

# SOcial Fingerprint Analysis Software

## A Software Model for Identifying Users of Social Networks by their Data Footprint

Denise K. Gosnell  
The University of Tennessee  
Center for Intelligent Systems  
and Machine Learning  
Knoxville, TN  
Denise@utk.edu

Michael T. Adams  
The University of Tennessee  
Center for Intelligent Systems  
and Machine Learning  
Knoxville, TN  
madams44@utk.edu

Michael W. Berry  
The University of Tennessee  
Center for Intelligent Systems  
and Machine Learning  
Knoxville, TN  
mberry@eecs.utk.edu

### ABSTRACT

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 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 by providing a vehicle for large-scale dynamic social network analysis. The primary objectives in the first iteration of the SOFA software are as follows: (1) to apply well-understood properties from published models in the construction process, (2) to create scalable software that can reasonably model up to 10 million users, and (3) to create a dynamic multi-edge graph that can easily be altered to study other social phenomena.

### Categories and Subject Descriptors

G.1 [Discrete mathematics]: Graph theory  
; J.3 [Collaborative and social computing]: Collaborative and social computing design and evaluation methods  
; F.5 [Design and analysis of algorithms]: Graph algorithms analysis

### Keywords

social fingerprint, social network analysis, software, statistics, data mining

## 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 fingerprint are all that are required for accurate human identification [4]. 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 [31], 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 [18, 13, 23, 33]. 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 fingerprint" and how much data is needed?

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, the software environment developed in this paper aims to fill the void for future work in academia by providing a simulated environment in which researchers can test the performance and scalability of cutting edge technologies on dynamic social network data.

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 by providing a vehicle for large-scale dynamic social network analysis. The primary objectives in the first iteration of the SOFA software are as follows: (1) to apply well-understood properties from published models in the construction process, (2) to create scalable software that can reasonably model up to 10 million users, and (3) to create a dynamic multi-edge graph that can easily be altered to study other social phenomena. Section 2 describes the current technologies for social network analysis. Section 3 details each step of the construction process and presents our methods of validation. Section 4 presents the statistics on

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD '14 New York, New York USA

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

the scalability and accuracy of our model with a discussion in Section 5. Section 6 concludes with a discussion of the features for a future version of this software.

## 2. 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 [19, 21, 9, 20, 26]. Specifically, the authors of [9] 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 [20] applies an understanding of information diffusion to calculate market share. The simulation built in [26] 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 [25] 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 [15] 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 [15].

To complement the growing creation of synthetic models, other efforts provide analysis into the constructs of real social network data [3, 2, 16, 29]. 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 [6] 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 [8]. 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 lays 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 com-

bined \$5.1 *billion* on market analytics and advertising in 2010 [24]. To address this need, a quickly emerging field of private companies are creating custom software solutions to address the demand for accurate business analytics [1, 7, 10]. One of the leading competitors in this space, IBM Research, introduced the Social Media Networks Modeling and Simulation (SMSim) software in late 2013 [7, 14]. The SMSim software enables businesses to explore the impact of social media events through simulation. According to [7], 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, the authors of [7] 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 [7].

While the behavior of mobile users presented in [6, 8] 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 [26, 20, 25], 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. Section 4 demonstrates that the software created for this paper 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.

## 3. THE SOFA SOFTWARE

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 3.1. 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  $\omega$ . For reference purposes, Table 1 contains a listing of all symbols and terminology used throughout Section 3.

Section 3.1 describes the initialization and construction of the vertices in the graph. Section 3.2 details the construction of the dynamic relationships from time points  $t_1, t_2 \dots t_n$ . Section 3.3 outlines the distribution of weight throughout the edges of the graph and Section 3.4 presents the underlying algorithm used by the SOFA software followed by a description of validation procedures in Section 3.5.

**Table 1: A listing of all symbols and terminology used throughout the remaining sections and subsequent chapters of this paper.**

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 $i$
$\delta$	Minimum degree of $V(G)$
$\Delta$	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 $v_i$ : $N(v_i) = \{v_j \in V(G^t)   e(i, j)_b^t \in E(G^t)\}$
$\alpha$	Controls the steepness of the power-law distribution
$\beta$	Controls the tail of the power-law distribution
$t$	Time
$n$	Number of time steps
$G^t$	Graph at time $t$
$\omega$	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 $w$

### 3.1 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:  $N, \alpha, \beta, \delta$ , and  $\Delta$ . 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  $\alpha$  and  $\beta$  control the steepness and tail, respectively, of the scale-free connectivity between the nodes (see Equation 1). An example of a scale-free distribution is displayed in Figure 1. The endpoints of the connectivity distribution are  $\delta$  and  $\Delta$ , where  $\delta$  represents the minimum degree and  $\Delta$  represents the maximum degree within the base model.

