# SYNTHETIC GENERATORS FOR SIMULATING SOCIAL NETWORKS

by

AWRAD MOHAMMED ALI
B.S. University of Mosul, 2005

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science in Computer Engineering
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2014

Major Professor: Gita Sukthankar

# ABSTRACT

An application area of increasing importance is creating agent-based simulations to model human societies. One component of developing these simulations is the ability to generate realistic human social networks. Online social networking websites, such as Facebook, Google+, and Twitter, have increased in popularity in the last decade. Despite the increase in online social networking tools and the importance of studying human behavior in these networks, collecting data directly from these networks is not always feasible due to privacy concerns. Previous work in this area has primarily been limited to 1) network generators that aim to duplicate a small subset of the original network's properties and 2) problem-specific generators for applications such as the evaluation of community detection algorithms.

In this thesis, we extended two synthetic network generators to enable them to duplicate the properties of a specific dataset. In the first generator, we consider feature similarity and label homophily among individuals when forming links. The second generator is designed to handle multiplex networks that contain different link types. We evaluate the performance of both generators on existing real-world social network datasets, as well as comparing our methods with a related synthetic network generator. In this thesis, we demonstrate that the proposed synthetic network generators are both time efficient and require only limited parameter optimization.

.

To my mother who always encouraged and believed in me. I wish you were here to thank for everything and to make you proud of me.

To my father who is the world for me. Thank you so much for always being a perfect dad.

To my husband and my wonderful kids (Dima and Yazen). Thank you for supporting me and making my dream comes true. You are more than what I deserve. Love you all so much.

# ACKNOWLEDGMENTS

I would like to express the deepest appreciation and thanks to my adviser, Professor Gita Sukthankar who supported me in every step working on this thesis. This work would not be possible without her guidance.

I would like to thank the higher committee for education development in Iraq who gave me the opportunity to do my masters in the United States.

I want to express my warm thanks to my friend Hector Lugo-Cordero who helped me a lot during my academic career and who always encouraged me to believe in myself and my abilities.

I would also like to thank my labmates and friends Hamidreza Alvari and Alireza Hajibagheri for their support and their help with completing this document.

I would also want to thank my wonderful friend Dr. Xi Wang whose research provided a foundation for this thesis.

I would like to thank Dr. Kiran Lakkaraju (Sandia National Labs) and Dr. Rolf Wigand (University of Arkansas) for providing the MMOG datasets used for the evaluation.

Last, but by no means least, I thank my labmates for being such a wonderful friends and family. I will miss you all.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1: INTRODUCTION

Online social networking applications, such as Facebook, Twitter, and YouTube, have rapidly increased in popularity, due to their ability to offer compelling visual communication platforms. Data from these services has enabled interdisciplinary researchers to study human behavior at an unprecedented scale. Yet even in this era of "big data", the research questions that we can address are often sharply constrained by data availability. Access to data is often limited for privacy reasons; this jeopardizes the reproducibility of experiments conducted by one research group on a privately held dataset. Synthetic network generators can provide a common benchmark allowing multiple groups to evaluate their research on the same dataset. They facilitate the rapid prototyping of network analysis software, by simplifying the process of testing the algorithms on a broad spectrum of networks. However, one question that often arises is how similar are synthetically-generated networks to the original networks extracted from social media data?

One modeling approach is to create a generator that reproduces properties commonly found in human networks, such as homophily, scale-free structure, and dyadic closure. However, the real data is often messy, possessing isolate nodes, unbalanced class distributions, and strange degree distributions. Thus network analysis algorithms which perform well on synthetically generated networks, may perform poorly when deployed in the actual application. In this thesis, we propose that the best approach to preserve both privacy and verisimilitude is to *clone* the original network. In this usage scenario, companies release limited statistics about the network characteristics, and the generator creates a network that matches as many of those statistics as possible. Although the adjacency matrix of the final network is significantly different from that of the original network, preserving the network statistics may lead to comparable performance of algorithms such as community detection and link prediction on both the original and synthetically generated datasets.

A second issue which commonly arises with publicly available datasets is that missing data elements restrict the applicability of the data and hinder the development of algorithms that are

substantially different from the original authors'. Many network datasets consist solely of the adjacency matrix for a single time slice. A standard procedure is to clean datasets in order to create a single, large connected component. However, it can be useful to consider the node features as well, which are often omitted from standard datasets; for instance, collective classification algorithms, such as ICA [24], make extensive use of the node features when predicting the labels of neighboring nodes. In reality, people are connected by *multiplex* networks, in which each link type represents a different form of social interaction (e.g., messaging, common interests, geographic neighborhood); unfortunately there is a dearth of publicly available multiplex datasets, making it difficult to evaluate algorithms that leverage different link types.

Hence synthetic network generators that can simulate human social networks serve as a valuable complement to the real social media datasets. In this thesis, we introduce a network generator that supports both the use of node-level features and different link types. Our generator uses preferential attachment and link homophily to select link targets, and stochastic optimization to tweak the feature distributions to match the original dataset. We envision three different usage cases for our generator:

- cloning privately-held datasets for debugging purposes;
- simulating realistic human populations within agent-based simulations;
- benchmarking collective classification, link prediction, and community detection algorithms on multiplex networks.

# CHAPTER 2: LITERATURE REVIEW

The synthetic network is commonly denoted by $G = (V, E)$ where $V$ and $E$ are sets of vertices (nodes) and edges respectively. Nodes often represent the entities in the environment, such as users in the social networks or computers in computer networks while the edges represent the interactions between them. These networks can be categorized into two types based on the direction of the links: directed and undirected networks. In the former, the edges have directions while in the latter, they have no direction (bidirectional edges). Nodes can have a vector of features that describe their properties. For example, in a dataset extracted from a massively multiplayer online game, nodes would represent players or their avatars, links would represent in-game message exchanges, and node features could be used to designate the avatar's combat or crafting skills.

The synthetic networks are often evaluated based on the number of features that they have in common with the real networks, among which the most important one is the power law distribution of node degrees, which exists in virtually all real networks, such as online social networks, WWW, and the Internet. The power law is described as follows:

$$n_k \propto k^{-a}, \text{where } a > 0 \tag{2.1}$$

Where $a$ is the power law exponent, and $n_k$ is the number of nodes with degree $k$. In networks formed by social causal reasons, in addition to possessing a power law degree distribution, nodes are more likely to interact with other nodes within the same community (group) than nodes outside that community. Travers and Milgram [33] also discovered that most real networks have a small diameter, meaning many nodes are only few hops away from each other.

## 2.1 Statistical Generators

Synthetic network generators can be broadly categorized as being statistical [11, 38] or agent-based [6, 4]. The statistical models focus on modeling one or more aspects of the network statistics, therefore they are good at preserving these properties from the original dataset. The generators in statistical field are subdivided into several submodels according to a survey by Chakrabarti and Faloutsos [7]. These models are as follows:

- **Random graph model** has attracted much research interest due to its simplicity. Two problems with this model are that it does not generate graphs with a power law degree distribution or community structure.

- **Preferential attachment model**, effectively models power law degree distributions and thus is one of the most popular models (e.g., the Barabási and Albert model [2]). The graphs in this model start with few nodes and continuously grow by adding both nodes and links to the current network structure. Nodes in these networks prefer to connect to higher degree nodes.

- **Optimization based models** are used to generate a network with limited resources. This assigns available resources to events having higher probability of being costly in the future. Graphs belonging to this model are able to follow the power law distribution.

- **Geographical models** consider the effects of the nodes' positions on the growth of the network. These types of networks are important for modeling power-grid networks or routers.

- **Internet-specific approaches** use concepts from the other models (e.g., preferential attachment and geographical) and adapt them for the Internet requirements.

One weakness is that these models may focus on a single graph property, while neglecting other patterns in the network structure.

## 2.2 Agent Based Modeling and Social Network Simulators

Agent based simulators depend on actions to form the networks. Thus, the resulting networks simulate the agent constraints and model the targeted behavior of the individual agent. Agent-based models tend to be very domain specific and are not easily modified for other problems; for instance Carley et al. [6] created an agent-based model for simulating urban disease spread after bioattacks. However Bernstein and O'Brien [4] proposed a model that can be used to generate many types of networks, including social, computer, human mobility, and cellular communications. Their model uses a rule-based system to govern the agents' actions. In some social networks, the agent decisions are affected by the environment. For example, in [12] the agent observes the behavior of its neighbors in each timestep and based on these observations, the agent decides which action maximizes its utility without possessing global information across the network.

## 2.3 Classical Synthetic Generators

The focus of this thesis is simulating human social networks (e.g., [40]), but there is also an extensive literature on constructing other types of networks including biological [27] and computer networks [31, 18]. First we review the classical models that have been used as the basis for many applications.

### *2.3.1 Erdös and Rényi (ER) Model*

The earliest work on graph generators was done by Erdös and Rényi in 1960 [11] which is commonly known as the *ER* model. The graphs in this model are generated randomly, i.e., the nodes follow a random distribution that governs the probability of connecting nodes. The network starts with *N* nodes. Each pair of nodes is independently connected by an edge with a probability *p* as shown in Figure 2.1. As a result a set of graphs $G_{N,p}$ will be formed with the same parameters *N*

and $p$. The degree distribution for this model follows a Poisson distribution as shown in Equation 2.2.

$$p_k = \binom{N}{k} p^k (1-p)^{N-k} \approx \frac{p(N-1)^k e^{-p(N-1)}}{k!} \tag{2.2}$$

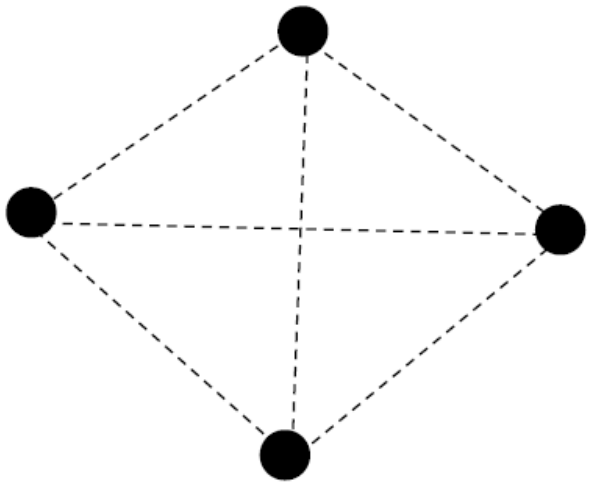

Figure 2.1: In the ER model, an edge is generated with uniform probability between every pair of nodes.

