Synthetic Generators for Cloning Social Network Data

Awrad Mohammed Ali[1], Hamidreza Alvari[1], Alireza Hajibagheri[1], Kiran Lakkaraju[2], Gita Sukthankar[1]
[1] University of Central Florida, Orlando, Florida
[2] Sandia National Labs, Albuquerque, New Mexico

awrad.emad@knights.ucf.edu, halvari@eecs.ucf.edu,
alireza@eecs.ucf.edu, klakkar@sandia.gov, and gitars@eecs.ucf.edu

Abstract

Synthetic social network generators are useful for a variety of purposes, including benchmarking algorithms, modeling human interactions within agent-based simulations, and debugging code. Despite the increased availability of social media data, collecting data directly from these networks is not always feasible due to privacy concerns. Often data access is restricted to "silos" of analysts with privileged access. Lack of access to the original dataset increases the challenge of debugging the network analysis software. To combat this problem, this paper introduces a multi-purpose synthetic network generator designed for *cloning* the network statistics of an existing dataset. Our network generator supports the synthetic generation of two properties commonly present in real-world networks: node features and multiple link types. We describe common usage cases for our software and provide an evaluation on its performance on recreating the original network.

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 paper, 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 [16], 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 paper, 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.

Moreover, our synthetic network generator was designed to be easily extensible to duplicate other network properties by simply modifying the fitness function to penalize discrepancies in other network statistics. The next section presents an overview of related work on synthetic network generators.

2 Related Work

Synthetic network generators can be broadly categorized as being statistical [5, 25] or agent-based [3, 2]. Statistical approaches focus on reproducing aspects of the network statistics, therefore they are good at preserving these characteristics of the original datasets. Note that these generators do not offer the same dataset cloning functionality as our proposed method, but simply have parameters that can be tweaked by the experimenter. One weakness is that these models may focus on a single graph property, while neglecting other patterns in the network structure. Though these statistical models are good at reproducing the end result of repeated social interactions, they fail to simulate the actual social processes [2]. In agent-based network generators, the networks are constructed by directly simulating the agents' social choices. Agent-based models tend to be very domain specific and are not easily modified for other problems; for instance Carley et al. [3] created an agent-based model for simulating urban disease spread after bioattacks.

2.1 Classical Models

The focus of this paper is simulating human social networks (e.g., [26]), but there is also an extensive literature on constructing other types of networks including biological [19] and computer networks [10]. First we review three classical models that have been used as the basis for many applications.

The earliest work on graph generators was done by Erdös and Rényi in 1960 [5]. The ER model can be used to generate random graphs according to the following procedure. The network starts with N nodes, and each pair of nodes is connected with probability p. When p has a small value, the generated graphs are composed of small equally-sized components with few edges. A high value of p creates graphs with a huge component with size *O(N)*. The diameter of this model grows slowly as the network increases in size, since the diameter is concentrated around $\frac{\log N}{\log z}$, where z is the average degree of the nodes in the graph.

Human networks often exhibit small world characteristics, as illustrated by the famous six degrees of separation experiment [22]. Watts and Stogatz proposed a random graph generator for creating small world graphs with high clustering coefficients and small diameters [25]. The network generation process is initialized with a ring lattice containing N nodes, where each node has k neighbors. For every node, edges are rewired with probability p to a target node, selected uniformly at random. During this process, self-connection and edge duplication is forbidden.

One issue with this procedure is that the degree distribution of Watts-Strogatz graphs remains very close to the initial condition, with every node possessing k neighbors. This is very different from human social networks, that commonly exhibit power law degree distributions. These networks are a result of the Matthew effect in which "the rich get richer"—high degree nodes are more likely than low degree nodes to gain connections over time. In 1999, Barabási and Albert proposed a model to generate scale-free graphs that have power law degree distributions [1]. The network structure is formed by two processes: 1) gradually growing the network over time and 2) preferentially attaching links to high degree nodes. Like many synthetic network generators, our work includes a preferential attachment mechanism to create a power law degree distribution.