It is well understood that the distribution of edges in a social network is scale-free [5, 27, 11]. For this model, we implement the power-law distribution defined in Equation 1. This scale-free distribution establishes a maximum potential degree for each vertex where the user is able to control the input parameters  $\alpha$  and  $\beta$ . Each vertex  $v_i$  in the graph receives a maximum potential degree where the probability of award is given by Equation 2. That is,

$$p(x) = \frac{\alpha \cdot x^{-\beta}}{AUC}, \quad (1)$$

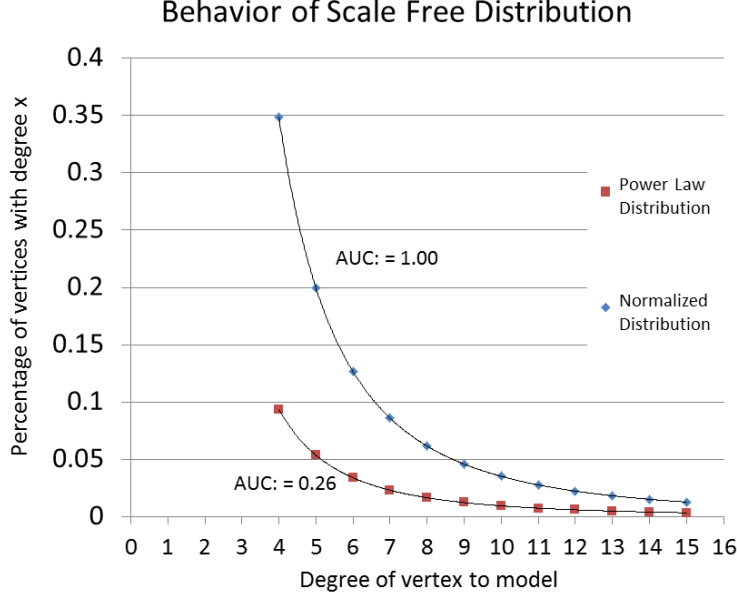
where  $AUC$  stands for *area under the curve* and is calculated

as

$$AUC = \sum_{i=\delta}^{\Delta} \alpha \cdot i^{-\beta}. \quad (2)$$

As seen in Equation 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 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.

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 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 3.3 details how a user’s activity level is diffused throughout the network.



**Figure 1:** An example of the effect of normalization on the power law distribution shown in Equation 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 0.26. The resulting normalized curve has an AUC of 1.00.

### 3.2 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  $e(i, j)_{b,w}^t$  represent an edge between vertices  $v_i$  and  $v_j$  during time step  $t$  of type  $b$  with weight  $w$ . Section 3.3 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  $\omega$  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 [6]. 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  $t$ .

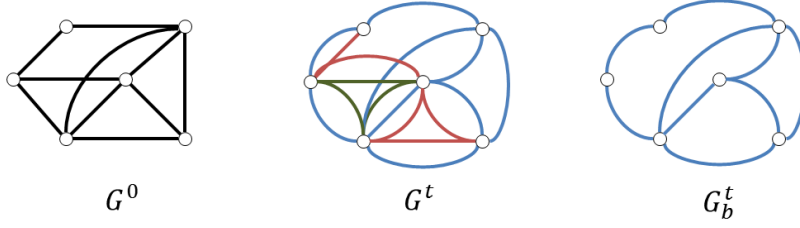
As outlined later in Algorithm 2, the edges are built in the simulation after all of the vertices are awarded a maximum degree potential. As such, an edge is created between two vertices if and only if each vertex has remaining degree potential. To be considered for a relationship, two vertices are drawn 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 [30] 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 stochas-

tic elements contribute to the observation of a relationship over time; Figure 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,  $e(i, j)_{0,1}^0 \in E(G^0)$ . 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  $b$ , then an edge of type  $b$  with weight 1 is added to the base graph  $G^0$ . That is,  $e(i, j)_{b,1}^0 \in E(G^0)$ . 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.

Given the information from the base graph  $G^0$ , the global attrition rate  $\omega$  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 on in social networking that require a dynamic graph should mine the observable graph between time steps  $[1, t]$  which are represented in graphs  $\{G^1, G^2, \dots, G^t\}$ . The base graph  $G^0$  serves as a baseline constructor since it represents  $\{G^1 \cup G^2 \cup \dots \cup G^t\}$ .

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



**Figure 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.**

diffusion process in Section 3.3 and examine the observable relationships between  $G^1, G^2, \dots, G^t$ .

### 3.3 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  $c_{i,b}^t$  onto the edges  $e(i, j)_{b,1}^t \in N(v_i)$  of  $v_i$  during time  $t$  for behavior  $b$ . An optimal flow would reduce the remaining capacity  $r_{i,b}^t$  of each vertex  $v_i$  to zero for each time step  $t$  and each behavior  $b$ . We believe this to be a novel instance of the max-flow problem and the algorithm presented herein is a first solution. Definition 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 1.** Given a graph  $G(V, E)$  where each vertex  $v_i \in V(G)$  has capacity  $c_{i,b}$ , there are  $m$  commodities  $b_1, b_2, \dots, b_m$ , 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)

The **value of diffusion** is  $r_{i,b}$  and the **value of flow** is  $|f| = \sum_{v_j \in N(v_i)} w(e(i, j))$ . The **maximal diffusion problem**

is to maximize  $|f|$  such that  $\sum_{v_j \in V(G)} r_{i,b} \rightarrow 0$ .

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.

