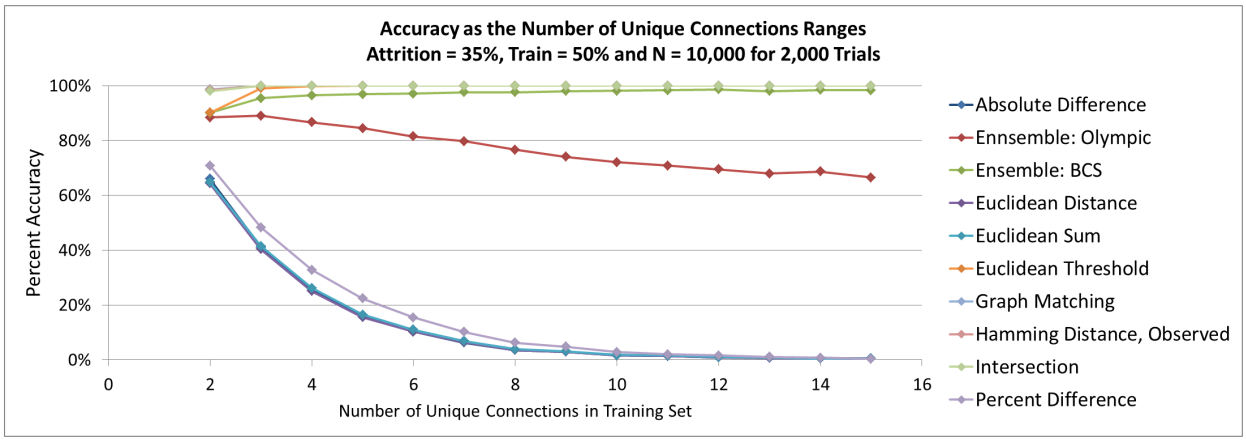
The average accuracy across all ranking functions and all values of attrition as the training window ranges for $N = 1,000$.



The average accuracy across all ranking functions and all values of attrition as the training window ranges for $N = 10,000$.



The performance of each ranking function as the number of friends of v_i ranges from 2 to 15 in G_i^R for $N = 1,000$, $\omega = 35$, and $d = 3$.



The performance of each ranking function as the number of friends of v_i ranges from 2 to 15 in G_i^R for $N = 10,000$, $\omega = 35$, and $d = 3$.

Weighted vs. Binary Results Tables

Attrition
10% Train

Binary	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.746	0.738	0.731	0.979	0.737	0.998	0.998	0.998	0.961	0.952
10	0.659	0.641	0.629	0.962	0.639	0.995	0.995	0.995	0.944	0.922
15	0.593	0.567	0.552	0.946	0.564	0.991	0.991	0.991	0.929	0.891
20	0.547	0.507	0.492	0.931	0.506	0.986	0.986	0.985	0.916	0.861
25	0.505	0.466	0.453	0.916	0.467	0.977	0.978	0.975	0.898	0.825
30	0.473	0.431	0.422	0.901	0.437	0.966	0.968	0.964	0.883	0.797
35	0.455	0.417	0.410	0.883	0.425	0.950	0.952	0.947	0.866	0.768
40	0.417	0.381	0.377	0.861	0.391	0.934	0.935	0.929	0.843	0.726
45	0.404	0.376	0.375	0.833	0.390	0.900	0.902	0.894	0.811	0.696
50	0.377	0.352	0.352	0.804	0.367	0.877	0.878	0.868	0.784	0.660
55	0.361	0.342	0.345	0.770	0.360	0.839	0.837	0.827	0.746	0.628
60	0.337	0.317	0.323	0.735	0.337	0.810	0.801	0.792	0.710	0.594
65	0.320	0.309	0.315	0.675	0.329	0.740	0.727	0.723	0.646	0.546
70	0.299	0.290	0.297	0.627	0.311	0.693	0.666	0.668	0.595	0.508
75	0.282	0.274	0.281	0.552	0.295	0.607	0.572	0.584	0.518	0.453
80	0.250	0.245	0.251	0.468	0.264	0.514	0.466	0.489	0.430	0.386
85	0.206	0.204	0.207	0.344	0.217	0.371	0.326	0.357	0.312	0.291
90	0.140	0.138	0.139	0.204	0.144	0.216	0.184	0.210	0.183	0.177
95	0.055	0.054	0.054	0.067	0.055	0.069	0.059	0.068	0.061	0.060

Weighted	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.714	0.636	0.622	0.984	0.628	0.998	0.998	0.998	0.972	0.943
10	0.618	0.535	0.521	0.970	0.528	0.995	0.995	0.995	0.957	0.904
15	0.575	0.487	0.472	0.955	0.481	0.991	0.992	0.991	0.942	0.872
20	0.522	0.442	0.429	0.940	0.439	0.984	0.985	0.984	0.926	0.838
25	0.484	0.410	0.399	0.923	0.410	0.977	0.978	0.975	0.909	0.807
30	0.457	0.383	0.374	0.904	0.386	0.966	0.968	0.964	0.893	0.783
35	0.438	0.380	0.374	0.887	0.387	0.949	0.950	0.945	0.872	0.757
40	0.408	0.354	0.349	0.863	0.363	0.934	0.936	0.929	0.850	0.725
45	0.394	0.346	0.344	0.836	0.359	0.911	0.913	0.904	0.824	0.702
50	0.375	0.337	0.340	0.805	0.355	0.877	0.878	0.868	0.788	0.669
55	0.351	0.318	0.321	0.774	0.336	0.850	0.846	0.836	0.756	0.635
60	0.335	0.310	0.315	0.730	0.331	0.804	0.795	0.787	0.709	0.598
65	0.317	0.297	0.304	0.682	0.320	0.751	0.737	0.731	0.658	0.561
70	0.296	0.279	0.289	0.631	0.303	0.698	0.670	0.670	0.601	0.517
75	0.282	0.268	0.276	0.564	0.291	0.621	0.582	0.595	0.532	0.470
80	0.252	0.241	0.249	0.475	0.262	0.520	0.471	0.496	0.440	0.398
85	0.211	0.202	0.207	0.357	0.216	0.385	0.335	0.370	0.325	0.304
90	0.144	0.139	0.141	0.213	0.146	0.225	0.190	0.218	0.192	0.186
95	0.052	0.051	0.051	0.064	0.052	0.066	0.057	0.065	0.059	0.058