2.2 Newer Models

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. The R-MAT [4] model can be used to create graphs with community structure, power-law degree distributions, and a small diameter. Rather than viewing graph generation as a random rewiring process, it can modeled with a matrix recursion procedure in which the adjacency matrix is recursively subdivided and edges are distributed across the partitions.

Following the same recursion idea, Leskovec et al. [13] use Kronecker multiplication to generate self-similar graphs. The network starts with an initial graph G_1 that contains N_1 nodes and E_1 edges. Using matrix 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. Many graphs often densify over time, exhibiting a growth in the number of edges that is superlinear to the number of nodes [14]. Kronecker multiplication produces graphs with a fixed diameter and a densification power law degree distribution with exponent $k = \frac{\log(E1)}{\log(N1)}$. The graph generation process introduces a staircase effect in the nodes' degrees, and each community consists of smaller nested communities that are formed through expansion and recursion.

Due to its ease of use, the most popular benchmark for evaluating community detection algorithms is the LFR generator, an extension of the Girvan-Newman model developed by Lancichinetti et al. [12]. The *LFR* model is a scalable and efficient model that can create networks with 10^6 nodes in linear execution time. *LFR* networks follow a power law degree distribution and can possess heterogeneous community sizes. A mixing parameter governs the amount of connections between different communities and can be used to create more challenging networks for community detection algorithms.

All of the synthetic network generators described in this section are parameterized, allowing different graphs to be generated by modifying the parameters. However, it is left to the experimenter to determine appropriate parameters by trial and error. Different than these generators, the aim of our work is to create a generator that can autonomously reproduce the characteristics of specific datasets. Moreover, we wanted our generator to be able to model two aspects of real-world social media datasets: 1) node level features and 2) different link types.

2.3 Baseline

We selected the Wang et al. [23] generator as a basis for generating the network structure of our model. The Wang et al. generator, extends on previous work by [20], 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" [17]. 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.

3 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.[1]

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 1 depicts the architecture of the Attribute Synthetic Generator. The core of ASG is similar to the Wang

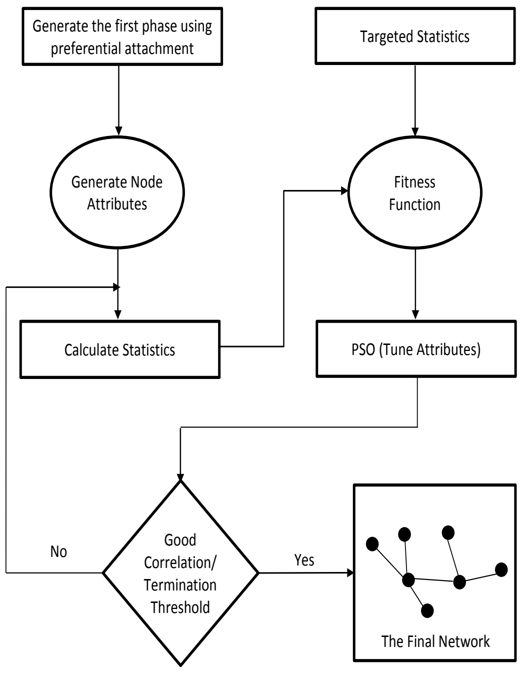

Figure 1: Attribute Synthetic Generator (ASG)

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 (α), 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.

Algorithm 1 shows the pseudocode 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, α 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 NxA matrix, where A is the length of the attribute vector.

3.1.1 Network Growth

The network growth is based on the link density parameter α, which governs whether nodes or links are added. Adding nodes decreases the density of the network and adding links

[1]Code available at: `github.com/AwradAli/Synthetic-Generators`.