**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  $c_{i,b}^t \leq c_{i+1,b}^t$ ;

**for each**  $v_i \in V(G)_{sorted}$  **do**

    Initialize:  $r_{i,b}^t = c_{i,b}^t$ ;

    Determine total community capacity  $C$  of  $v_i$ ;

$$C = \sum_{v_j \in N(v_i)} c_{j,b}^t;$$

    where  $N(v_i) = \{v_j \in V(G^t) | e(i, j)_b^t \in E(G^t)\}$ ;

**for each**  $v_j \in N(v_i)$  **do**

$$w = c_{i,b}^t \cdot \frac{c_{j,b}^t}{C};$$

$$e(i, j)_b^t = w;$$

$$c_{j,b}^t = c_{j,b}^t - w;$$

$$r_{i,b}^t = r_{i,b}^t - w;$$

**end**

    Sort all nodes starting from  $v_i$  such that  $c_{i,b}^t \leq c_{i+1,b}^t$ ;

**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.

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 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, 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 is 34.66, which is 20% of the original total capacity of the vertices in the network.

### 3.4 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 2 lists each input parameter in the algorithm.

The design logic of our implementation differs from existing open source models, such as [25, 30], in that we are primarily interested in the degree distribution of the 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 consideration permits us to create models which more closely 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 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.

### 3.5 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 implement a least-squares regression [12] to fit the observed degree distribution

**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**  $\text{roll}() > l_b$  **then**  
 Create  $e(i, j)$  in the base graph  $G^0$ ;  
 newEdge = true;  
**for** each time step  $t$  to model **do**  
**if**  $\text{roll}() > \omega$  **then**  
 Create  $e(i, j)$  in the subgraph  $G^t$   
**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.

to a function of the form:

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

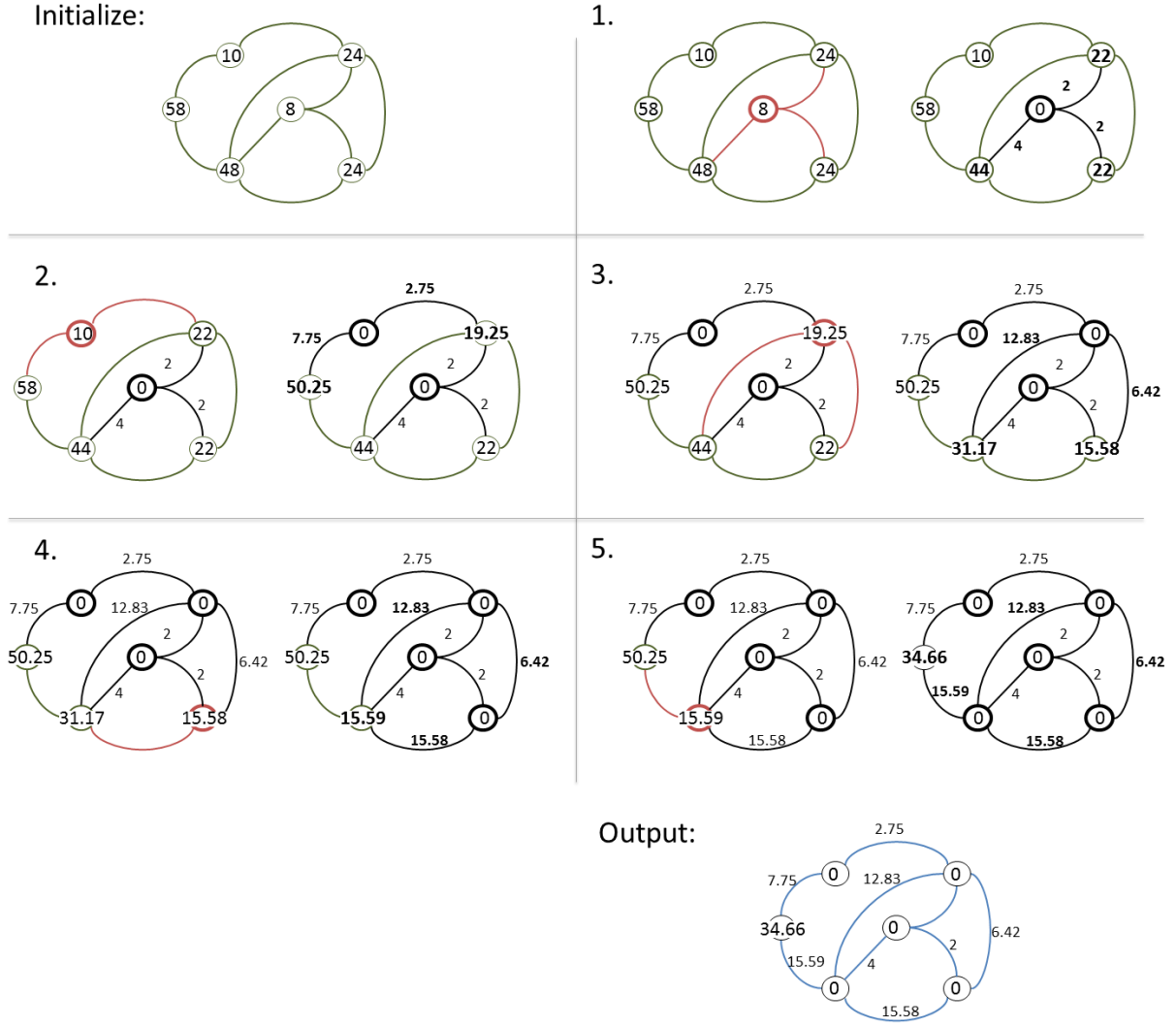
The least squares fitting yields:

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

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

where  $B = b$ ,  $A = e^a$  and  $n = \Delta - \delta + 1$  [32]. 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 4 showcases the range of  $R^2$  values observed over 5,000 trials.