Attrition
30% Train

Binary	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.760	0.758	0.758	0.999	0.760	1.000	1.000	1.000	0.972	0.974
10	0.687	0.681	0.682	0.995	0.685	1.000	1.000	1.000	0.963	0.961
15	0.637	0.623	0.623	0.989	0.628	0.999	0.999	0.999	0.957	0.949
20	0.608	0.582	0.581	0.980	0.587	0.998	0.998	0.999	0.952	0.935
25	0.572	0.543	0.538	0.970	0.545	0.997	0.997	0.997	0.943	0.914
30	0.549	0.514	0.508	0.958	0.516	0.994	0.995	0.995	0.934	0.895
35	0.528	0.491	0.482	0.945	0.493	0.990	0.991	0.991	0.924	0.871
40	0.483	0.443	0.434	0.925	0.444	0.984	0.985	0.985	0.905	0.830
45	0.472	0.433	0.424	0.909	0.435	0.971	0.971	0.972	0.888	0.802
50	0.435	0.395	0.385	0.880	0.398	0.960	0.959	0.959	0.864	0.757
55	0.414	0.378	0.369	0.854	0.382	0.940	0.936	0.938	0.835	0.722
60	0.380	0.343	0.335	0.819	0.349	0.922	0.910	0.914	0.800	0.675
65	0.368	0.337	0.332	0.777	0.346	0.876	0.857	0.866	0.752	0.630
70	0.347	0.318	0.315	0.736	0.329	0.839	0.804	0.820	0.702	0.587
75	0.337	0.313	0.311	0.675	0.325	0.766	0.720	0.746	0.636	0.543
80	0.308	0.290	0.289	0.596	0.302	0.675	0.610	0.649	0.548	0.476
85	0.272	0.261	0.261	0.468	0.272	0.516	0.454	0.500	0.423	0.386
90	0.200	0.194	0.195	0.299	0.202	0.320	0.274	0.312	0.267	0.256
95	0.084	0.083	0.083	0.103	0.085	0.107	0.092	0.105	0.094	0.093

Weighted	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.733	0.664	0.657	0.997	0.660	1.000	1.000	1.000	0.983	0.966
10	0.650	0.578	0.572	0.992	0.575	1.000	1.000	1.000	0.974	0.944
15	0.622	0.543	0.535	0.984	0.539	0.999	0.999	0.999	0.967	0.926
20	0.578	0.501	0.492	0.975	0.497	0.998	0.998	0.998	0.958	0.903
25	0.550	0.471	0.462	0.963	0.467	0.997	0.997	0.997	0.950	0.884
30	0.530	0.447	0.435	0.950	0.442	0.994	0.995	0.995	0.940	0.865
35	0.509	0.433	0.423	0.941	0.431	0.990	0.990	0.990	0.929	0.846
40	0.472	0.399	0.388	0.920	0.397	0.984	0.985	0.985	0.912	0.810
45	0.460	0.388	0.378	0.903	0.389	0.976	0.976	0.976	0.897	0.791
50	0.434	0.371	0.362	0.883	0.374	0.960	0.959	0.960	0.874	0.761
55	0.399	0.338	0.332	0.854	0.345	0.945	0.939	0.941	0.845	0.717
60	0.382	0.329	0.324	0.824	0.338	0.918	0.907	0.912	0.811	0.684
65	0.366	0.317	0.316	0.790	0.330	0.884	0.865	0.874	0.771	0.648
70	0.344	0.301	0.301	0.744	0.315	0.844	0.809	0.824	0.719	0.602
75	0.333	0.294	0.296	0.689	0.312	0.780	0.729	0.754	0.654	0.559
80	0.311	0.280	0.284	0.607	0.299	0.681	0.615	0.655	0.564	0.491
85	0.277	0.255	0.259	0.484	0.271	0.532	0.464	0.514	0.441	0.403
90	0.205	0.193	0.196	0.311	0.204	0.333	0.282	0.323	0.278	0.267
95	0.082	0.080	0.081	0.101	0.082	0.104	0.090	0.102	0.092	0.091

Attrition
50% Train

Binary	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.775	0.786	0.787	1.000	0.788	1.000	1.000	1.000	0.977	0.980
10	0.705	0.713	0.714	0.998	0.716	1.000	1.000	1.000	0.969	0.969
15	0.652	0.655	0.653	0.993	0.656	0.999	1.000	1.000	0.962	0.957
20	0.624	0.615	0.613	0.986	0.617	0.998	0.999	0.999	0.957	0.945
25	0.587	0.569	0.564	0.975	0.570	0.996	0.997	0.997	0.947	0.927
30	0.562	0.535	0.529	0.963	0.536	0.994	0.996	0.996	0.937	0.910
35	0.542	0.509	0.500	0.949	0.508	0.989	0.991	0.993	0.928	0.885
40	0.495	0.455	0.445	0.927	0.454	0.984	0.986	0.988	0.908	0.846
45	0.479	0.436	0.425	0.910	0.436	0.973	0.975	0.978	0.891	0.818
50	0.440	0.397	0.385	0.880	0.396	0.963	0.962	0.968	0.864	0.770
55	0.417	0.376	0.365	0.853	0.377	0.944	0.940	0.949	0.836	0.731
60	0.381	0.338	0.327	0.817	0.339	0.931	0.916	0.931	0.801	0.684
65	0.367	0.332	0.323	0.780	0.335	0.890	0.868	0.889	0.756	0.639
70	0.345	0.312	0.304	0.743	0.318	0.859	0.818	0.849	0.710	0.598
75	0.337	0.311	0.304	0.690	0.318	0.792	0.740	0.780	0.649	0.555
80	0.314	0.292	0.287	0.617	0.301	0.706	0.635	0.687	0.565	0.492
85	0.288	0.276	0.274	0.499	0.285	0.554	0.488	0.540	0.449	0.413
90	0.221	0.216	0.215	0.328	0.223	0.351	0.302	0.344	0.292	0.282
95	0.096	0.095	0.095	0.116	0.097	0.119	0.104	0.118	0.106	0.105