Algorithm 1 Attribute Synthetic Generator (ASG)

```
1: Input: N, numCom, $\alpha$, $dh$, $fs$, targetStats
2: Output: Net, Labels, Attributes
3: while i < N do
4:     r = random number between (0,1)
5:     if i $\leq$ 2 then
6:         addNodes(Net,i,numCom,Labels,$dh$)
7:         i = i + 1
8:     else
9:         if r $\leq \alpha$ then
10:            connectNode(Net,i,Labels,$dh$)
11:        else
12:            addNodes(Net,i,numCom,Labels,$dh$)
13:            i = i + 1
14:        end if
15:    end if
16: end while
17: while i $\leq$ N do
18:     Attributes = genAttr(i,targetStats,Net)
19: end while
20: AddSimilarConnections(Net,Attributes,$fs$)
```

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 example 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:

$$M = \mu + Z\sigma \qquad (1)$$

where M is the generated attribute value after normalization and σ and μ 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 1 shows an example distribution of node attributes collected from a massively-multiplayer online game (Game X).

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

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

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 [9] and genetic algorithms [8].

3.1.4 Particle Swarm Optimization

The Particle Swarm Optimization algorithm (PSO) introduced by Kennedy [9] is a stochastic optimization technique for maximizing the quality of a problem solution based on a fitness function; prior work [15] 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 Algorithm 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 [8] to find the best feature allocation to match the target statistics since it has been used widely as an optimization technique, such as in [7, 11, 6]. Genetic algorithms (GA) are inspired from biological evolutionary processes; the basic idea is to have a population of individuals who are selected 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 [18] to select the best individuals among the population based on their fitness function. For the crossover operation, we used uniform crossover [21]. 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. The user needs to specify the mutation rate that regulates 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

Table 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

Algorithm 2 Particle Swarm Optimization (PSO)

```
1: Input: Attributes, TargetStats, c1, c2
2: Output: TunedAttributes
3: MaxAgents = 30
4: MaxGeneration = 200
5: $F_{Gb} = 0$
6: for agent ≤ MaxAgents do
7:     P = random number between(1,8)
8:     Calculate F(agent) according to Equation 2
9:     $\hat{i}$(agent) = P(agent)
10:    $F_{\hat{i}(agent)}$ = F(agent)
11: end for
12: Gb = $\hat{i}$(agent(1))
13: $F_{Gb} = F_{\hat{i}(agent(1))}$
14: for generation ≤ MaxGeneration do
15:     for agent ≤ MaxAgents do
16:         V(agent) = c1 · (P(agent) - $\hat{i}(agent)$)
17:         V(agent) = V(agent)+ c2 · (P(agent)- Gb)
18:         P(agent)= P(agent) · V(agent)
19:         Calculate F(agent) according to Equation 2
20:         if F(agent) > $F_{\hat{i}(agent)}$ then
21:             $\hat{i}$(agent) = P(agent)
22:             $F_{\hat{i}(agent)}$ = F(agent)
23:             if F(agent) > $F_{Gb}$ then
24:                 Gb = P(agent)
25:                 $F_{Gb}$ = F(agent)
26:             end if
27:         end if
28:     end for
29:     Attributes = Gb
30: end for
```

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). Algorithm 3 shows the pseudocode of *GA* algorithm. The notations are as follows: *pop* refers to the population size, *F* refers to the fitness function, *p1* and *p2* refers to parent1 and parent2 respectively, *ch1* and *ch2* are child1 and child2. Finally, *i* is the current individual.

3.1.6 Adding Links 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 2. The

Algorithm 3 Genetic Algorithm (GA)

```
1: Input: Attributes, targetStats,
2: Output: TunedAttributes
3: pop = 100 of size (No.nodes × No. Attributes)
4: MaxGeneration = 200
5: Calculate F(pop)
6: for generation ≤ MaxGeneration do
7:     for i ≤ pop do
8:         r = random number
9:         if r < 0.5 then
10:            Tournament Selection p1, p2
11:            Xover ch1, ch2
12:            Mutate ch1, ch2
13:            if F(i) < F(ch1) then
14:                i = ch1
15:            else
16:                if F(i) < F(ch2) then
17:                    i = ch2
18:                end if
19:            end if
20:        end if
21:    end for
22: end for
```

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 (α), 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:

- **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.