Further, we implement the standard **Pearson's chi-squared test** or  $\chi^2$  to quantify the error in the model [12]:



**Figure 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.

**Table 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 [28].

Parameter	Usage	Default	Accepted
$N$	Number of vertices in the model	10000	[100, 10000000]
$\delta$	Minimum degree of $V(G)$	2	[2, $N - 2$ ]
$\Delta$	Maximum degree of $V(G)$	$\delta + 1$	[ $\delta + 1$ , $N - 1$ ]
$\alpha$	Controls the steepness of the degree distribution	1.8	[1, 4]
$\beta$	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]
$\omega$	Attrition of an edge from $t$ to $t + 1$	37	[0, 99]

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

where the observed degree distribution in the base graph  $G^0$  is compared against the user's input distribution for  $\alpha$  and  $\beta$ . Our hypothesis is that there is no significant difference between constructed distribution and the requested distribution. Section 4 contains the range of  $\chi^2$  values observed over 5,000 trials.

## 4. 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. A complete tables of all results is available in [17]. We discuss all results in Section 5.

### 4.1 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 4 and 5 illustrate the memory required to construct a graph with no edge weights. Figure 6 shows the memory required to construct a graph with edge weights when using 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. As confirmed in the tables given in [17], 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 [22].

### 4.2 Run Time

Next, we examine 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. Figures 7 and 8 illustrate the run time required to construct a graph with no edge weights. Figure 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 [17] for exact values of run times summarized in Figures 7, 8 and 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 [22].

Noting the significant increase in time from Figure 7 to Figure 9, we implemented further tests to determine the

bottleneck in our implementation of Algorithm 1. Table 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 shows the overall percentage of run time spent on the diffusion process.

### 4.3 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 10 illustrates the average number of edges in the model as  $N$  varies from 10,000 to 1 million nodes. Figure 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 [17], 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.

### 4.4 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  $\alpha$  ranged from  $[1, 4]$  in increments of 0.1 and  $\beta$  ranged from  $[1, 4]$  in increments of 0.1. Each simulation for a given  $N$ ,  $\alpha$ , and  $\beta$  was 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  $m = 3$ .

#### 4.4.1 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 2 which dictates the maximum potential degree awarded to each vertex during the construction process. In this equation, the parameters  $\alpha$  and  $\beta$  are input parameters for the construction process. Then, as detailed in Section 3.5, the observed graph is fit to a power law to assess the accuracy of the model. Figure 12 shows the percent difference between the input  $\alpha$  and the observed  $\alpha$  across 24,025 different simulations. Figure 13 shows the percent difference between the input  $\beta$  and the observed  $\beta$  across 24,025 different simulations. The percent difference is calculated as

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

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

In Figures 12 and 13, the  $x$ -axis ranges across 31 different input values for  $\alpha$  and the  $y$ -axis ranges across various 31 different input values for  $\beta$ . 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



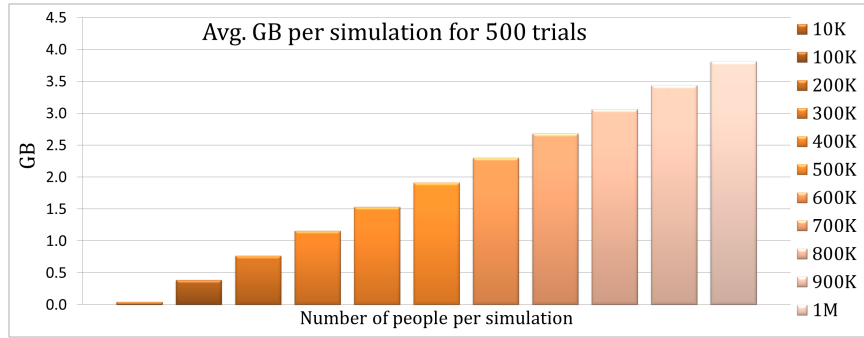


Figure 4: The amount of RAM required to execute a simulation of size  $N$  without edge weights as  $N$  ranges from 10,000 to 1 million. The total RAM required during execution was recorded for 500 trials for each simulation. The standard deviation for each simulation fell in the range of [0.0 MB, 3.5 MB]. Exact values are shown in [17].

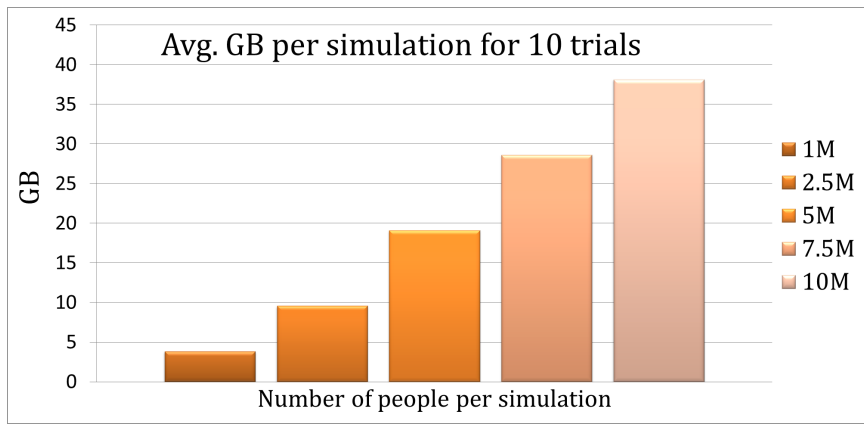


Figure 5: The amount of RAM required to execute a simulation of size  $N$  without edge weights as  $N$  ranges from 1 million to 10 million. Due resource constraints, the total RAM required during execution was recorded for only 10 trials for each simulation. The standard deviation for each simulation fell in the range of [0.06 MB, 7.9 MB]. Exact values are shown in [17].