The generated graphs have small components with the same size and few edges when *p* has a small value. On the other hand, a high value of *p* causes the graphs to have a huge component of size *O(N)*. The diameter of this model grows slowly as the network grows since the diameter is concentrated around $\frac{\log N}{\log z}$, where $z$ is the average degree of the nodes in the graph. However, the simplicity of random graphs is not sufficient enough to be considered as viable alternative to real graphs since there are many drawbacks associated with this model as summarized in [7]:

- The degree distribution in random graphs follows a Poisson distribution which is very different from the power law distribution that occurs in real graphs.

- They do not exhibit community structure (i.e., groups that have dense intra-connections versus inter-connections) as opposed to many real graphs.
- The clustering coefficient *cc* depends on the number of nodes *N* in the graph, $cc = \frac{k}{N}$, while this is not true in many real graphs.

To make the generated graphs follow a power law of the degree distribution instead of following a Poisson distribution, several extensions have been proposed, such as the work by Aiello et al. in [1] and Palmer and Steffan [28].

### *2.3.2 Watts and Strogatz Model*

Human networks often exhibit small world characteristics, as illustrated by the famous six degrees of separation experiment [33]. Watts and Stogatz proposed a random graph generator for creating small world graphs with high clustering coefficients and small diameters [38]. The generated graphs consist of vertices that form a one dimensional lattice through connecting each vertex to its *k* nearest neighbors (where $k = 4$). The process of generating the network is initialized with a ring lattice that contains *N* nodes, where each node has k neighbors. Later, with probability *p*, a node and the edge incident to it are chosen from the network, where the chosen edge is reconnected to another node chosen randomly. During this process, self-connection and edge duplication is forbidden. The process is continued by performing one pass in which all the nodes are considered. Later, the second neighbors are considered, following the same process. This model is constructed with $0 < p < 1$. When $p = 0$, the clustering coefficient and the average distance between the nodes are very high which is implausible while when $p = 1$, both the clustering coefficient and the average distance are very low, which is also unrealistic. The resulting degree distribution is similar to that of random graphs and does not follow the power law distribution, making graphs generated with this model significantly different from many types of real-world graphs.

### *2.3.3 Barabási and Albert Model*

In 1999, Barabási and Albert proposed a model commonly known as the BA model [2] to generate graphs that follow the power law distribution for the node degrees. The network structure is formed using two processes:

- Gradually growing the network by adding nodes over time;
- Preferentially attaching the nodes to already existing nodes following the rich get richer phenomena as shown in Figure 2.2.

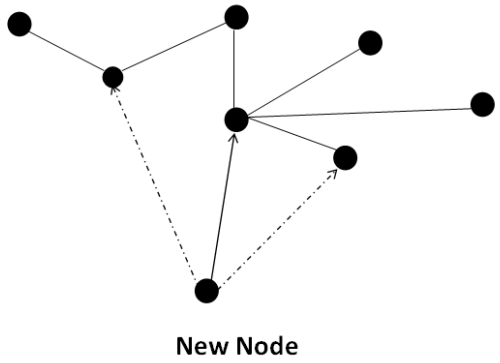


Figure 2.2: Preferential attachment. A new node prefers to attach to the higher degree node. The link is shown as a solid line, while the dashed lines show potential links that have not formed.

The network is initialized with $m_0$ nodes. In each timestep a new node is added with *m* edges where $m \leq m_0$. After *t* timesteps, the network has $t + m_0$ vertices and *mt* edges. The generated graphs organize themselves into a scale free network that reflects the distribution observed in real graphs whose properties are time independent. The scale free network exhibits real network properties due to several reasons that can be summarized as follows:

- The rapidly growing of the network insures that older nodes receive more links.

- People tend to link to other people (nodes) that are easier to reach [34]. This phenomena usually occurs in the crowded environments.

- The scale free graphs can tolerate the random behavior of up to 80% of the nodes.

In this model, the probability of choosing the sink node from the existing nodes can be described by:

$$P = \frac{k(v)}{\sum_i k(i)} \tag{2.3}$$

Where *k(i)* is the degree of node *i*. This formula specifies that the nodes with high degree are more likely to be chosen as sink nodes. Empirical results show that both gradual growing of the network and preferential attachment are important to have a stationary power law distribution [3]. The degree distribution of the *BA* model follows the power law distribution with a fixed exponent of 3 ($p_k = k^{-3}$). In addition, the distribution is independent of the number of edges added per new node. Although the *BA* model is able to follow the most important property of real graph (following the power law distribution), there are several drawbacks associated with it that make the model unsuitable as an alternative for real graphs. As stated in the Chakrabarti and Faloutsos survey [7], the *BA* model drawbacks can be summarized as follows:

- The power law exponent is fixed to 3 while in many real graphs this value can deviate.

- The model assumes linear preferential attachment while in many real graphs the attachment is nonlinear.

- *BA* model only generates undirected graphs while many real graphs are directed.

- In the *BA* model each node has a degree of at least one while in real graphs some nodes are isolated.

- In many real graphs, the network is formed by both adding/removing nodes and links.

- The community structure is absent in the *BA* model.

The ideas of preferentially attaching nodes and gradually growing the network makes the *BA* model the basis of many later works.

## 2.4 Recent Graph Generators

One weakness common to all the classical models is that they lack explicit mechanisms for creating community structures within the graphs, which are often present in graphs extracted from social media data. Here we discuss the recent work in synthetic network generators that addresses this issue.

### *2.4.1 R-MAT graph generator*

Chakrabarti et al. proposed a method based on matrix recursion known as *R-MAT* model [8]. *R-MAT* is considered an important graph generator since it captures many properties that are present in real graphs, such as having a power law degree distribution, small diameter, and community structure. In order to generate the network, the user needs to decide the number of nodes and the number of desired edges. The network starts with an empty matrix of size $N$ by $N$ where $N$ is the number of nodes. This matrix is then divided into four equal parts recursively. Later, at each timestep an edge is established in the matrix based on the probabilities in each partition (a,b,c, and d). After the partition is chosen, it is divided again into four equal partitions. The process will repeat until each square is of size 1 by 1, as shown in Figure 2.3. Two of the four initial partitions represent two different communities while the other two represent "cross-links" between the two communities. For example, one of the communities could be for people interested in football and the other for people interested in soccer. The other two partitions contain friends with separate interests (football and soccer). The sub-partitions that are created through recursion represent nested communities within the main communities, such as interest in a specific team.

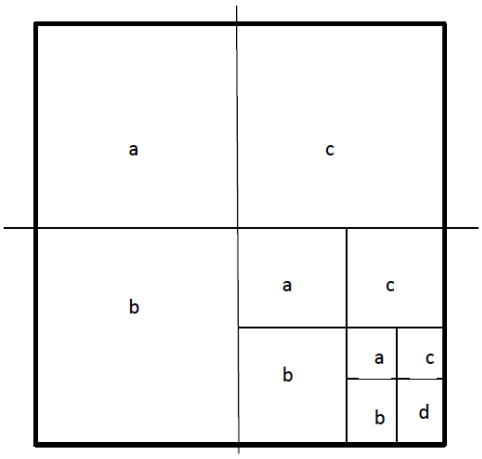


Figure 2.3: The R-MAT model. The matrix is divided into four equal partitions. According to a non-uniform probability distribution one of these partitions is chosen. This process will repeat until we have a one by one cell where we can place the edge.

### *2.4.2 Kronecker Graph Generator*

Following the recursion idea proposed in *R-MAT* [8], Leskovec et al. [21] proposed a graph model that depends on Kronecker multiplication. The main idea is to produce self-similar graphs by recursion. The network starts with an initial graph $G_1$ that has $N_1$ nodes and $E_1$ edges. By recursion, larger successive graphs $G_2$, $G_3 \ldots G_n$ are generated. The $k^{th}$ graph $G_k$ contains $N_k = N_1^k$ nodes. To have a densification power law, the edges in $G_k$ should be equal to $E_1^k$ [21]. In addition to the ability to have a densification power law distribution with densification exponent $k = \frac{\log(E1)}{\log(N1)}$, Kronecker multiplication produces graphs with fixed diameter. The graph generated by Kronecker can be described as follows:

$$G_k = G_1 \otimes G_1 \otimes G_1 \otimes \ldots G_1 \quad \text{for k times} \tag{2.4}$$

The process of generating the graphs is simple since it can be shown as that each community consists of smaller nested communities that are formed through expansion and recursion. The nodes within each community are connected to nodes in the same community and nodes in the other communities. The way that the graph is generated in this model introduces a staircase effect in the node degrees. To eliminate this problem Leskovec et al. [21] developed a stochastic version of this model through introducing a probabilistic matrix and replacing the ones and zeros in the initial graph shown in Figure 2.4 with probability values $\alpha$ and $\beta$ respectively where $\beta \leq \alpha$. This approach introduces random graphs that have the properties of the Kronecker deterministic version (described earlier) with limited staircase.