4 Evaluation

Evaluating synthetic network generators is difficult since performance tends to be application specific; if a generator does a good job modeling the elements of interest, then it is less important whether it reproduces other network characteristics. This section presents an evaluation of the performance of our synthetic network generators (ASG and MLG) vs. the Wang et al. generator to show the specific benefits of our proposed improvements. We compare the following network statistics: node degree distribution, network diameter, path length, and clustering coefficient. Experiments were conducted using the best shared parameters for both generators (α of 0.3, dh of 0.8).

4.1 Social Media Datasets

To evaluate our work, we cloned the following datasets:

DBLP-A: This dataset is a collaboration network that includes information about 10,708 authors in 6 different computer science disciplines (Databases, Data Mining, Artificial Intelligence, Information Retrieval, Computer Vision and Machine Learning) who published papers between 2006 and 2008 [24]. In this network, the nodes represent the authors, and two authors are linked to each other if they co-authored at least one paper. This dataset was used to evaluate the performance of ASG.

Travian: This dataset was extracted from players participating in a massively multiplayer online game from the real-time strategy genre. This network has 7601 nodes, two link types (attack and message), and multiple time slices. This dataset was used to evaluate the performance of MLG.

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.

4.2 Results

Figure 2 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. The running time of ASG with PSO is practically identical to the Wang et al. generator. Thus, in all the experiments, we use PSO as our tuning algorithm since it is faster.

Figure 3 and Figure 4 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 2 shows the network statistics comparison for the

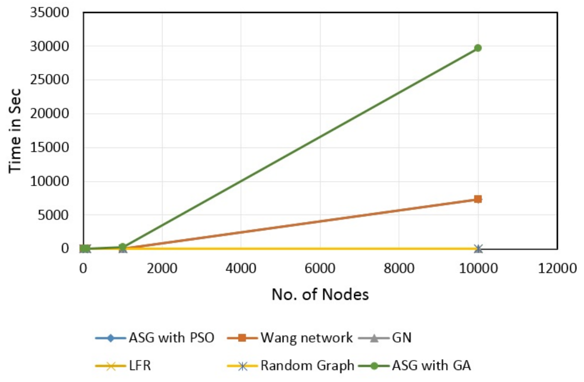

Figure 2: Running time of different approaches

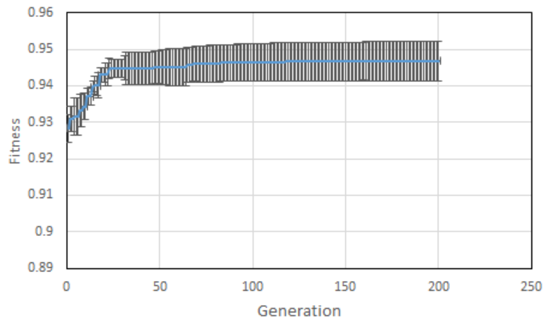

Figure 3: Fitness improvement of PSO (error bars mark the standard deviation between runs)

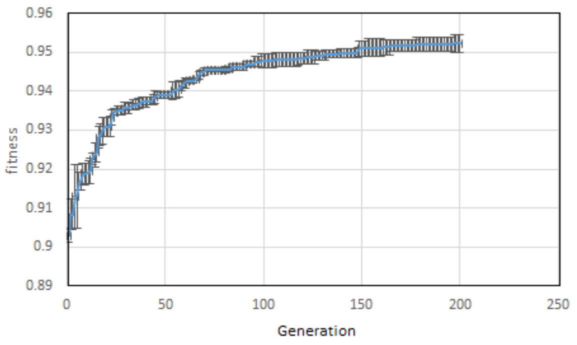

Figure 4: Fitness improvement of GA (error bars mark the standard deviation between runs)