Weighted	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.740	0.673	0.666	0.998	0.670	1.000	1.000	1.000	0.984	0.969
10	0.661	0.594	0.587	0.994	0.590	1.000	1.000	1.000	0.977	0.949
15	0.635	0.561	0.552	0.987	0.556	0.999	1.000	1.000	0.971	0.935
20	0.591	0.516	0.508	0.978	0.511	0.998	0.999	0.999	0.962	0.915
25	0.567	0.489	0.480	0.967	0.484	0.997	0.998	0.998	0.955	0.897
30	0.545	0.462	0.449	0.953	0.455	0.994	0.995	0.996	0.944	0.877
35	0.527	0.448	0.437	0.944	0.443	0.990	0.992	0.993	0.934	0.862
40	0.489	0.410	0.397	0.922	0.405	0.984	0.986	0.988	0.917	0.827
45	0.476	0.397	0.385	0.904	0.394	0.978	0.978	0.983	0.902	0.806
50	0.446	0.373	0.362	0.886	0.372	0.964	0.963	0.970	0.881	0.777
55	0.405	0.336	0.328	0.857	0.339	0.951	0.944	0.954	0.851	0.731
60	0.385	0.322	0.314	0.827	0.327	0.927	0.915	0.929	0.817	0.696
65	0.369	0.312	0.307	0.795	0.320	0.898	0.875	0.896	0.779	0.658
70	0.347	0.295	0.291	0.752	0.305	0.863	0.821	0.852	0.729	0.610
75	0.337	0.292	0.292	0.704	0.307	0.803	0.745	0.786	0.670	0.574
80	0.318	0.282	0.284	0.629	0.298	0.712	0.641	0.693	0.585	0.511
85	0.293	0.267	0.269	0.515	0.282	0.568	0.497	0.553	0.469	0.430
90	0.227	0.213	0.215	0.341	0.223	0.364	0.311	0.357	0.306	0.295
95	0.095	0.092	0.093	0.114	0.094	0.117	0.102	0.116	0.104	0.104

Attrition
70% Train

Binary	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.769	0.768	0.769	0.997	0.772	0.999	0.999	1.000	0.972	0.974
10	0.692	0.687	0.687	0.991	0.692	0.998	0.999	0.999	0.961	0.958
15	0.633	0.622	0.621	0.980	0.627	0.996	0.997	0.998	0.950	0.941
20	0.604	0.580	0.576	0.967	0.583	0.993	0.995	0.996	0.940	0.921
25	0.561	0.530	0.522	0.950	0.530	0.988	0.991	0.993	0.926	0.897
30	0.533	0.496	0.486	0.932	0.495	0.982	0.986	0.989	0.911	0.874
35	0.510	0.468	0.455	0.913	0.466	0.972	0.976	0.982	0.896	0.844
40	0.457	0.413	0.400	0.887	0.410	0.964	0.965	0.973	0.871	0.797
45	0.439	0.396	0.382	0.869	0.394	0.945	0.947	0.958	0.852	0.769
50	0.399	0.358	0.345	0.837	0.356	0.931	0.927	0.943	0.820	0.720
55	0.372	0.336	0.324	0.807	0.335	0.906	0.899	0.917	0.786	0.679
60	0.342	0.304	0.293	0.774	0.305	0.890	0.867	0.894	0.750	0.636
65	0.333	0.304	0.294	0.736	0.306	0.845	0.817	0.849	0.707	0.600
70	0.315	0.292	0.282	0.702	0.295	0.811	0.764	0.806	0.662	0.563
75	0.314	0.296	0.288	0.651	0.301	0.742	0.688	0.736	0.604	0.528
80	0.297	0.283	0.278	0.581	0.290	0.657	0.590	0.645	0.529	0.473
85	0.276	0.269	0.266	0.466	0.277	0.510	0.452	0.502	0.419	0.395
90	0.214	0.211	0.210	0.303	0.217	0.320	0.280	0.316	0.273	0.268
95	0.089	0.089	0.089	0.104	0.091	0.106	0.095	0.106	0.097	0.097

Weighted	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.732	0.665	0.657	0.996	0.662	0.999	1.000	1.000	0.982	0.964
10	0.647	0.578	0.571	0.987	0.574	0.998	0.999	0.999	0.972	0.940
15	0.618	0.541	0.532	0.975	0.537	0.996	0.997	0.998	0.962	0.919
20	0.572	0.494	0.484	0.962	0.490	0.992	0.995	0.996	0.950	0.895
25	0.543	0.461	0.449	0.945	0.455	0.988	0.991	0.993	0.937	0.871
30	0.522	0.433	0.420	0.927	0.428	0.983	0.986	0.989	0.924	0.849
35	0.494	0.410	0.398	0.913	0.406	0.973	0.977	0.982	0.908	0.826
40	0.455	0.374	0.362	0.887	0.372	0.965	0.966	0.974	0.887	0.790
45	0.438	0.358	0.347	0.867	0.358	0.952	0.952	0.963	0.868	0.766
50	0.409	0.336	0.326	0.845	0.338	0.932	0.929	0.944	0.840	0.734
55	0.371	0.305	0.296	0.813	0.308	0.916	0.904	0.925	0.807	0.690
60	0.356	0.295	0.289	0.784	0.302	0.887	0.868	0.895	0.773	0.658
65	0.340	0.287	0.283	0.749	0.296	0.853	0.823	0.856	0.731	0.622
70	0.322	0.275	0.272	0.709	0.286	0.814	0.766	0.809	0.683	0.583
75	0.319	0.278	0.277	0.662	0.291	0.752	0.693	0.743	0.628	0.548
80	0.303	0.273	0.273	0.590	0.286	0.662	0.595	0.650	0.547	0.492
85	0.285	0.264	0.266	0.484	0.277	0.526	0.463	0.518	0.441	0.414
90	0.221	0.210	0.212	0.317	0.220	0.334	0.288	0.330	0.287	0.280
95	0.090	0.088	0.089	0.104	0.090	0.106	0.095	0.106	0.097	0.097