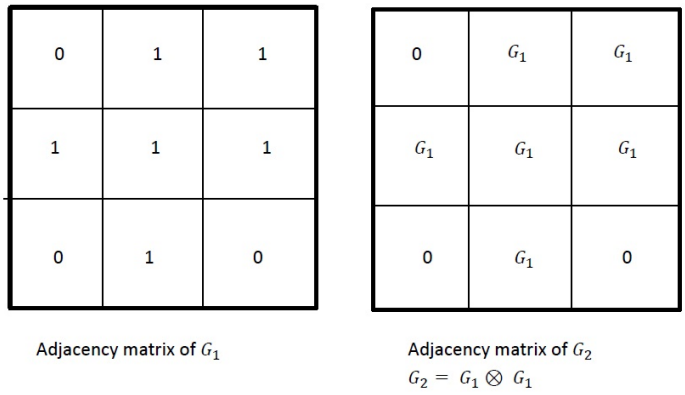


Figure 2.4: The Kronecker model. An example that shows the adjacency matrices for the first and second Kronecker power graphs.

Introducing the probabilistic parameters solves the discrete effect in the degree distribution while it also makes the model less plausible since the user must estimate the values of $\alpha$ and $\beta$. Changing the values of $\alpha$ and $\beta$ plays an important role in generalizing this model to other models as follows:

- For $\alpha = \beta$, the resulting graph is the Erdös and Rényi [11] random graph;
- For $\alpha = 1$ and $\beta = 0$, the Kronecker stochastic version;

- Additionally, by setting the $G_1$ matrix to 2 by 2 matrix, the *R-MAT* [8] graph generator can be obtained.

### *2.4.3 Community-based generators*

Community detection algorithms attempt to uncover subsets of the network that are more tightly interconnected. These clusters can be the result of a community formation process in which the members preferentially communicate to people within their group. Figure 2.5 shows how a graph with community structure looks. The most important methods used to generate synthetic networks for evaluating community detection algorithms are discussed here.

#### *2.4.3.1 GN Network Generator*

Girvan and Newman [13] proposed a model known as *GN* model. This model was designed to handle a maximum of 128 nodes with only four communities of the same size (32 nodes for each community). During the process of generating the network, the new nodes are preferentially attached to other existing nodes within the same community. The *GN* generator has a parameter known as $K_{out}$ that controls the number of edges that link the new node to the other existing nodes. When $K_{out}$ is less than 8, each node shares more links to the nodes within its community than any other communities. The empirical results show that due to the random fluctuation, in a *GN* network, it is possible that some nodes have more links outside their community than links inside that community even when $K_{out} < 8$ which can cause a problem in evaluating the new algorithms.

Nodes in *GN* networks have an average degree of 16 with a nearly uniform distribution, whereas in most of real-world networks, the degree of the nodes follows the power law degree distribution. Besides, real networks often have many more nodes, and the communities can have different sizes. All of these limitations of the *GN* network generator make it a poor choice for being considered an alternative to real networks.

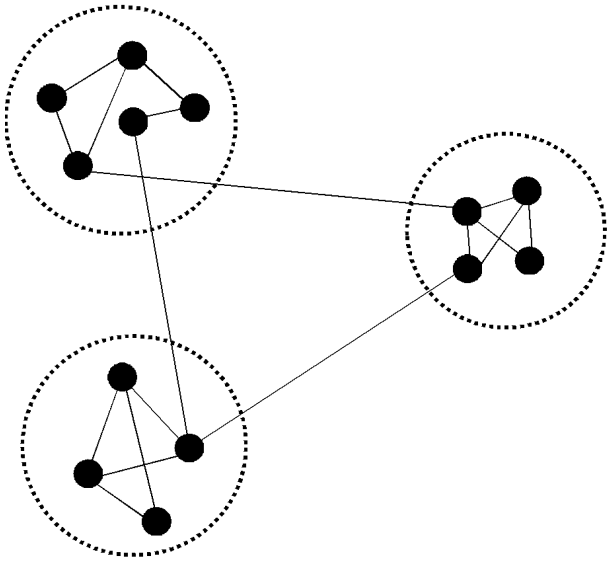

Figure 2.5: This figure shows how nodes are clustered based on their community. Nodes within the same community have more links compared to the nodes in the other communities.

#### *2.4.3.2 LFR Network Generator*

Lancichinetti et al. [20] developed a model based on the *GN* model, known as *LFR*. The *LFR* model is considered a scalable and efficient model that can deal with various numbers of nodes up to $10^5$ - $10^6$ with linear execution time. *LFR* has heterogeneous community sizes and node degrees that follow a power law distribution. Also, in the *LFR* network, the nodes are modeled in a way that makes nodes within the same community more likely to be connected to each other than nodes from other communities based on a mixing parameter. The sizes of the communities follow a power law distribution. In this model, each node is guaranteed to be attached to at least one community. Although, this model addresses drawbacks associated with the *GN* model, it requires tuning several parameters.

#### *2.4.3.3 Kleinberg Model*

Kleinberg proposed a generator in [18] to model the community effect of websites. The process for generating the network is as follows:

- In each timestep, the generator adds or deletes a node based on some probability distribution.
- In each timestep, based on some probability $x$, a node is selected at random and $k$ degrees are assigned to it. That node is linked uniformly at random to the existing nodes.
- With a probability equal to $1 - x$, a new node is chosen at random. Later, edges are copied from multiple nodes and assigned to the previously chosen node, as shown in Figure 2.6.

The idea of copying the edges can be used to create a power law degree distribution and community effects.

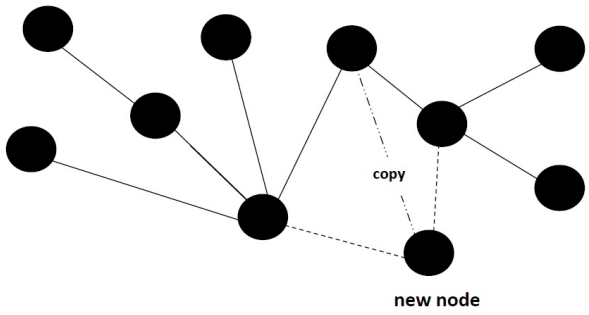


Figure 2.6: Edge copying model in which a new node can copy the links from the other nodes.

### *2.4.4 Forest Fire Model*

Leskovec et al. [22] proposed a model known as the Forest Fire Model. The idea is based on the combination of several models (preferential attachment, copying model, and community guided attachment). The network is generated by starting with a single node and in each subsequent

timestep, one node is added to the network. The newly added node, *A*, is linked to an already existing node, *B*, based on a uniform distribution. After that, the process of burning starts from node *B* by linking to new nodes based on a probabilistic model [22]. The generated graphs have several properties that are present in real graphs, such as: 1) a power law distribution for both the in-degree and the out-degree; 2) community structure created from the edge copying model; 3) small diameters that shrink over time.

### *2.4.5 Chung-Lu (CL) Model*

In 2001, Aiello et al. [1] proposed a model based on a massive graph of long distance telephone "call graphs". This model is able to generate random graphs that can follow any arbitrary desired degree distribution including the power law. This model was improved later by Chung and Lu [10, 9] to generate graphs that model the small world property, such as having a small diameter. The generated graphs are undirected with no isolated nodes. The clustering coefficient is low because there are few closing edges. In order to generate the graphs, the user needs to specify two parameters: $\alpha$ which is the logarithm of the size of the graph and $\beta$ which is the log-log growth rate of the graph (the exponent in the power law distribution). The value of $\beta$ specifies the component sizes that can be summarized as follows [1]:

- When $0 < \beta < 1$, the graph is almost connected.
- When $\beta = 1$, the case is not trivial to decide whether the graphs are connected or not.
- When $1 < \beta < 2$, the second largest component is of size $\theta(n)$, and all the small components are of size $O(1)$.
- When $2 < \beta < 3.47875$, the second largest component is of size $\theta(\log n)$.
- When $\beta = 2$, the second largest component is of size $\theta(\log n \log\log n)$.
- When $\beta > 3.47875$, the graph has no giant component.

The average distance between the nodes is affected by $\beta$ as follows [10]:

- When $\beta > 3$, the average distance is $\log n$.
- When $2 < \beta < 3$, the average distance is $\log \log n$.

In this model the probability that an edge is going to be established depends on the degrees of the endpoints. If all the node degrees are the same, the model will be the Erdös and Rényi random model [11]. In this model the probability of an edge existing depends on the following equation:

$$Pr((i,j) \in Edge) = \frac{d_i d_j}{2m} \tag{2.5}$$

Where $d_i$ and $d_j$ represent the desired degree of nodes *i* and *j* respectively and *m* is the total number of edges. The edges in this model are generated independently which permits edge duplications.

### *2.4.6 Other Models*

Some models were created for specific problems, such as a model designed by Waxman [39] to generate a graph that models network connections following the idea that it is easier to connect routers geographically closer to each other. In this model new nodes form connections with the nearest node as shown in Figure 2.7.

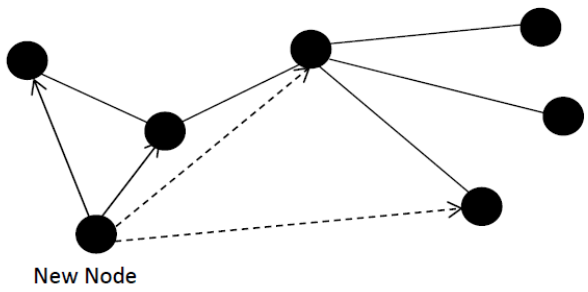


Figure 2.7: Waxman model. Nodes prefer to link to the nodes with the shortest distance between them

Another model was proposed by Calvert et al. [5] to simulate LAN connections on the Internet. Both models are problem specific to computer networks.