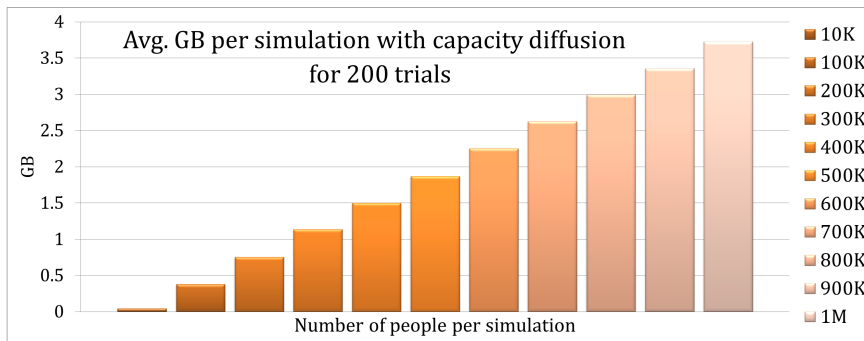


Figure 6: The amount of RAM required to execute a simulation of size  $N$  with edge weights as  $N$  ranges from 10,000 to 1 million. The total RAM required during execution was recorded for only 200 trials for each simulation. The standard deviation for each simulation fell in the range of [0.0 MB, 2.7 MB]. Exact values are shown in [17].

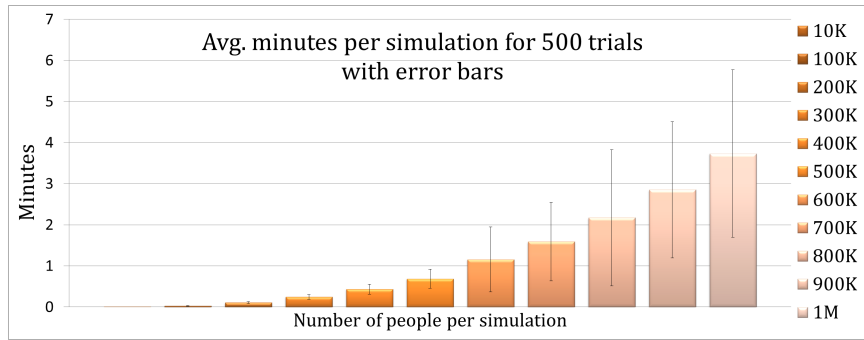


Figure 7: The amount of run time required to execute a simulation of size  $N$  without edge weights as  $N$  ranges from 10,000 to 1 million. The total time required during execution was recorded for 500 trials for each simulation. The standard deviation across all simulations is shown for each average with an error bar. Exact values are shown in [17].

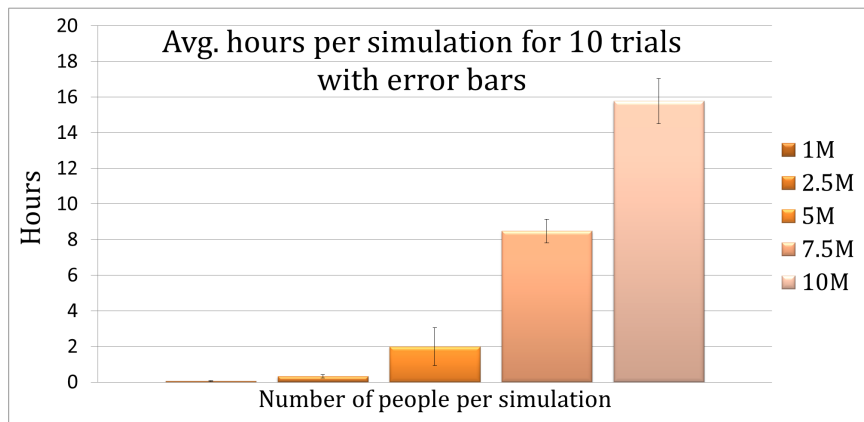


Figure 8: The amount of run time required to execute a simulation of size  $N$  without edge weights as  $N$  ranges from 1 million to 10 million. Due to constraints on computing resources, total time required during execution was recorded for only 10 trials for each simulation. The standard deviation across all simulations is shown for each average with an error bar. Exact values are shown in [17].