Attrition
90% Train

Binary	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.746	0.757	0.745	0.958	0.752	0.987	0.991	0.992	0.951	0.939
10	0.644	0.641	0.621	0.927	0.631	0.978	0.982	0.986	0.918	0.892
15	0.563	0.548	0.525	0.900	0.537	0.968	0.971	0.978	0.889	0.846
20	0.509	0.479	0.454	0.876	0.468	0.956	0.958	0.969	0.863	0.805
25	0.453	0.420	0.397	0.852	0.410	0.941	0.940	0.954	0.837	0.761
30	0.414	0.378	0.358	0.829	0.371	0.924	0.920	0.938	0.812	0.728
35	0.392	0.357	0.340	0.808	0.352	0.899	0.896	0.917	0.788	0.695
40	0.344	0.314	0.298	0.778	0.310	0.880	0.869	0.894	0.754	0.651
45	0.335	0.311	0.298	0.756	0.310	0.847	0.836	0.863	0.729	0.631
50	0.307	0.290	0.278	0.727	0.289	0.824	0.804	0.834	0.697	0.596
55	0.294	0.282	0.273	0.695	0.284	0.785	0.765	0.794	0.662	0.570
60	0.277	0.267	0.259	0.667	0.269	0.761	0.727	0.765	0.629	0.542
65	0.281	0.276	0.270	0.625	0.281	0.703	0.670	0.707	0.585	0.518
70	0.273	0.271	0.265	0.589	0.277	0.661	0.618	0.658	0.544	0.492
75	0.275	0.275	0.270	0.532	0.282	0.586	0.544	0.584	0.489	0.456
80	0.263	0.264	0.262	0.463	0.273	0.503	0.454	0.497	0.421	0.402
85	0.234	0.237	0.236	0.351	0.245	0.371	0.332	0.367	0.321	0.315
90	0.168	0.170	0.170	0.213	0.175	0.219	0.196	0.218	0.197	0.197
95	0.063	0.063	0.063	0.068	0.064	0.069	0.064	0.069	0.066	0.066

Weighted	PD	AD	ED	ET	ES	INT	HD	GM	BCS	OLY
5	0.712	0.624	0.610	0.965	0.620	0.990	0.993	0.994	0.962	0.929
10	0.607	0.515	0.502	0.934	0.512	0.979	0.983	0.987	0.933	0.876
15	0.554	0.458	0.442	0.911	0.454	0.969	0.973	0.980	0.912	0.840
20	0.497	0.404	0.391	0.887	0.403	0.954	0.957	0.966	0.886	0.797
25	0.453	0.365	0.352	0.861	0.364	0.941	0.940	0.955	0.860	0.764
30	0.419	0.331	0.319	0.836	0.331	0.926	0.921	0.940	0.837	0.732
35	0.397	0.321	0.312	0.815	0.325	0.901	0.896	0.918	0.812	0.708
40	0.358	0.290	0.281	0.784	0.294	0.883	0.871	0.896	0.780	0.669
45	0.345	0.287	0.280	0.762	0.293	0.858	0.844	0.872	0.754	0.650
50	0.329	0.279	0.274	0.734	0.287	0.824	0.808	0.838	0.721	0.624
55	0.307	0.263	0.260	0.704	0.272	0.799	0.772	0.806	0.687	0.591
60	0.300	0.264	0.263	0.670	0.275	0.755	0.726	0.762	0.648	0.568
65	0.296	0.267	0.267	0.634	0.279	0.712	0.678	0.714	0.609	0.541
70	0.285	0.261	0.262	0.595	0.274	0.666	0.623	0.663	0.564	0.512
75	0.285	0.264	0.267	0.542	0.279	0.599	0.552	0.594	0.510	0.475
80	0.271	0.257	0.261	0.468	0.272	0.508	0.460	0.502	0.434	0.416
85	0.242	0.235	0.237	0.363	0.246	0.384	0.344	0.381	0.336	0.329
90	0.177	0.174	0.176	0.225	0.181	0.233	0.207	0.231	0.210	0.209
95	0.062	0.062	0.062	0.068	0.063	0.069	0.064	0.068	0.065	0.065

Result Tables for all tests with N = 100,000

Similar results observed with N = 1,000 and N = 10,000; available upon request.

All average results for the Percent Difference ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.888	0.775	0.662	0.559	0.465	0.381	0.307
10	0.844	0.697	0.569	0.459	0.364	0.283	0.217
15	0.802	0.621	0.478	0.363	0.271	0.2	0.146
20	0.776	0.581	0.436	0.323	0.237	0.171	0.122
25	0.755	0.546	0.396	0.285	0.203	0.143	0.099
30	0.742	0.528	0.376	0.265	0.185	0.128	0.088
35	0.703	0.482	0.331	0.225	0.153	0.103	0.068
40	0.669	0.442	0.292	0.192	0.126	0.082	0.053
45	0.65	0.426	0.278	0.181	0.117	0.075	0.048
50	0.614	0.396	0.251	0.16	0.102	0.063	0.04
55	0.574	0.367	0.229	0.145	0.09	0.055	0.035
60	0.538	0.349	0.219	0.137	0.084	0.052	0.032
65	0.497	0.331	0.21	0.131	0.082	0.05	0.03
70	0.466	0.334	0.225	0.148	0.097	0.062	0.04
75	0.431	0.341	0.249	0.177	0.121	0.083	0.056
80	0.38	0.339	0.277	0.217	0.166	0.123	0.091
85	0.308	0.322	0.302	0.269	0.232	0.195	0.163
90	0.201	0.247	0.268	0.277	0.275	0.266	0.254
95	0.07	0.098	0.122	0.143	0.159	0.175	0.186