### *2.4.7 Baseline*

We selected the Wang et al. [35] generator as a basis for generating the network structure of our model. The Wang et al. generator, extends on previous work by [30], and has been used as a benchmark for evaluating trust prediction and collective classification algorithms. Similar to the BA model, the Wang et al. generator uses both growth and preferential attachment processes for network creation. The network starts with a small number of nodes, and new nodes are added until the network reaches the maximum number of nodes specified by the user. It has two basic parameters: one governing homophily (*dh*) and the other for controlling link density (*ld*). Homophily is a property often exhibited by human social networks such that "birds of a feature flock together" [25]. A high homophily value indicates that links are more likely to be formed between nodes with the same label; these labels can be viewed as being equivalent to community membership. The Wang et al. generator supports the creation of binary, rather than continuous, node

features that are designed to model personal attributes.

This generator yields scale-free networks with some community structure, since nodes with similar labels are more likely to be connected when the homophily parameter is high. Our proposed network generators improve on the Wang et al. generator by adding the following functionality:

- continuous node features;
- multiple link types;
- stochastic optimization procedures for tuning the node features and link formation to match distributions from an existing social media dataset.

## 2.5 Evaluation Metrics

Researchers have found that the best way to evaluate a synthetic network is by its similarity to properties commonly found in real-world social networks. The aim is to match as many features as possible from the real networks. Commonly considered features include:

- Node Degree Distribution
- Network Diameter
- Community Effect
- Clustering Coefficient

### *2.5.1 Node Degree Distribution*

A network can have a complex structure that is not easy to analyze, therefore, it is important to have a measure that captures the network structure in a simplified way. Such a simplification can ignore the complex structure and deal with each node separately by only considering its connections to the other nodes. Counting the degrees of all the nodes in the network generates the degree distribution. In the case of undirected networks, the degree of the nodes is the sum over all the

node connections, while in directed networks, the degree of a node is the sum over all in-degree and out-degree connections. The in-degree is the total number of connections ending at the target node, while the out-degree is the number of connections going out from that node.

Following the power law distribution for the node degree is one of the most important static properties found in real graphs. In many real graphs, most of the nodes have low degrees while only a small fraction of them have high degrees. This fact can be handled easily with the power law distribution since the power law distribution $p_k$ decays polynomially quickly as $k \to \infty$ [7]. The power law distribution exists in various real networks, such as online social networks, WWW, citation graphs, and the Internet. The power law distribution can be described by Equation 2.6.

$$n_k \propto k^{-a},\ where\ a > 0 \tag{2.6}$$

where $a$ is the power law exponent and $n_k$ is the number of nodes with degree $k$.

### *2.5.2 Network Diameter*

Travers and Milgram [33] found that many real networks have relatively small diameters, meaning that most nodes lie within a few hops of one another. In their famous six degrees of separation experiment several participants were asked to deliver a chain letter to different targets chosen at random. The results show that the average length of the chains was six which is a small number compared to the size of the population containing the participants and targets. Leskovec et al. [22] also found that the diameter in real graphs tends to shrink in size or become more stable as the graph grows. The diameter of the network can be measured as the longest distance between the most distant nodes in the network.

### 2.5.3 Community Structure

In many real graphs nodes tend to connect more frequently to nodes in the same community than to the nodes belonging to other communities. For example, students in the same school are more likely to have friends within the school than with people outside that school.

### 2.5.4 Clustering Coefficient

Clustering coefficient is a measure of how nodes are embedded in their neighborhood. A high value indicates that friends of friends are more likely to be friends, which makes the graph more clustered [7]. Given a network where a node $i$ is connected to $k_i$ neighbors and there is also a connection between the neighbors themselves by $n_i$ edges, then the clustering coefficient $C_i$ can be measured as follows:

$$C_i = \begin{cases} \frac{n_i}{k_i} & k_i > 0 \\ 0 & k_i = 0 \; or \; 1 \end{cases}$$

The average of the clustering coefficient gives an indication of the clustering across the network.

## 2.6 Datasets

To evaluate our work, we used the following real datasets:

**DBLP-A**: This dataset includes information about the network generated from 10708 authors in 6 different computer science disciplines (Databases, Data Mining, Artificial Intelligence, Information Retrieval, Computer Vision and Machine Learning).

**DBLP-B**: This dataset includes the network generated between 6251 authors in 6 different computer science disciplines (Algorithms & Theory, Natural Language Processing, Bioinformatics, Networking, Operating Systems, and Distributed & Parallel Computing).

The DBLP-A and DBLP-B datasets [36] have information about collaborating authors who published papers between 2006 and 2008. In these networks, the authors represent the nodes and two authors are linked to each other if they co-authored at least one paper.

**DBLP-C**: The third real dataset that we used is from [37]. It is also a collaboration network of authors who published at least two works in the period from 2000 to 2010. In this network a link is added between two authors if they collaborated at least twice. The data was extracted from 15 different conferences in 6 research fields (Databases, Data Mining, Artificial Intelligence, Information Retrieval, Computer Vision and Machine Learning).

**GameX**: This dataset was extracted by observing gameplay between 3453 players in a massively multiplayer online game. This network contains multiple snapshots and also two link types, message and attack. Nodes have attributes representing crafting, movement, and combat skills possessed by the player avatar.

**Travian**: This dataset was extracted from players participating in a massively multiplayer online game from the real-time strategy genre. This network has 7476 nodes, two link types (attack and message), and multiple time slices.

Table 2.1: The table denotes which models possess the following properties: 1) ability to generate directed graphs 2) ability to generate undirected graphs 3) ability to gradually add nodes 4) usage of a pool of fixed nodes 5) power law degree distribution 6) community structure 7) high clustering coefficient 8) small diameter.

| Generator | Graph type | | Network structure | | Graph properties | | | |
|---|---|---|---|---|---|---|---|---|
| | Direct | Undir. | Grad. add nodes | Fixed nodes | Power law | Com. structure | High CC | Small diameter |
| Erdös and Rényi [11] | | × | | × | | | | |
| Watts and Stogatz [38] | | × | × | | | | × | × |
| Barabási and Albert (BA) [2] | | × | × | | × | | | |
| R-MAT model [8] | × | × | | × | × | × | | × |
| Kronecker Graph [21] | × | | × | | × | × | | × |
| GN network [13] | | × | × | | | × | | |
| LFR network [20] | × | | × | | × | × | | |
| Kleinberg [18] | × | | × | | × | × | | |
| Forest Fire Model [22] | × | | × | | × | × | | × |
| Chung-Lu model [1, 10, 9] | | × | | × | × | | | × |
| Waxman [39] | | × | × | | | | | |
| Sen and Getoor [29] | | × | × | | × | × | | |
| Wang et al. [35] | × | | × | | × | × | | |

# CHAPTER 3: PROPOSED METHOD

This section introduces our proposed methods for cloning social networks. First, we describe the Attribute Synthetic Generator (ASG), a network generator for reproducing the node feature distribution of standard networks and rewiring the network to preferentially connect nodes that exhibit a high feature similarity. Then we describe our Multi-Link Generator (MLG), which uses link co-occurrence statistics from the original dataset to create a multiplex network. The code for both generators can be downloaded from:
`https://github.com/AwradAli/Synthetic-Generators`.

## 3.1 Attribute Synthetic Generator (ASG)

Unlike previous work, the Attribute Synthetic Generator aims to recreate the node level features of the network. In social media datasets, nodes represent users, and node features can be used to denote user profile values. For example, in a dataset extracted from a massively multiplayer online game, nodes would represent players or their avatars, links would represent in-game message exchanges, and node features could be used to designate the avatar's combat or crafting skills. Social media datasets can exhibit profile homophily, an increased likelihood of connection between users with similar profiles. To model this effect, we add extra connections between users with similar node profiles.

Figure 3.1 depicts the architecture of the Attribute Synthetic Generator. The core of ASG is similar to the Wang et al. generator, however links are formed based on feature similarity, in addition to label homophily and preferential attachment. The network is initialized with a group of three nodes, and new nodes and links are added to the network based on the following parameters: link density ($\alpha$), homophily ($dh$), and feature similarity ($fs$). As new nodes are created, their labels are assigned based on the prior label distribution of the social media dataset. ASG includes the

following:

- **dynamic label homophily:** as the network grows, the homophily increases according to the formula $dh = i \times 0.05$, where $i$ is the maximum node index, until the label homophily reaches a maximum predefined value.
- **attribute assignment and optimization:** node feature attributes are initially randomly assigned and then modified to fit the statistics of the social media dataset;
- **link formation based on feature similarity:** to create feature homophily, additional links are created between nodes with similar feature vectors according to the $fs$ parameter.

Table 1 shows the pseudo code of our synthetic network generator, where *N* is the number of nodes, *numCom* is the number of labels (communities), $i$ is the current node index, *dh* is the homophily value, $\alpha$ is the link density, $fs$ is the feature homophily, and *targetStats* contains the targeted statistics from the social media dataset. The output consists of *Net*, a *N* by *N* adjacency matrix, *Label*, a $N$ length vector of label assignments, and $Attributes$, a $N \times A$ matrix, where $A$ is the length of the attribute vector.