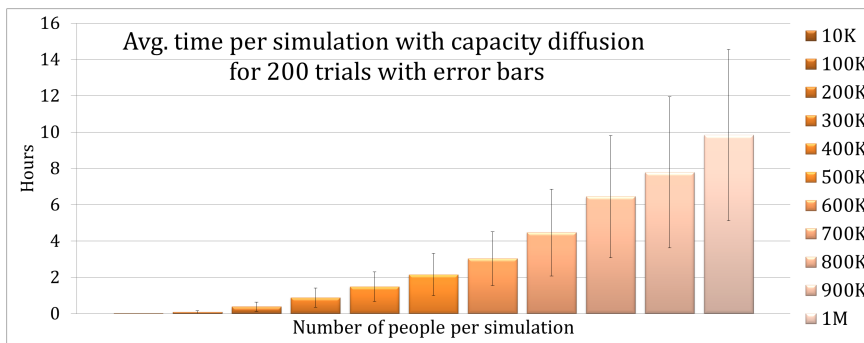


Figure 9: The amount of run time required to execute a simulation of size  $N$  with edge weights as  $N$  ranges from 10,000 to 1 million. Total time required during execution was recorded for 200 trials for each simulation. The standard deviation across all simulations is shown for each average with an error bar. Exact values are shown in [17].

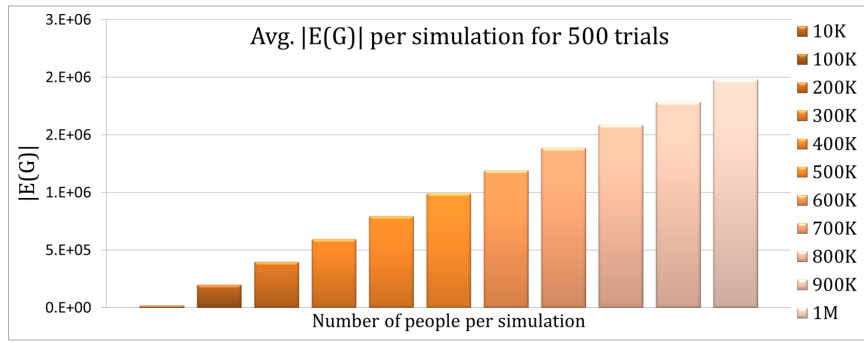


Figure 10: The average number of edges observed as  $N$  ranges from 10,000 to 1 million for 500 simulations.

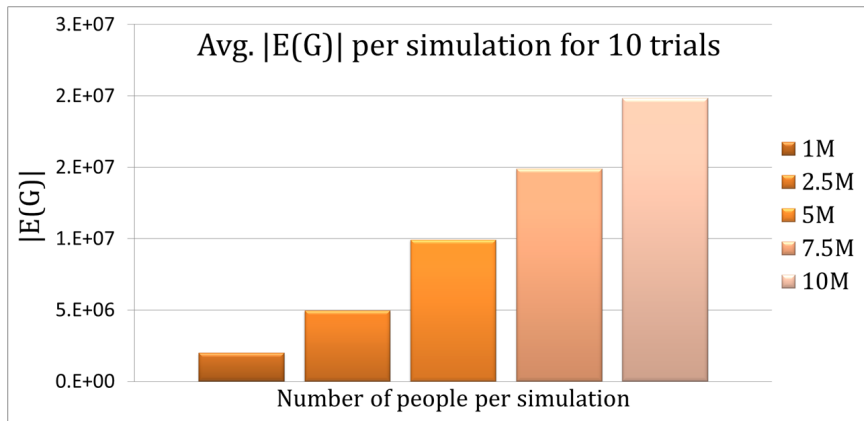


Figure 11: The average number of edges observed as  $N$  ranges from 10,000 to 1 million for 500 simulations.

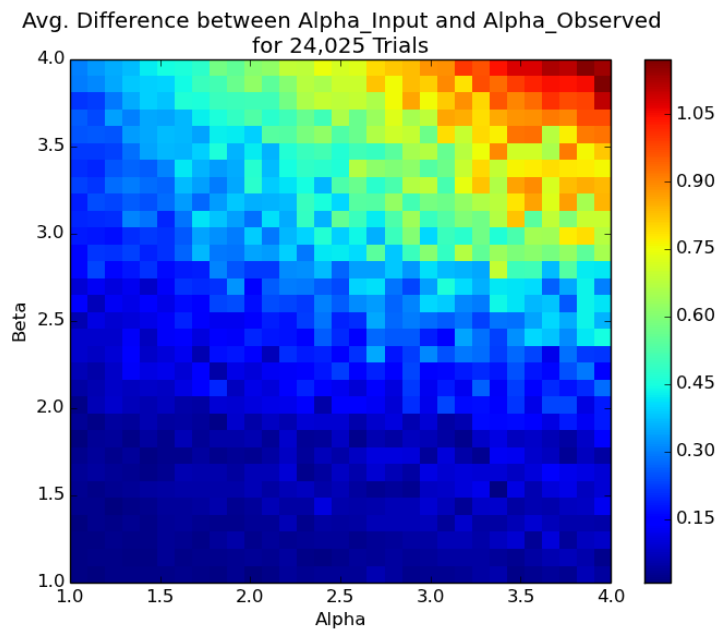
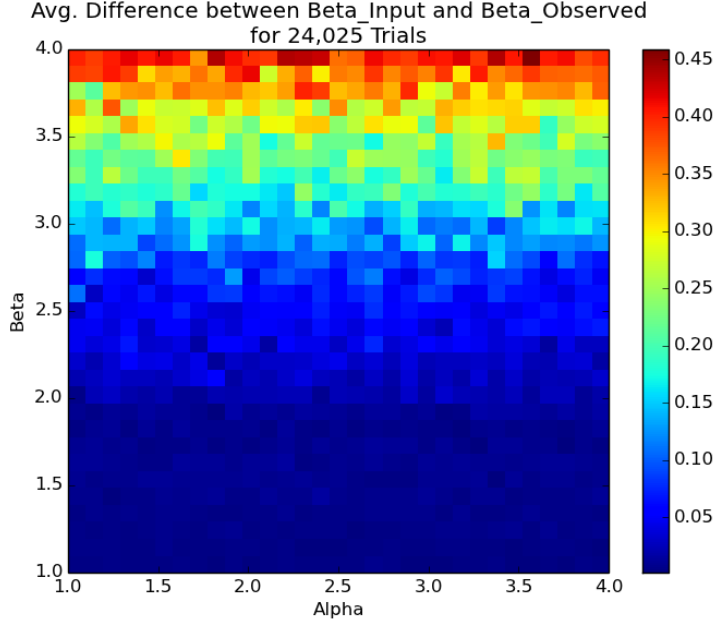