Attrition	9	10	11	12	13	14	15
5	0.244	0.195	0.159	0.125	0.104	0.092	0.079
10	0.165	0.127	0.097	0.078	0.063	0.054	0.047
15	0.106	0.076	0.056	0.042	0.033	0.025	0.022
20	0.088	0.062	0.045	0.033	0.025	0.02	0.015
25	0.068	0.047	0.034	0.025	0.018	0.014	0.01
30	0.06	0.041	0.029	0.019	0.014	0.01	0.008
35	0.047	0.03	0.021	0.014	0.01	0.007	0.005
40	0.035	0.022	0.015	0.009	0.007	0.004	0.003
45	0.03	0.019	0.012	0.008	0.005	0.003	0.002
50	0.025	0.015	0.01	0.006	0.004	0.003	0.002
55	0.021	0.013	0.008	0.005	0.003	0.002	0.001
60	0.02	0.013	0.007	0.005	0.003	0.002	0.001
65	0.018	0.011	0.007	0.004	0.003	0.002	0.001
70	0.025	0.016	0.011	0.006	0.004	0.003	0.002
75	0.038	0.025	0.017	0.011	0.008	0.005	0.004
80	0.067	0.049	0.036	0.026	0.019	0.013	0.011
85	0.131	0.108	0.089	0.072	0.057	0.044	0.037
90	0.236	0.22	0.204	0.189	0.172	0.157	0.144
95	0.199	0.206	0.215	0.221	0.226	0.235	0.236

All average results for the Absolute Difference ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.889	0.733	0.594	0.475	0.378	0.297	0.233
10	0.839	0.645	0.493	0.375	0.286	0.215	0.164
15	0.79	0.567	0.405	0.286	0.202	0.142	0.101
20	0.753	0.517	0.356	0.245	0.17	0.118	0.082
25	0.722	0.476	0.315	0.208	0.138	0.092	0.062
30	0.699	0.453	0.292	0.188	0.122	0.08	0.053
35	0.651	0.405	0.251	0.156	0.098	0.063	0.041
40	0.611	0.369	0.223	0.135	0.082	0.051	0.032
45	0.586	0.352	0.21	0.126	0.076	0.047	0.028
50	0.547	0.325	0.191	0.113	0.068	0.041	0.025
55	0.507	0.303	0.178	0.106	0.064	0.038	0.024
60	0.473	0.29	0.174	0.104	0.063	0.038	0.023
65	0.438	0.28	0.173	0.106	0.065	0.039	0.024
70	0.415	0.287	0.191	0.124	0.08	0.051	0.032
75	0.389	0.3	0.218	0.153	0.107	0.073	0.05
80	0.348	0.305	0.25	0.194	0.15	0.112	0.083
85	0.289	0.298	0.282	0.249	0.216	0.183	0.153
90	0.193	0.235	0.258	0.265	0.264	0.255	0.245
95	0.069	0.096	0.12	0.141	0.158	0.173	0.185

Attrition	9	10	11	12	13	14	15
5	0.184	0.144	0.117	0.093	0.077	0.07	0.064
10	0.124	0.096	0.075	0.063	0.053	0.048	0.044
15	0.073	0.052	0.039	0.031	0.025	0.021	0.018
20	0.059	0.043	0.031	0.024	0.019	0.016	0.014
25	0.042	0.03	0.021	0.016	0.012	0.01	0.009
30	0.036	0.025	0.018	0.013	0.01	0.008	0.006
35	0.028	0.019	0.012	0.009	0.007	0.005	0.004
40	0.021	0.013	0.009	0.006	0.004	0.003	0.002
45	0.018	0.011	0.007	0.005	0.003	0.002	0.002
50	0.016	0.009	0.006	0.004	0.003	0.002	0.001
55	0.014	0.009	0.006	0.004	0.002	0.002	0.001
60	0.014	0.009	0.005	0.003	0.002	0.001	0.001
65	0.015	0.01	0.006	0.004	0.002	0.002	0.001
70	0.02	0.013	0.009	0.005	0.004	0.002	0.002
75	0.033	0.023	0.015	0.01	0.007	0.005	0.003
80	0.061	0.044	0.033	0.023	0.016	0.012	0.009
85	0.123	0.1	0.082	0.065	0.051	0.04	0.032
90	0.227	0.212	0.193	0.177	0.159	0.141	0.128
95	0.196	0.204	0.212	0.218	0.221	0.229	0.229

All average results for the Euclidean Distance ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.885	0.731	0.596	0.478	0.383	0.302	0.238
10	0.833	0.643	0.496	0.379	0.291	0.22	0.168
15	0.783	0.565	0.408	0.292	0.208	0.147	0.106
20	0.743	0.513	0.357	0.249	0.173	0.122	0.086
25	0.71	0.471	0.314	0.209	0.141	0.094	0.064
30	0.684	0.446	0.29	0.188	0.123	0.082	0.055
35	0.634	0.397	0.248	0.155	0.098	0.064	0.042
40	0.594	0.362	0.22	0.134	0.083	0.051	0.033
45	0.568	0.344	0.206	0.125	0.076	0.047	0.029
50	0.529	0.316	0.186	0.111	0.068	0.041	0.025
55	0.489	0.294	0.173	0.104	0.063	0.038	0.024
60	0.457	0.283	0.17	0.102	0.062	0.038	0.023
65	0.426	0.275	0.17	0.104	0.064	0.039	0.024
70	0.405	0.283	0.189	0.122	0.08	0.051	0.032
75	0.382	0.299	0.218	0.153	0.107	0.073	0.05
80	0.343	0.305	0.252	0.197	0.152	0.114	0.085
85	0.287	0.299	0.284	0.254	0.221	0.187	0.156
90	0.193	0.236	0.259	0.268	0.268	0.26	0.251
95	0.069	0.096	0.121	0.142	0.159	0.176	0.188