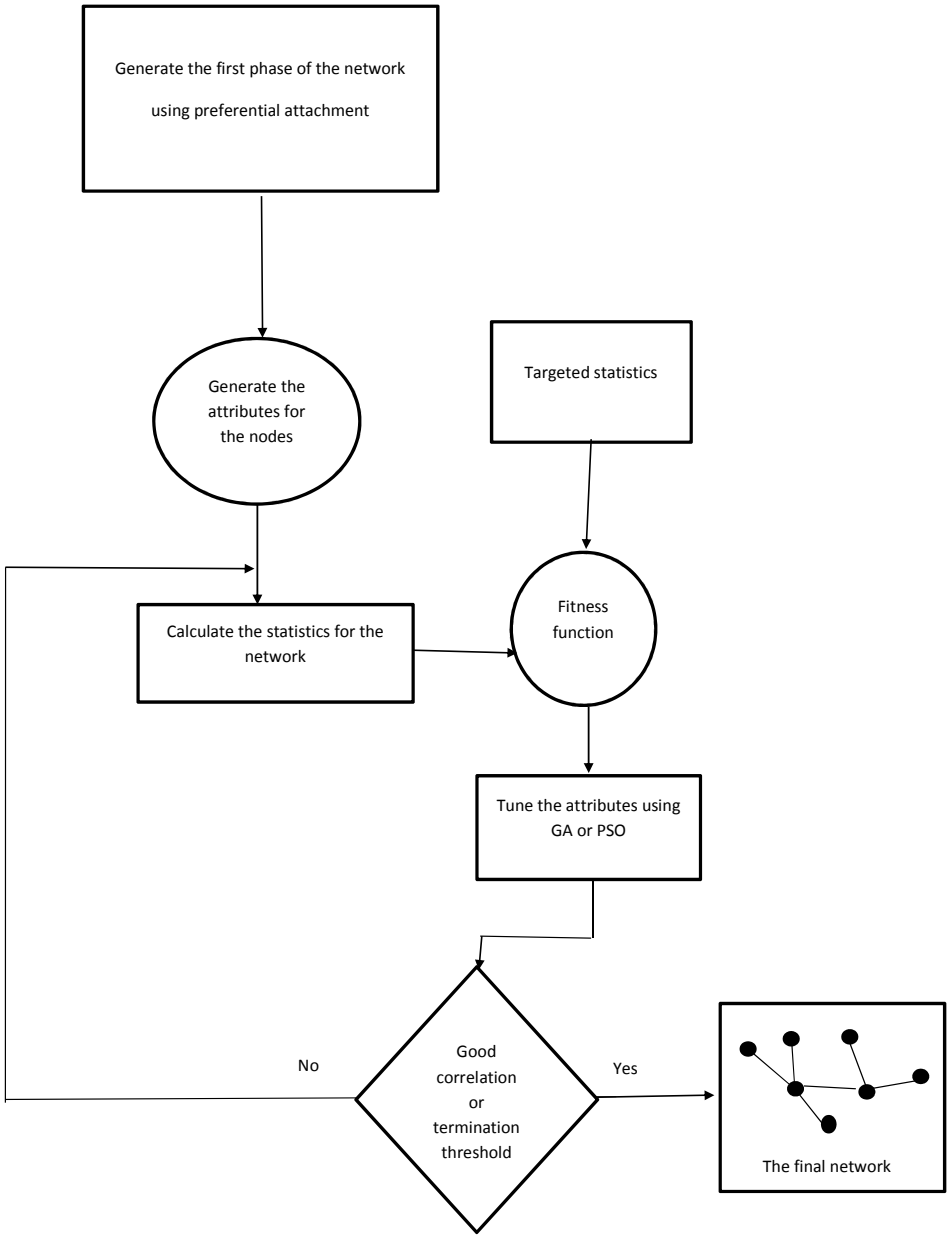


Figure 3.1: A diagram showing the process of generating the network using the ASG generator.

**Algorithm 1** Attribute Synthetic Generator (ASG)

```
Input: N, numCom, α, dh, fs, targetStats
Output: Net, Labels, Attributes
while i < N do
    r = random number between (0,1)
    if i ≤ 2 then
        addNodes(Net,i,numCom,Labels,dh)
        i = i + 1
    else
        if r ≤ α then
            connectNode(Net,i,Labels,dh)
        else
            addNodes(Net,i,numCom,Labels,dh)
            i = i + 1
        end if
    end if
end while
while i ≤ N do
    Attributes = genAttr(i,targetStats,Net)
end while
AddSimilarConnections(Net,Attributes,fs)
```

### *3.1.1 Network Growth*

The network growth is based on the link density parameter $\alpha$, which governs whether nodes or links are added. Adding nodes decreases the density of the network and adding links increases the density. During the growth phase, links are added based on the current label homophily ($dh$) and a preferential attachment model; this growth process results in links are added to high degree nodes with the same label.

### *3.1.2 Attribute Assignment*

After the network has reached the same number of nodes as the original social media dataset, the growth phase terminates and attribute assignment begins. Each node initially receives a random attribute assignment ranging from [0-8] for each possible attribute in the node feature vector. An assignment of zero indicates that the node does not possess that attribute. In our exam-

ple MMOG dataset, attributes represent crafting, movement, and combat skills possessed by the player avatar. The initial assignments are shifted by $-4$ and treated as Z-scores, assuming a normal distribution ($\mu = 0$ and $\sigma = 1$) with $Z$ values ($Z\epsilon[-4, 4]$). The attribute value is calculated as:

$$M = \mu + Z\sigma \tag{3.1}$$

where *M* is the generated attribute value after normalization and $\sigma$ and $\mu$ are target standard deviation and mean respectively.

### *3.1.3 Optimizing Attribute Assignments*

After an initial assignment has been made, a stochastic optimization process is used to move the initial assignments closer to the target distribution extracted from social media dataset. Table 3.1 shows an example distribution of node attributes collected from a massively-multiplayer online game (Game X).

Table 3.1: Target Statistics for Example MMOG Dataset

| Skills | 1st Quartile | 3rd Quartile | Median | Max | Skewness | Std Deviation | Actual Mean |
|---|---|---|---|---|---|---|---|
| Economy1 | 10.24 | 20.01 | 14.18 | 46.56 | 0.87 | 5.62 | 15.67 |
| Economy2 | 10 | 15 | 10.24 | 43.64 | 1.43 | 2.58 | 11.94 |
| Economy3 | 10 | 14.60 | 10.42 | 34.40 | 1.17 | 4.05 | 12.88 |
| Economy4 | 10.72 | 17.49 | 15 | 81.36 | 1.43 | 5.11 | 15.24 |
| Movement1 | 10 | 10.94 | 10 | 47.01 | 2.90 | 2.65 | 11.31 |
| Movement2 | 11.25 | 27.95 | 16.60 | 95.21 | 1.58 | 15.76 | 22.95 |
| Combat1 | 10.42 | 24.18 | 15.46 | 44.01 | 0.93 | 8.80 | 18.52 |
| Combat2 | 11.21 | 27.32 | 20 | 95.11 | 1.49 | 14.99 | 23.15 |
| Combat3 | 11.24 | 31.89 | 19.02 | 96.55 | 1.26 | 16.94 | 25.19 |
| Combat4 | 10 | 15 | 10 | 76.14 | 4.26 | 5.18 | 12.51 |

The fitness function for the stochastic optimization is the average correlation coefficient between the target statistics and our generated statistics:

$$r = \frac{n(\sum xy) - (\sum x(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} \tag{3.2}$$

where *x* is the target statistics, *y* is the current network statistics, *n* is the number of elements and *r* is the average correlation coefficient. The fitness function measures the similarity between our generated attributes and those of the target social media dataset. To tune the generated attributes to match the target social media dataset, we evaluated the performance of two stochastic techniques: particle swarm optimization [17] and genetic algorithms [16].

**Algorithm 2** Particle Swarm Optimization (PSO)

```
Input: Attributes, TargetStats, c1, c2
Output: Attributes
MaxAgents = 30
MaxGeneration = 200
F_Gb = 0
for agent ≤ MaxAgents do
    P = random number between(1,8)
    Calculate F(agent) according to Equation 3.2
    î(agent) = P(agent)
    F_î(agent) = F(agent)
end for
Gb = î(agent(1))
F_Gb = F_î(agent(1))
for generation ≤ MaxGeneration do
    for agent ≤ MaxAgents do
        V(agent) = c1 · (P(agent) - î(agent))
        V(agent) = V(agent)+ c2 · (P(agent)- Gb)
        P(agent)= P(agent) · V(agent)
        Calculate F(agent) according to Equation 3.2
        if F(agent) > F_î(agent) then
            î(agent) = P(agent)
            F_î(agent) = F(agent)
            if F(agent) > F_Gb then
                Gb = P(agent)
                F_Gb = F(agent)
            end if
        end if
    end for
    Attributes = Gb
end for
```

### *3.1.4 Particle Swarm Optimization*

The Particle Swarm Optimization algorithm (*PSO*) introduced by Kennedy [17] is a stochastic optimization technique for maximizing the quality of a problem solution based on a fitness function; prior work [23] has shown that PSO performs well on network optimization problems. The algorithm starts by generating a random set of possible solutions (particles); particles have a velocity which governs the exploration of alternative configurations. Particles update their individual knowledge and global experience as better solutions are discovered. Individual particle knowledge is responsible for exploring the search space, while the global experience is used to exploit the best known solution. The relative weighting of these two components is determined by setting the coefficients.

Pseudo code for PSO is shown in Table 2. *P* is the set of particles (agents). *Gb* is the global knowledge (exploration) which is initialized to be the best individual $\hat{i}$. The $\hat{i}$ is initialized as the current particle *P*. The relative contributions of global and individual knowledge are weighted by the constant coefficients, *c1* and *c2*. *F* is the fitness, the individual best fitness is denoted by $F_{\hat{i}}$, and the global fitness is $F_{Gb}$.