Table 2: Network statistics comparison (DBLP-A)

Statistic	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 ± 1.1	**14.0 ± 1.0**
Average Degree	5.23	**4.9 ± 0.26**	2.9 ± 0.01
Average Clustering Coefficient	0.7	0.01 ± 0.01	0.01 ± 0.001
Avg. Path Length	6.235	5.6 ± 0.68	5.6 ± 0.04

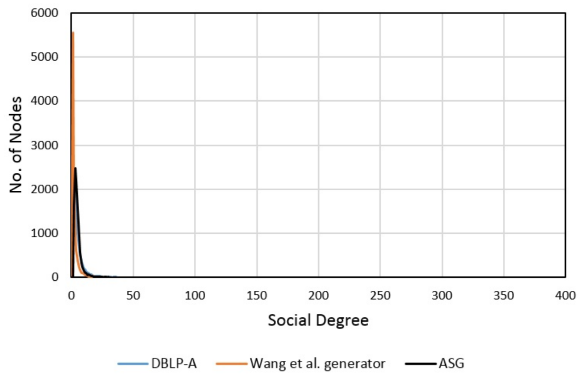

Figure 5: Node degree distributions from DBLP-A, ASG, and Wang et al. networks

DBLP-A dataset; we compare how well the synthetic networks generated by our proposed method (ASG) match the real dataset and the networks generated by the original Wang et al. generator. The table shows that 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.

Figure 5 depicts the degree distribution for the three networks (DBLP-A, ASG and Wang et al.). As shown in this figure, ASG models the node degree distribution in the DBLP-A network better than the original Wang et al. synthetic network generator. We zoomed this figure to show the similarity between the real DBLP-A and our ASG as shown in Figure 6. We also fit a power law function to the data from the three networks (Table 3) 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.

Table 3: Power law distribution fit

Model	Power Law Exponent	R^2
DBLP-A	0.073	0.6623
ASG	0.078	0.6442
Wang et al. network	0.052	0.5431

Tables 4 and 5 provide the statistics summary for both Game X and our MLG networks. For generating the synthetic networks, we set α (link density) to a high value (0.9). Here, MLG synthetic networks have almost the same diameter and the average path length compared to the cor-

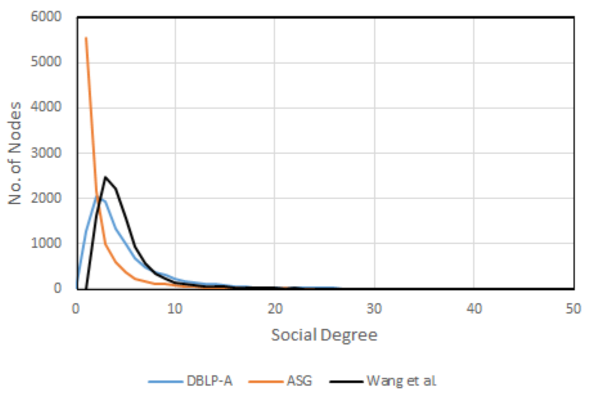

Figure 6: Node degree distributions from DBLP-A, ASG, and Wang et al. networks after zooming

responding Game X networks. Although the number of edges for both links in our network are different from the real ones but it is important to notice that our network can have more edges in the message network as opposed to the attack network. Figures 7, 8, 9 and 10 visualize the node degree distribution. In these figures, our networks have fewer nodes with high and low degrees than the Game X network.

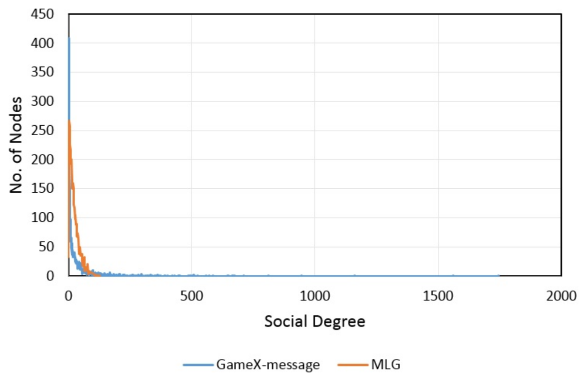

Figure 7: Node degree distributions for the message network from Game X (day 10) and the corresponding MLG synthetic network

Table 6 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 Travian networks across both link types.