Attrition	9	10	11	12	13	14	15
5	0.187	0.147	0.118	0.094	0.077	0.073	0.068
10	0.128	0.098	0.078	0.065	0.055	0.051	0.046
15	0.077	0.056	0.042	0.034	0.027	0.022	0.021
20	0.062	0.046	0.034	0.026	0.021	0.018	0.016
25	0.044	0.031	0.023	0.017	0.014	0.011	0.01
30	0.038	0.026	0.019	0.014	0.01	0.008	0.007
35	0.029	0.019	0.013	0.01	0.008	0.006	0.005
40	0.022	0.014	0.01	0.007	0.005	0.003	0.003
45	0.018	0.012	0.008	0.005	0.004	0.002	0.002
50	0.016	0.01	0.007	0.004	0.003	0.002	0.001
55	0.015	0.01	0.006	0.004	0.003	0.002	0.001
60	0.014	0.009	0.006	0.004	0.002	0.002	0.001
65	0.015	0.01	0.006	0.004	0.002	0.002	0.001
70	0.02	0.013	0.009	0.005	0.004	0.003	0.002
75	0.034	0.023	0.016	0.011	0.007	0.005	0.003
80	0.062	0.045	0.033	0.024	0.017	0.012	0.01
85	0.126	0.102	0.083	0.067	0.053	0.041	0.034
90	0.232	0.216	0.197	0.18	0.163	0.146	0.132
95	0.2	0.207	0.215	0.221	0.225	0.232	0.233

All average results for the Euclidean Threshold ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.998	1	1	1	1	1	1
10	0.993	1	1	1	1	1	1
15	0.983	1	1	1	1	1	1
20	0.967	0.999	1	1	1	1	1
25	0.948	0.998	1	1	1	1	1
30	0.925	0.995	0.999	1	1	1	1
35	0.896	0.99	0.999	1	1	1	1
40	0.866	0.982	0.997	1	1	1	1
45	0.838	0.968	0.995	1	1	1	1
50	0.801	0.946	0.99	0.999	1	1	1
55	0.762	0.916	0.981	0.997	1	1	1
60	0.724	0.88	0.966	0.994	0.999	1	1
65	0.679	0.832	0.941	0.985	0.997	0.999	1
70	0.636	0.776	0.901	0.968	0.991	0.997	0.999
75	0.579	0.712	0.844	0.933	0.973	0.989	0.995
80	0.499	0.628	0.762	0.867	0.931	0.962	0.978
85	0.383	0.507	0.63	0.739	0.822	0.876	0.913
90	0.234	0.327	0.418	0.504	0.582	0.648	0.704
95	0.075	0.111	0.146	0.182	0.214	0.249	0.278

Attrition	9	10	11	12	13	14	15
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1
45	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1
55	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1
65	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1
75	0.997	0.999	0.999	1	1	1	1
80	0.987	0.992	0.995	0.997	0.998	0.999	0.999
85	0.937	0.954	0.965	0.974	0.98	0.984	0.987
90	0.747	0.785	0.815	0.841	0.861	0.879	0.896
95	0.311	0.338	0.367	0.392	0.414	0.441	0.462

All average results for the Euclidean Sum ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.886	0.734	0.6	0.482	0.388	0.307	0.241
10	0.834	0.648	0.502	0.385	0.296	0.225	0.172
15	0.784	0.571	0.415	0.298	0.213	0.151	0.11
20	0.745	0.521	0.365	0.256	0.179	0.126	0.089
25	0.713	0.48	0.324	0.217	0.147	0.099	0.068
30	0.688	0.457	0.301	0.197	0.13	0.087	0.058
35	0.64	0.409	0.259	0.164	0.105	0.068	0.045
40	0.601	0.375	0.232	0.143	0.089	0.056	0.036
45	0.577	0.359	0.22	0.135	0.083	0.051	0.031
50	0.539	0.333	0.201	0.122	0.075	0.046	0.028
55	0.5	0.312	0.189	0.115	0.071	0.043	0.027
60	0.469	0.303	0.188	0.115	0.071	0.044	0.027
65	0.437	0.295	0.189	0.119	0.075	0.046	0.029
70	0.416	0.305	0.211	0.141	0.094	0.061	0.039
75	0.392	0.32	0.243	0.177	0.126	0.088	0.061
80	0.351	0.325	0.278	0.224	0.177	0.135	0.102
85	0.292	0.314	0.307	0.283	0.251	0.216	0.183
90	0.195	0.244	0.273	0.288	0.293	0.287	0.279
95	0.069	0.098	0.123	0.146	0.164	0.183	0.196

Attrition	9	10	11	12	13	14	15
5	0.19	0.15	0.121	0.096	0.079	0.075	0.069
10	0.13	0.1	0.08	0.066	0.056	0.052	0.047
15	0.079	0.058	0.043	0.035	0.028	0.023	0.022
20	0.064	0.047	0.035	0.027	0.022	0.019	0.017
25	0.047	0.033	0.024	0.018	0.014	0.011	0.01
30	0.04	0.028	0.02	0.015	0.011	0.009	0.007
35	0.031	0.021	0.014	0.01	0.008	0.006	0.005
40	0.024	0.015	0.01	0.007	0.005	0.003	0.003
45	0.02	0.013	0.008	0.006	0.004	0.003	0.002
50	0.018	0.011	0.007	0.005	0.003	0.003	0.002
55	0.017	0.011	0.007	0.005	0.003	0.002	0.002
60	0.016	0.01	0.007	0.004	0.003	0.002	0.001
65	0.018	0.012	0.007	0.005	0.003	0.002	0.001
70	0.025	0.016	0.011	0.007	0.005	0.004	0.002
75	0.042	0.029	0.02	0.014	0.009	0.007	0.004
80	0.076	0.056	0.042	0.03	0.023	0.016	0.013
85	0.15	0.124	0.102	0.084	0.066	0.053	0.044
90	0.262	0.245	0.226	0.209	0.192	0.174	0.159
95	0.209	0.217	0.227	0.234	0.239	0.247	0.249