### *3.1.5 Genetic Algorithm*

We also experimented with using genetic algorithms [16] to find the best feature allocation to match the target statistics since it has been used widely as an optimization technique, such as in [15, 19, 14]. Genetic algorithms (*GA*) are inspired from biological evolutionary processes; the basic idea is to have a population of individuals who are chosen to form the next generation according to a fitness function. Operations such as crossover and mutation are used to maintain diversity in the generated individuals (children). In our work, we used tournament selection [26] to select the best individuals among the population based on their fitness function. In tournament selection, the tournament size is used to specify the number of individuals that enter the tourna-

ment. For the crossover operation, we used uniform crossover [32]. The crossover points in the uniform crossover are based on the crossover ratio or mask that indicates which individual (parent) will transfer its chromosomes to the generated children. For mutating the generated individuals (children) from the crossover operation, the user needs to specify the mutation rate that specifies the percentage of the population that will be mutated. We generate a random value (mutation step) that produces both positive and negative numbers randomly. The mutation step will modify all the genes in the selected child (its attributes), i.e., the mutation step value will be added or deleted from all the attributes that belong to that individual (child). Table 3.2 shows the pseudo code of *GA* algorithm. The notations are as follows: *pop* refers to the population size, *p1* and *p2* refers to parent1 and parent2 respectively, *ch1* and *ch2* are child1 and child2. Finally, *i* is the current individual.

Table 3.2: Genetic Algorithm pseudo code

```
function tuneGA(attributes, targetStats)
  Initialize pop ← 100 of size (no.nodes × no.attributes)
  Calculate fitness of pop
  for generation 1 to 200 do
    for i 1 to pop do
      rand ← random number(0 − 1)
      if rand < 0.5
        Tournament selection for p1 and p2
        Xover ch1 and ch2
        Mutate ch1 and ch2
        if fitness(i) < ch1
          i = ch1
        elseif fitness(i) < ch2
          i = ch2
        end if
      end if
    end for
  end for
return best individual
```

### *3.1.6 Adding Nodes based on Feature Similarity*

The tuned attributes are then used to add additional links to the network based on the feature similarity parameter, $fs$. We measure the similarity between the nodes by calculating the correlation between them using Equation 3.2. The higher the $fs$ parameter is, the more links are generated, based on the node similarity function. To generate a link, we select a source node randomly and connect it to the most similar node, its nearest neighbor in feature space.

## 3.2 Multi-Link Generator (MLG)

To handle multiplex networks, we also need to match the co-occurrence statistics of the links in the original social media dataset. We extract the frequency of each link type and also a 2D matrix that models the co-occurrence frequency of pairs of link types. MLG uses the same network growth process as ASG. Based on the link density parameter ($\alpha$), either a new node is generated with a label based on the label distribution of the target dataset or a new link is created between two existing nodes. The link selection process occurs as follows:

The link selection process occurs as follows:

- **Selecting the node anchors:** The first node is chosen at random from the existing node pool while the second node is selected based on a combination of label homophily and degree, using the same procedure as the ASG generator.
- **Determining the main link type:** After selecting the two nodes that will be connected, the main link type is determined by sampling the prior distribution of link types in the social media dataset.
- **Selecting the secondary links:** The existence and type of a secondary link is based on the co-occurrence matrix.

Note that although this process never creates more than one or two links in a single pass, node pairs can be connected by more than two link types if they are selected again at a later iteration.

# CHAPTER 4: RESULTS

This section summarizes our results from comparing networks generated using ASG and MLG to the original real-world social media data.

## 4.1 Attributes Synthetic Generator (ASG)

### *4.1.1 Running Time*

Figure 4.1 shows the running time for several commonly used synthetic generators (LFR, GN, Random graph, and Wang et al. generator) compared to our ASG generator. For our ASG generator, we provide two experiments: in the first one we use the particle swarm optimization (PSO) algorithm to tune the node feature distribution while in the second experiment, a genetic algorithm (GA) is used for modifying the attributes. All the results are reported over an average of three runs. Neither version of ASG shows exponential growth in running time, despite the complexity in the network generation process.

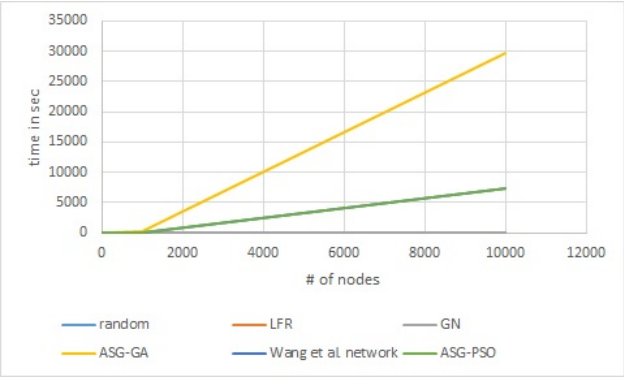


Figure 4.1: Running time of different approaches. Here, the LFR, GN and Random graphs were hard to be see since they are almost a line with the x-axis. The Wang et al. generator and the ASG with PSO were almost the same with few seconds difference.

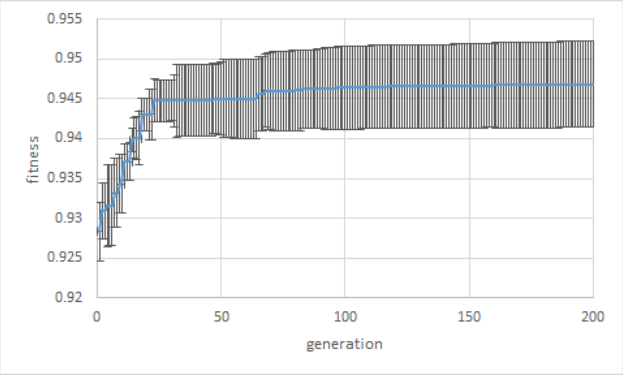


Figure 4.2: Fitness improvement of PSO (error bars mark the standard deviation between runs)

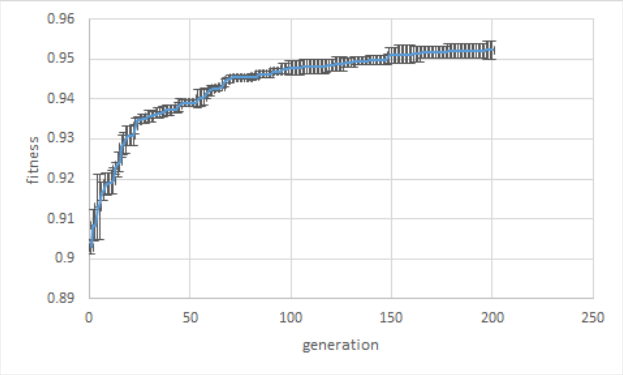


Figure 4.3: Fitness improvement of GA (error bars mark the standard deviation between runs)

### *4.1.2 Fitness Function*

Figure 4.2 and Figure 4.3 show fitness function performance improvement over successive generations with the PSO and GA algorithms respectively for 100 nodes. It clearly asymptotes before the 200 generation termination point and achieves a high correlation coefficient with the target feature statistics for both algorithms. Since there was little difference between the results, we opted to use particle swarm optimization since it is faster.

Table 4.1: Networks comparison with DBLP-A dataset

| | **DBLP-A** | **ASG** | **Wang et al. Generator** |
|---|---|---|---|
| # of nodes | 10,708 | **10,708** | 10,708 |
| # of links | 28,000 | **26,180 ± 86.6** | 15,292 ± 41.8 |
| Network Diameter | 17 | **10.3 ± 1.15** | 14 ± 1 |
| Average Degree | 5.23 | **4.9 ± 0.26** | 2.9 ± 0.01 |
| Average Clustering Coefficient | 0.7 | **0.012 ± 0.01** | 0.01 ± 0.001 |
| Avg. Path Length | 6.235 | **5.58 ± 0.68** | 5.6 ± 0.04 |

Table 4.2: Networks comparison with DBLP-B dataset

| | **DBLP-B** | **ASG** | **Wang et al. Generator** |
|---|---|---|---|
| # of nodes | 6,251 | **6,251** | 6,251 |
| # of links | 16,418 | **15,919.6± 98.85** | 8,832.3± 92.11 |
| Network Diameter | 21 | **8± 0** | 15± 2.082 |
| Average Degree | 5.253 | **4.68± 0.03** | 3 ± 0.029 |
| Average Clustering Coefficient | 0.69 | **0.02± 0.002** | 0.0113± 0.007 |
| Avg. Path Length | 6.589 | **4.675± 0.007** | 6± 1.2 |

## 4.2 Network Statistics

Tables 4.1, 4.2 and 4.3 show the network statistics comparison for the DBLP-A, DBLP-B and DBLP-C datasets respectively; we compare how well the synthetic networks generated by our proposed method (ASG) match the real datasets and the networks generated by the original Wang et al. generator. Table 4.4 shows the Euclidean distance for the network statistics comparison; our modifications to the Wang et al. generator result in a more similar synthetic network, in terms of link number and average degree. The main weakness with both our generator and the Wang et al. generator is that they do a poor job in duplicating the clustering coefficient of the original network, since they lack a procedure for rewiring the network to increase dyadic closure. This can be seen in Figure 4.4.