Figure 12: The percent difference between the input  $\alpha$  and the observed  $\alpha$  across 24,025 different simulations.

**Table 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



**Figure 13: The percent difference between the input  $\beta$  and the observed  $\beta$  across 24,025 different simulations.**

and shaded according to the scale on the right.

#### 4.4.2 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 3.5, we implement a least-squares regression [12] to fit the 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 14 shows the average R-squared error across 24,025 different simulations. In Figure 14, the  $x$ -axis ranges across 31 different input values for  $\alpha$  and the  $y$ -axis ranges across various 31 different input values for  $\beta$ . 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 14.

#### 4.4.3 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  $\chi$ -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  $\chi$ -squared error to quantify how confident we are that the hypothesis stated in Section 3.5 is represented in the data. Figure 15 displays the average  $\chi$ -squared error across 24,025 different simulations. In Figure 15, the  $x$ -axis ranges across 31 different input values for  $\alpha$  and the  $y$ -axis ranges across various 31 different input values for  $\beta$ . As such, there are a total of 961 different combinations of input parameters represented across this grid. The  $\chi$ -squared error between the observed distribution and the expected distribution for 5 trials is recorded and shaded according to the scale on the

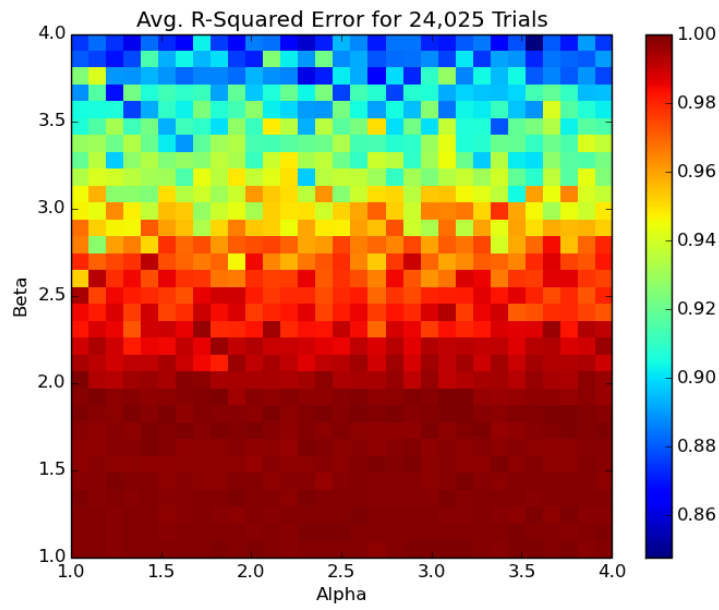


Figure 14: The average R-squared error across 24,025 different simulations.

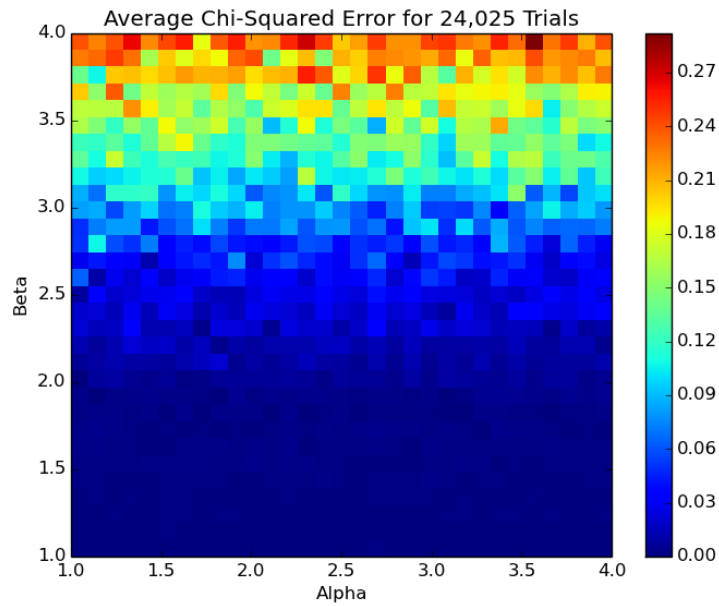


Figure 15: The average  $\chi$ -squared error across 24,025 different simulations.

right. There are  $\Delta - \delta - 1 = 45$  degrees of freedom in the simulations.

## 5. DISCUSSION

### 5.1 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 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 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. Upon identifying a large bottleneck in the algorithm’s time spent on sorting, we implemented a linked list data structure and a smart sort routine to observe 5 times the speed-up over the previous implementation. 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 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 4 and 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.