Table 4: Network statistics comparison (Game X, 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 5: Network statistics comparison (Game X, 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**

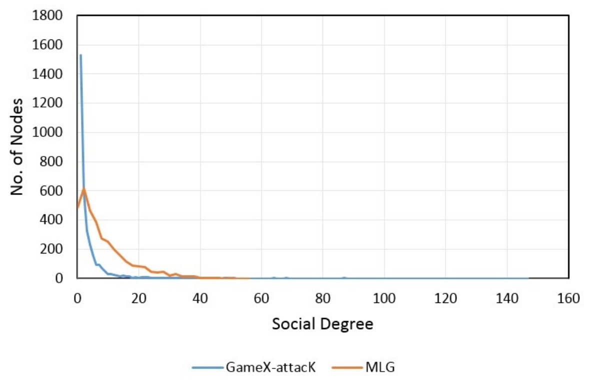

Figure 8: Node degree distributions for the attack network from Game X (day 10) and the corresponding MLG synthetic network

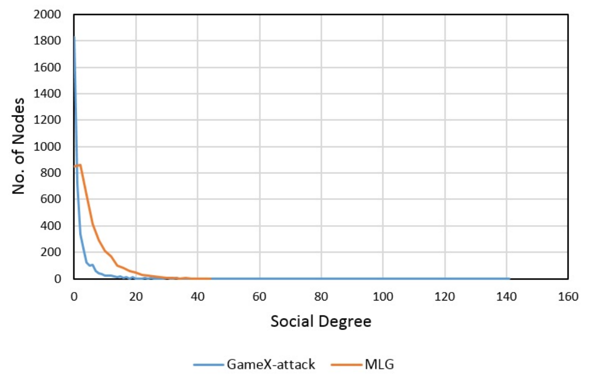

Figure 9: Node degree distributions for the attack network from Game X (day 40) and the corresponding MLG synthetic network

5 Conclusion

This paper introduces 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) 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.

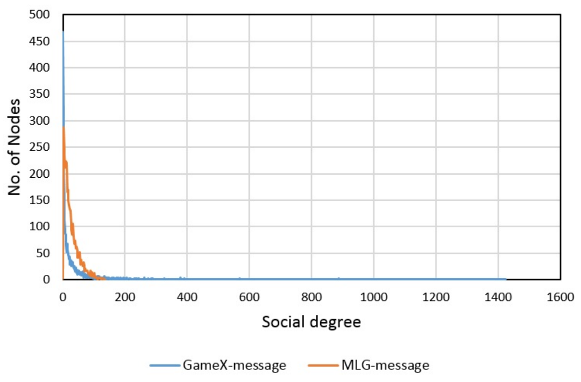

Figure 10: Node degree distributions for the message network from Game X (day 40) and the corresponding MLG synthetic network

6 Acknowledgments

Research at University of Central Florida was supported by NSF award IIS-0845159. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. We would like to thank Dr. Rolf Wigand (University of Arkansas) for providing the Travian dataset used for the evaluation. The first author would like to thank the higher committee for education development in Iraq (HCED) for sponsoring her academic career.

Table 6: Network statistics comparison (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	**65,242** ± **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 Coefficient	0.21	**0.004** ± **0.003**	0.13	**0.001** ± **0**

References

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

[2] 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.

[3] 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.

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

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

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

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

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

[9] 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.

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

[11] 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.

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

[13] 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.

[14] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

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

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

[17] 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.

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

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

[20] 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.

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

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

[23] 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.

[24] 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.

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

[26] 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.