Table 4.3: Networks comparison with DBLP-C dataset

| | **DBLP-C** | **ASG** | **Wang et al. Generator** |
|---|---|---|---|
| # of nodes | 8,865 | **8,865** | 8,865 |
| # of links | 12,989 | **13,768.67± 109.44** | 12,598± 99 |
| Network Diameter | 29 | **13± 0** | 13± 0 |
| Average Degree | 2.93 | **3.14± 0.065** | 2.842± 0 |
| Average Clustering Coefficient | 0.662 | **0.009± 0.001** | 0.011± 0 |
| Avg. Path Length | 8.389 | **5.408± 0.006** | 5.582± 0 |

Table 4.4: Euclidean distances between real and synthetic graphs for several graph metrics.

| Real dataset | Model | Avg. degree | Avg. path length | Avg. cc | Diameter |
|---|---|---|---|---|---|
| DBLP-A | ASG | 0.34 | 0.6547 | 0.6943 | 6.667 |
| | Wang et al. network | 2.374 | 0.6177 | 0.69767 | 3 |
| DBLP-B | ASG | 0.578 | 1.914 | 0.578 | 13 |
| | Wang et al. network | 2.253 | 0.589 | 0.6777 | 6 |
| DBLP-C | ASG | 0.21 | 2.981 | 0.661 | 16 |
| | Wang et al. network | 0.088 | 2.807 | 0.651 | 6 |

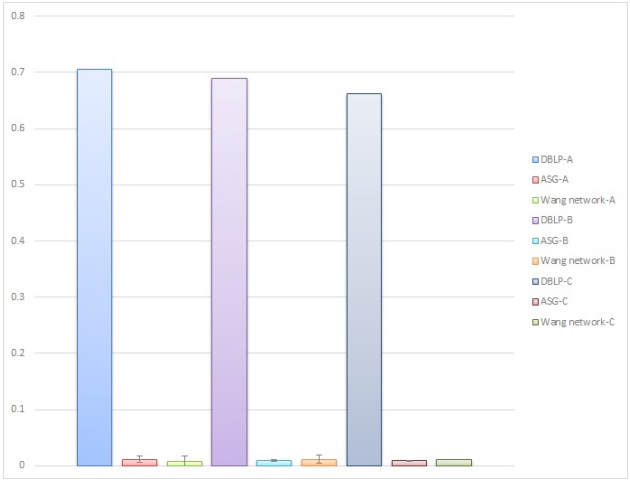


Figure 4.4: Histogram of the clustering coefficient for the real datasets and the synthetic generators.

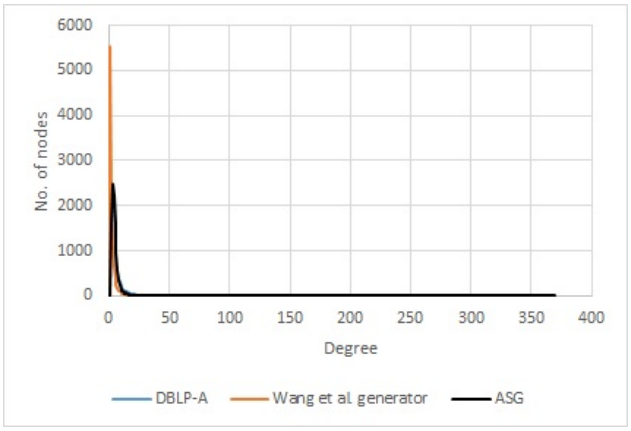


Figure 4.5: Node degree distributions of DBLP-A, ASG and Wang et al. graphs.

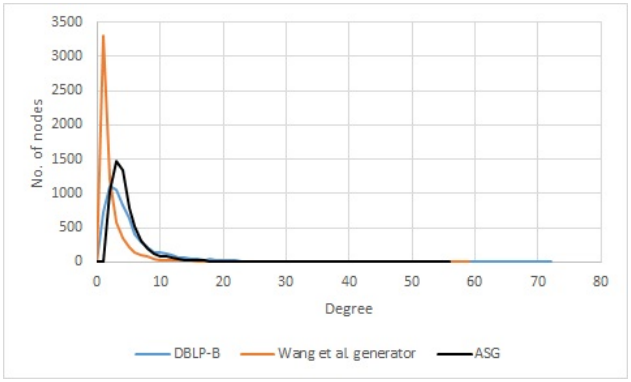


Figure 4.6: Node degree distributions of DBLP-B, ASG and Wang et al. graphs.

### *4.2.1 Degree Distribution*

Figures 4.5, 4.6, and 4.7 depict the degree distribution for (DBLP-A, DBLP-B and DBLP-C with ASG and Wang et al. respectively). As shown in the figures, ASG models the node degree distribution in the DBLP-A network better than the original Wang et al. synthetic network generator while in Figure 4.7, ASG and Wang et al. generator have very similar performance. The effect of the feature homophily $f_s$ can be seen here since in Figures 4.5 and 4.6, this value set high, allowing more connections between similar nodes while this is not the case in Figure 4.7.

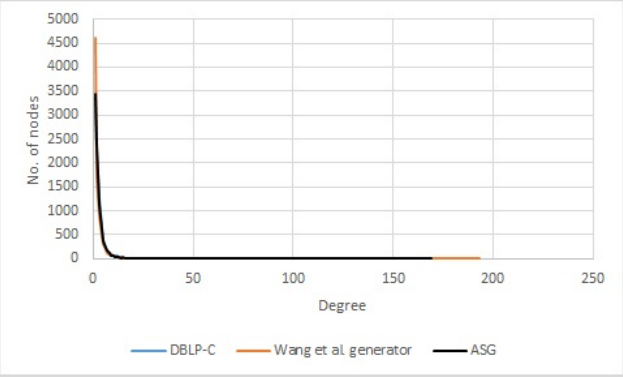


Figure 4.7: Node degree distributions of DBLP-C, ASG and Wang et al. graphs.

Table 4.5: Examining the power law effect in node degree distribution

| Graph | Model | Estimated power law exp. | $R^2$ |
|---|---|---|---|
| A | DBLP-A | 0.073 | 0.6623 |
| | ASG | 0.078 | 0.6442 |
| | Wang et al. network | 0.052 | 0.5431 |
| B | DBLP-B | 0.121 | 0.7515 |
| | ASG | 0.055 | 0.5326 |
| | Wang et al. network | 0.033 | 0.3434 |
| C | DBLP-C | 0.098 | 0.5915 |
| | ASG | 0.084 | 0.696 |
| | Wang et al. network | 0.072 | 0.6288 |

We also fit a power law function to the data from the discussed networks (Table 4.5) to determine the exponent and the $R^2$ (a measure of the goodness of fitting the graph to the power law curve); ASG simultaneously matches the exponent well while achieving a good fit specially for DBLP-A dataset.

## 4.3 Multi-link Generator

Tables 4.6, 4.7 and 4.8 show the network statistics from the GameX network and the synthetic network created by MLG. We set $\alpha$ (link density) to a high value (0.9). Here, MLG synthetic networks have almost the same diameter and the average path length compared to GameX networks. Although the number of edges for both links in our network are different from the real

ones, it is important to note that our network can have more edges in the message network as opposed to the attack network. Figures 4.8, 4.9, 4.10 and 4.13 show the node degree distribution for the real networks and our MLG synthetic networks.

Finally, Table 4.9 shows the network statistics from the Travian game network and the synthetic network created by MLG; none of the other generators we evaluated were capable of duplicating a multiplex network. Our generator performs well at matching the average diameter and the average path length of the GameX and Travian networks across both link types.

Table 4.6: Comparing the statistics of MLG synthetic networks with the online game GameX during day 10

| **Data for day 10** | **GameX-message** | **MLG-message** | **GameX-attack** | **MLG-attack** |
|---|---|---|---|---|
| # of nodes | 3453 | **3453** | 3453 | **3453** |
| # of links | 63,327 | **39,908.3 ± 826.2** | 5,908 | **14,280 ± 411.8** |
| Network Diameter | 9 | **9.33± 0.577** | 15 | **14.6 ± 0.58** |
| Average Degree | 36.679 | **11.56± 0.24** | 3.421 | **4.136± 0.12** |
| Avg. Path Length | 3.79 | **3.77± 0.019** | 5.688 | **5.17± 0.06** |

Table 4.7: Comparing the statistics of MLG synthetic networks with the online game GameX during day 40

| **Data for day 40** | **GameX-message** | **MLG-message** | **GameX-attack** | **MLG-attack** |
|---|---|---|---|---|
| # of nodes | 3812 | **3812** | 3812 | **3812** |
| # of links | 72,711 | **49,506.7± 909.3** | 4,923 | **10,546.7 ± 438.4** |
| Network Diameter | 9 | **9± 0** | 22 | **16.33± 0.578** |
| Average Degree | 38.15 | **12.99± 0.24** | 2.58 | **2.87± 0.29** |
| Avg. Path Length | 3.827 | **3.71± 0.03** | 6.4 | **6.09± 0.08** |

Table 4.8: Comparing the statistics of MLG synthetic networks with the online game GameX during day 70

| **Data for day 70** | **GameX-message** | **MLG-message** | **GameX-attack** | **MLG-attack** |
|---|---|---|---|---|
| # of nodes | 4030 | **4030** | 4030 | **4030** |
| # of links | 73,074 | **57,792.7± 14,214.36** | 4,205 | **14,656.7± 3,777.7** |
| Network Diameter | 10 | **8.67± 0.577** | 19 | **14± 1** |
| Average Degree | 36.26 | **14.34± 3.53** | 2.086 | **3.63± 60.94** |
| Avg. Path Length | 3.88 | **3.65± 0.22** | 6.91 | **5.49± 0.49** |

Table 4.9: Comparing the statistics of MLG synthetic networks with the online game Travian

| | **Travian-message** | **MLG-message** | **Travian-attack** | **MLG-attack** |
|---|---|---|---|---|
| # of nodes | 7476 | **7476** | 7476 | **7476** |
| # of links | 37,904 | **13,996 ± 1751.8** | 77,167 | **65242 ± 1549.2** |
| Network Diameter | 14 | **11.3 ± 0.6** | 23 | **23.3 ± 3.8** |
| Average Degree | 7.34 | **8.7± 0.2** | 10.15 | **1.87 ± 0.2** |
| Avg. Path Length | 4.07 | **4.4 ± 0.04** | 7.63 | **7.67 ± 0.4** |
| Avg. Clustering coefficent | 0.21 | **0.004 ± 0.003** | 0.13 | **0.001 ± 0** |