All average results for the Intersection ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	0.998	1	1	1	1	1	1
20	0.997	1	1	1	1	1	1
25	0.993	1	1	1	1	1	1
30	0.989	1	1	1	1	1	1
35	0.982	0.999	1	1	1	1	1
40	0.972	0.998	1	1	1	1	1
45	0.957	0.996	1	1	1	1	1
50	0.938	0.992	0.999	1	1	1	1
55	0.915	0.985	0.998	1	1	1	1
60	0.883	0.973	0.994	0.999	1	1	1
65	0.836	0.95	0.986	0.996	0.998	0.999	1
70	0.778	0.914	0.969	0.988	0.995	0.998	0.999
75	0.696	0.853	0.933	0.969	0.985	0.992	0.996
80	0.588	0.753	0.858	0.917	0.951	0.97	0.981
85	0.436	0.587	0.705	0.79	0.85	0.89	0.919
90	0.255	0.358	0.452	0.535	0.605	0.663	0.714
95	0.079	0.115	0.151	0.186	0.218	0.253	0.282

Attrition	9	10	11	12	13	14	15
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1
45	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1
55	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1
65	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1
75	0.998	0.999	0.999	1	1	1	1
80	0.988	0.992	0.995	0.997	0.998	0.999	0.999
85	0.94	0.955	0.966	0.975	0.98	0.984	0.988
90	0.754	0.789	0.818	0.843	0.863	0.88	0.896
95	0.314	0.34	0.368	0.393	0.416	0.442	0.463

All average results for the Hamming Distance ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	0.999	1	1	1	1	1	1
20	0.998	1	1	1	1	1	1
25	0.995	1	1	1	1	1	1
30	0.992	1	1	1	1	1	1
35	0.985	0.999	1	1	1	1	1
40	0.975	0.998	1	1	1	1	1
45	0.959	0.995	0.999	1	1	1	1
50	0.937	0.99	0.998	1	1	1	1
55	0.906	0.979	0.995	0.999	1	1	1
60	0.863	0.96	0.988	0.996	0.999	1	1
65	0.801	0.923	0.97	0.988	0.995	0.998	0.999
70	0.73	0.868	0.936	0.968	0.984	0.992	0.996
75	0.638	0.784	0.872	0.923	0.952	0.97	0.981
80	0.525	0.667	0.766	0.833	0.88	0.912	0.936
85	0.385	0.506	0.601	0.675	0.734	0.779	0.816
90	0.226	0.308	0.377	0.437	0.488	0.532	0.574
95	0.072	0.103	0.132	0.158	0.18	0.203	0.22

Attrition	9	10	11	12	13	14	15
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1
45	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1
55	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1
65	1	1	1	1	1	1	1
70	0.998	0.999	0.999	1	1	1	1
75	0.988	0.993	0.996	0.997	0.998	0.999	0.999
80	0.953	0.965	0.975	0.981	0.986	0.989	0.992
85	0.846	0.872	0.892	0.91	0.925	0.935	0.946
90	0.609	0.642	0.67	0.699	0.72	0.741	0.764
95	0.24	0.256	0.271	0.287	0.298	0.314	0.327

All average results for the Graph Matching ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	0.999	1	1	1	1	1	1
20	0.998	1	1	1	1	1	1
25	0.996	1	1	1	1	1	1
30	0.993	1	1	1	1	1	1
35	0.988	0.999	1	1	1	1	1
40	0.979	0.999	1	1	1	1	1
45	0.966	0.997	1	1	1	1	1
50	0.947	0.993	0.999	1	1	1	1
55	0.921	0.987	0.998	1	1	1	1
60	0.884	0.976	0.995	0.999	1	1	1
65	0.828	0.951	0.986	0.996	0.999	0.999	1
70	0.76	0.913	0.969	0.988	0.995	0.998	0.999
75	0.67	0.848	0.932	0.969	0.985	0.992	0.996
80	0.556	0.743	0.855	0.917	0.951	0.97	0.981
85	0.411	0.578	0.703	0.789	0.85	0.89	0.919
90	0.242	0.353	0.451	0.535	0.605	0.664	0.714
95	0.076	0.114	0.15	0.186	0.219	0.253	0.283

Attrition	9	10	11	12	13	14	15
5	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1
45	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1
55	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1
65	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1
75	0.998	0.999	0.999	1	1	1	1
80	0.988	0.992	0.995	0.997	0.998	0.999	0.999
85	0.94	0.955	0.966	0.975	0.98	0.984	0.988
90	0.754	0.789	0.818	0.843	0.863	0.88	0.896
95	0.314	0.341	0.369	0.394	0.416	0.442	0.463

All average results for the BCS ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.983	0.979	0.977	0.976	0.977	0.978	0.979
10	0.974	0.974	0.973	0.974	0.975	0.976	0.978
15	0.962	0.967	0.968	0.97	0.972	0.974	0.976
20	0.95	0.966	0.968	0.971	0.973	0.975	0.977
25	0.936	0.963	0.967	0.97	0.973	0.975	0.977
30	0.921	0.962	0.967	0.971	0.973	0.977	0.978
35	0.897	0.955	0.965	0.97	0.973	0.976	0.978
40	0.87	0.946	0.962	0.968	0.972	0.975	0.977
45	0.846	0.935	0.959	0.967	0.971	0.975	0.977
50	0.811	0.916	0.953	0.965	0.97	0.974	0.977
55	0.771	0.889	0.944	0.962	0.969	0.972	0.975
60	0.727	0.854	0.928	0.956	0.966	0.97	0.974
65	0.671	0.805	0.9	0.943	0.959	0.966	0.97
70	0.616	0.745	0.856	0.919	0.946	0.958	0.965
75	0.548	0.673	0.79	0.872	0.915	0.937	0.95
80	0.462	0.58	0.696	0.789	0.849	0.886	0.91
85	0.352	0.459	0.56	0.649	0.719	0.769	0.806
90	0.216	0.296	0.368	0.435	0.496	0.547	0.591
95	0.072	0.104	0.134	0.163	0.189	0.216	0.239