### 5.2 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 14 and 15. First, Figure 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 [17] 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 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 [17].

The only result which requires further exploration is the percent difference between the input alpha and observed alpha shown in Figure 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 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,  $\alpha$  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 [17]. These figures show that for models of  $N \geq 100,000$ , the observed difference in  $\alpha$  is consistent for the entire range of possible inputs. As such, Figures 12 and 13 illustrate the sensitivity of the input parameters  $\alpha$  and  $\beta$  for a range of models, but the sequence of heat maps in [17] demonstrates convergence for  $\alpha$  and  $\beta$  as  $N \rightarrow 100,000$ .

## 6. CONCLUSION

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 3.1, the edge weights were dependent on the activity level of each vertex in the graph and each vertex’s activity level was a stochastic assigned 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 [25]. 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 [29]. 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 3.3.

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 [17].

## 7. ACKNOWLEDGMENTS

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 paper 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 [28]. The operating system is Scientific Linux 5.5 and the batch-queue system is Grid Engine. All other cluster documentation is available at [28].

## 8. REFERENCES

- [1] Alteryx. Intuitive workflow for data blending and advanced analytics, 2014.
- [2] Y. Bachrach, M. Kosinski, T. Graepel, P. Kohli, and D. Stillwell. Personality and patterns of facebook usage. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 24–32. ACM, 2012.
- [3] F. Bonchi, C. Castillo, A. Gionis, and A. Jaimes. Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):22, 2011.
- [4] C. Champod, C. J. Lennard, P. Margot, and M. Stoilovic. *Fingerprints and other ridge skin impressions*. CRC press, 2004.
- [5] A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.
- [6] C. Cortes, D. Pregibon, and C. Volinsky. Computational methods for dynamic graphs. *Journal of Computational and Graphical Statistics*, 12(4), 2003.
- [7] M. A. de C Gatti, A. P. Appel, C. N. dos Santos, C. S. Pinhanez, P. R. Cavalin, and S. B. Neto. A simulation-based approach to analyze the information diffusion in microblogging online social network. *Proceedings of the 2013 Winter Simulation Conference*, 2013.
- [8] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013.
- [9] S. A. Delre, W. Jager, and M. A. Janssen. Diffusion dynamics in small-world networks with heterogeneous consumers. *Computational and Mathematical Organization Theory*, 13(2):185–202, 2007.
- [10] D. Detectives. The latest and most innovative use of analytics and big data to learn about—and drive—your business, 2014.
- [11] D. Doran, V. Mendiratta, C. Phadke, and H. Uzunalioglu. 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, 2012.
- [12] N. R. Draper and H. Smith. *Applied regression analysis 2nd ed.* New York New York John Wiley and Sons., 1981.
- [13] R. Gallagher. How anonymous cellphone location data leave “fingerprints” that could identify you. *Slate*, 2013.
- [14] M. Gatti, A. P. Appel, C. Pinhanez, C. dos Santos, D. Gribel, P. Cavalin, and S. B. Neto. Large-scale multi-agent-based modeling and simulation of microblogging-based online social network. *Proc. of Int. Work. on Multi-Agent-based Simul.*, 2013.
- [15] M. Gomez Rodriguez, J. Leskovec, and B. Schölkopf. 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, 2013.
- [16] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.
- [17] D. K. Gosnell. *Social Fingerprinting: Identifying Users of Social Networks by their Data Footprint*. PhD thesis, The University of Tennessee, 2014.
- [18] D. Gross. How your movements create a gps fingerprint. *CNN*, 2013.
- [19] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proceedings of the 13th international conference on World Wide Web*, pages 491–501. ACM, 2004.
- [20] M. A. Janssen and W. Jager. Simulating market dynamics: Interactions between consumer psychology and social networks. *Artificial Life*, 9(4):343–356, 2003.
- [21] D. Kempe, J. Kleinberg, and É. Tardos. 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, 2003.
- [22] D. Keppel. Time(1), 2000.
- [23] W. Knight. How access to location data could trample your privacy. *MIT Technology Review*, 2013.
- [24] P. Laya. Do you pay enough for advertising? one big corporation spent a jaw-dropping \$4.2 billion last year. *Business Insider*, 2011.
- [25] J. Leskovec. Snap: Stanford network analysis project, 2014.
- [26] D. Liu and X. Chen. 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, 2011.
- [27] A. A. Nanavati, R. Singh, D. Chakraborty, K. Dasgupta, S. Mukherjee, G. Das, S. Gurumurthy, and A. Joshi. Analyzing the structure and evolution of massive telecom graphs. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):703–718, 2008.
- [28] G. Ragghianti. Newton program documentation, 2014.
- [29] M. Seshadri, S. Machiraju, A. Sridharan, J. Bolot, C. Faloutsos, and J. Leskovec. 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, 2008.
- [30] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *Boost Graph Library: User Guide and Reference Manual*, The. Pearson Education, 2001.
- [31] A. Smith. Americans and their gadgets. *Pew Research*



*Center*, 2012.

- [32] E. W. Weisstein. Least squares fitting-exponential. *MathWorld-A Wolfram Web Resource.*, 2011.
- [33] L. Zyga. Study shows how easy it is to determine someone's identity with cell phone data. *Physics.org*, 2013.