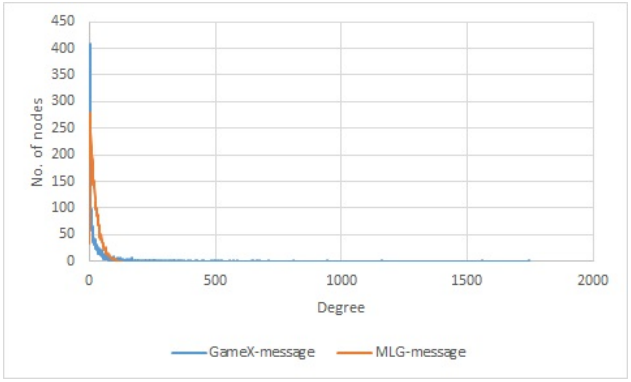


Figure 4.8: Node degree distributions of Game X during day 10 with the corresponding MLG network for message link

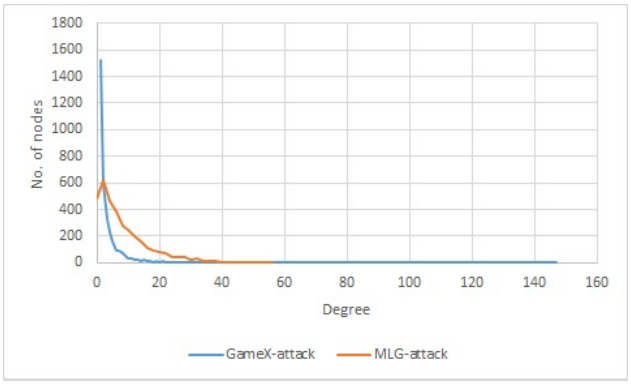


Figure 4.9: Node degree distributions of Game X during day 10 with the corresponding MLG network for attack link

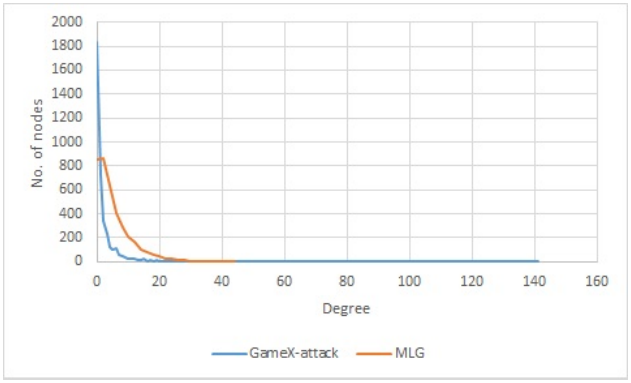


Figure 4.10: Node degree distributions of Game X during day 40 with the corresponding MLG network for attack link

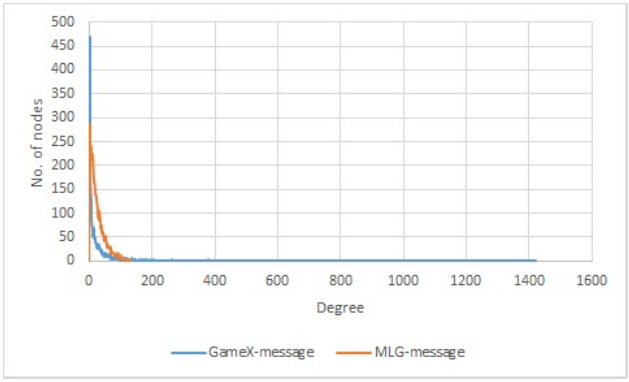


Figure 4.11: Node degree distributions of Game X during day 40 with the corresponding MLG network for message link

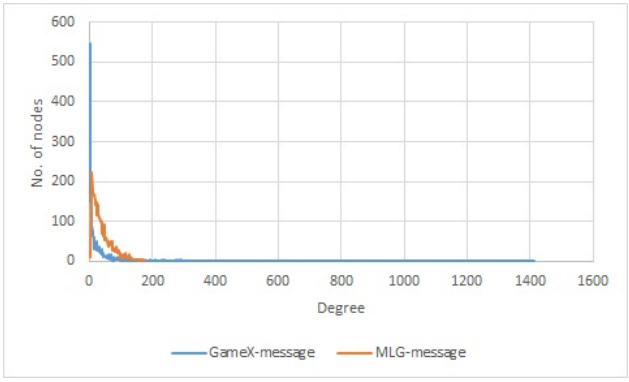


Figure 4.12: Node degree distributions of Game X during day 70 with the corresponding MLG network for message link

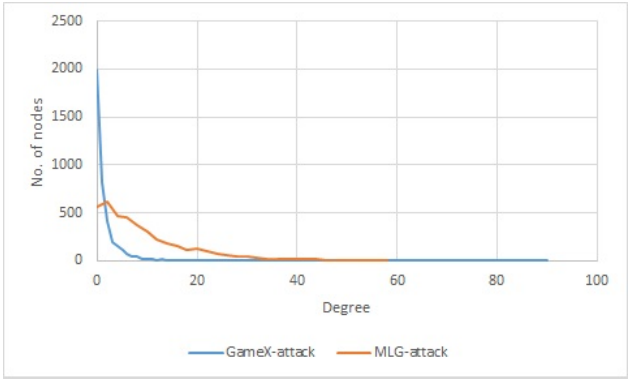


Figure 4.13: Node degree distributions of Game X during day 70 with the corresponding MLG network for attack link

# CHAPTER 5: CONCLUSION

In this thesis, we introduce two new synthetic network generators for cloning social media datasets from a limited set of statistics. Introducing this cloning functionality to network generators is an important step toward preserving user privacy when debugging network analysis software. Additionally our network generators support the creation of continuous node features and multiple link types, which are commonly found in real-world human networks. Our proposed generator, ASG, uses a stochastic optimization procedure (PSO and GA) to tune the node features to match the target dataset and modifies the network structure to link nodes with similar features. Our results show that the proposed improvements improve the generators' ability to match the network statistics of the original dataset. In future work, we plan to introduce dyadic closure to our generator; we believe that this will enable the generator to more accurately match the clustering coefficient.

# LIST OF REFERENCES


[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second Annual ACM Symposium on Theory of Computing*, pages 171–180. Acm, 2000.

[2] A.L.Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[3] A. Barabási, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations, 2002.

[4] G. Bernstein and K. O'Brien. Stochastic agent-based simulations of social networks. In *Proceedings of the Annual Simulation Symposium*. Society for Computer Simulation International, 2013.

[5] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling internet topology. *Communications Magazine, IEEE*, 35(6):160–163, 1997.

[6] K. M. Carley, D. B. Fridsma, E. Casman, A. Yahja, N. Altman, L.-C. Chen, B. Kaminsky, and D. Nave. Biowar: Scalable agent-based model of bioattacks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 36(2):252–265, 2006.

[7] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys (CSUR)*, 38(1):2, 2006.

[8] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.

[9] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[10] F. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2004.

[11] P. Erdös and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

[12] R. Gilles, T. James, R. Barkhi, and D. Diamantaras. Simulating social network formation: A case-based decision theoretic model. *International Journal of Virtual Communities and Social Networking (IJVCSN)*, 1(4):1–20, 2009.

[13] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[14] K. Hamza, H. Mahmoud, and K. Saitou. Design optimization of n-shaped roof trusses. *Ann Arbor*, 1001:48109–2102, 2002.

[15] R. L. Haupt. Thinned arrays using genetic algorithms. *Antennas and Propagation, IEEE Transactions on*, 42(7):993–999, 1994.

[16] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[17] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[19] A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.

[20] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[21] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In *Knowledge Discovery in Databases: PKDD 2005*, pages 133–145. Springer, 2005.

[22] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 177–187. ACM, 2005.

[23] H. M. Lugo-Cordero and R. K. Guha. Evolution of optimal heterogeneous wireless mesh networks. In *Military Communications Conference*, pages 1422–1427. IEEE, 2011.

[24] L. K. Mcdowell, K. M. Gupta, and D. W. Aha. Cautious inference in collective classification. *Machince Learning Research*, 10:2777–2836, 2009.

[25] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[26] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.

[27] G. Palla, L. Lovász, and T. Vicsek. Multifractal network generator. *Proceedings of the National Academy of Sciences*, 107(17):7640–7645, 2010.

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

[29] P. Sen and L. Getoor. Link-based classification. *Reading, Massachusetts: Technical Report, CS-TR-4858, University of Maryland*, 2007.

[30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, pages 93–106, 2008.

[31] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *Proceedings of the DARPA Information Survivability Conference*, volume 2, pages 307–321. IEEE, 2001.

[32] G. Syswerda. Uniform crossover in genetic algorithms. pages 2–9, 1989.

[33] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, pages 425–443, 1969.

[34] M. Tsvetovat and K. M. Carley. Generation of realistic social network datasets for testing of analysis and simulation tools. Technical report, DTIC Document, 2005.

[35] X. Wang, M. Maghami, and G. Sukthankar. Leveraging network properties for trust evaluation in multi-agent systems. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 288–295. IEEE Computer Society, 2011.

[36] X. Wang and G. Sukthankar. Link prediction in multi-relational collaboration networks. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1445–1447, Niagara Falls, Canada, Aug 2013.

[37] X. Wang and G. Sukthankar. Multi-label relational neighbor classification using social context features. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 464–472, Chicago, IL, Aug 2013.

[38] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

[39] B. M. Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988.

[40] M. R. Weeks, S. Clair, S. P. Borgatti, K. Radda, and J. J. Schensul. Social networks of drug users in high-risk sites: Finding the connections. *AIDS and Behavior*, 6(2):193–206, 2002.