Attrition	9	10	11	12	13	14	15
5	0.979	0.981	0.982	0.981	0.981	0.981	0.981
10	0.979	0.98	0.981	0.981	0.982	0.982	0.983
15	0.978	0.979	0.98	0.981	0.983	0.982	0.983
20	0.98	0.981	0.981	0.983	0.983	0.983	0.983
25	0.98	0.981	0.981	0.983	0.983	0.983	0.983
30	0.98	0.981	0.983	0.983	0.984	0.983	0.984
35	0.98	0.982	0.982	0.983	0.983	0.984	0.984
40	0.98	0.981	0.982	0.982	0.984	0.984	0.984
45	0.979	0.98	0.981	0.982	0.983	0.984	0.984
50	0.978	0.98	0.981	0.982	0.982	0.983	0.983
55	0.977	0.979	0.98	0.98	0.981	0.982	0.982
60	0.975	0.978	0.978	0.979	0.98	0.981	0.981
65	0.973	0.974	0.976	0.977	0.978	0.978	0.978
70	0.969	0.971	0.973	0.975	0.975	0.977	0.976
75	0.959	0.963	0.967	0.968	0.97	0.971	0.973
80	0.926	0.935	0.945	0.951	0.956	0.958	0.963
85	0.834	0.854	0.874	0.887	0.899	0.909	0.918
90	0.628	0.659	0.684	0.711	0.727	0.746	0.767
95	0.264	0.284	0.305	0.325	0.343	0.363	0.377

All average results for the Olympic ranking function for N = 100,000 as the number of friends of v_i ranges from 2 to 15.

Attrition	2	3	4	5	6	7	8
5	0.984	0.979	0.973	0.966	0.958	0.948	0.937
10	0.973	0.968	0.959	0.948	0.936	0.922	0.908
15	0.958	0.95	0.939	0.924	0.907	0.887	0.868
20	0.943	0.938	0.923	0.905	0.884	0.862	0.841
25	0.927	0.925	0.908	0.888	0.867	0.843	0.821
30	0.91	0.914	0.896	0.875	0.852	0.828	0.805
35	0.88	0.888	0.868	0.843	0.817	0.789	0.763
40	0.849	0.862	0.838	0.811	0.781	0.75	0.72
45	0.823	0.84	0.818	0.79	0.761	0.73	0.702
50	0.784	0.806	0.785	0.755	0.724	0.692	0.661
55	0.741	0.769	0.752	0.723	0.691	0.659	0.627
60	0.696	0.731	0.722	0.695	0.664	0.632	0.6
65	0.64	0.682	0.683	0.659	0.63	0.597	0.565
70	0.59	0.641	0.656	0.646	0.624	0.594	0.564
75	0.529	0.595	0.628	0.635	0.622	0.602	0.579
80	0.449	0.528	0.582	0.61	0.618	0.611	0.598
85	0.346	0.436	0.506	0.556	0.589	0.608	0.617
90	0.215	0.29	0.354	0.41	0.458	0.497	0.532
95	0.072	0.103	0.133	0.161	0.186	0.212	0.234

Attrition	9	10	11	12	13	14	15
5	0.928	0.92	0.913	0.902	0.899	0.897	0.894
10	0.893	0.88	0.87	0.862	0.855	0.85	0.85
15	0.848	0.833	0.818	0.806	0.8	0.792	0.787
20	0.821	0.801	0.788	0.774	0.766	0.758	0.755
25	0.799	0.779	0.763	0.751	0.74	0.73	0.726
30	0.784	0.763	0.746	0.733	0.723	0.716	0.711
35	0.738	0.714	0.697	0.682	0.672	0.658	0.655
40	0.696	0.67	0.65	0.633	0.62	0.609	0.602
45	0.674	0.652	0.628	0.614	0.6	0.587	0.579
50	0.633	0.605	0.584	0.567	0.553	0.54	0.534
55	0.597	0.574	0.551	0.534	0.519	0.509	0.499
60	0.573	0.548	0.526	0.512	0.497	0.48	0.473
65	0.535	0.509	0.489	0.468	0.452	0.435	0.428
70	0.537	0.511	0.492	0.475	0.459	0.448	0.432
75	0.553	0.532	0.514	0.497	0.482	0.465	0.456
80	0.584	0.567	0.553	0.54	0.528	0.516	0.508
85	0.62	0.618	0.617	0.615	0.607	0.603	0.595
90	0.557	0.581	0.597	0.615	0.628	0.64	0.655
95	0.258	0.278	0.298	0.318	0.336	0.356	0.371

Vita

Denise Koessler Gosnell was born in Pittsburgh, PA to parents Henry Joseph Koessler and Susan Margaret Koessler. She moved to Knoxville, TN at a young age where she attended Sacred Heart Cathedral School and later graduated from Farragut High School in 2004. She attended The College of Wooster and served as the captain of the college's swim team during both her junior and senior years. In May of 2008, she graduated Magna Cum Laude with a Bachelor of Arts degree in Mathematics and a license to teach Mathematics in the state of Ohio. She also received minors in Spanish and Education. Her undergraduate research explored elliptic curve cryptography. After graduation, Denise returned to Tennessee where she began a National Science Foundation Graduate STEM Fellowship through East Tennessee State University. She graduated in 2010 with a Master of Science in Mathematics. Her master's thesis created a predictive model for secondary RNA structures using graph theory and neural networks.

Denise returned in Knoxville in 2010 to pursue a doctoral degree in computer science at the University of Tennessee. She received research funding from the National Science Foundation Integrative Graduate Education and Research Traineeship. She also completed research as a Data Analyst with Link Analytics. In 2013, Denise received an Outstanding Teaching Assistant Award for the Department of Electrical Engineering and Computer Science. In 2014, Denise was featured as a "The Data Analyst: Tennessees Graduate Student to Watch" in the university's flagship magazine *The Torchbearer*. Throughout her doctoral studies, her research on Social Fingerprinting lead to presentations from San Diego to London. She received her Doctorate in Philosophy in Computer Science from the department of Electrical Engineering and Computer Science in August of 2014. After graduation, she accepted a position as a Data Scientist with PokitDok, a start-up company in Charleston, SC.