Population Generation for Large-Scale Simulation

Andrew C. Hannon[a], Gary King[a], Clayton Morrison[b], Aram Galstyan[b] and Paul Cohen[b]

[a]University of Massachusetts, 140 Governors Drive, Amherst, MA 01003, USA;
[b]USC Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina Del Rey, CA 90292, USA

ABSTRACT

Large-scale multi-agent simulations must generate agent populations that satisfy a large set of constraints in order to recreate the population structure and demographic properties of the world being simulated. The Hats Simulator is a lightweight model of an urban space in which agents engage in individual and collective activities. In Hats, we generate hundreds of thousands of agents, each of whom belongs to a subset of thousands of benign or terrorist organizations. The population of Hats and organizations must satisfy multiple local and global constraints. Populations need to be generated quickly and reliably to facilitate repeated runs of the simulation. We present two algorithms for solving this problem: one exact but slow and another much faster one based on random bipartite graphs.

Keywords: Population Generation, Simulation, Bipartite Graph

1. INTRODUCTION

Computer simulation is used to research phenomena ranging from the structure of the space-time continuum to population genetics and future combat.[1–3] Multi-agent simulations in particular are now commonplace in many fields.[4,5] By modeling populations whose complex behavior emerges from individual interactions, these simulations help to answer questions about effects where closed form solutions are difficult to solve or impossible to derive.[6] To be useful, simulations must accurately model the relevant aspects of the underlying domain. In multi-agent simulation, this means that the modeling must include both the agents and their relationships. Typically, each agent can be modeled as a set of attributes drawn from various distributions (e.g., height, morale, intelligence and so forth). Though these can interact – for example, agent height is related to agent weight – they are usually independent. Modeling relations between agents, on the other hand, adds a new layer of complexity, and tools from graph theory and social network analysis are finding increasing application.[7,8] Recognizing the role and proper use of these techniques, however, remains the subject of ongoing research.

We recently encountered these complexities while building large scale social simulations.[9–11] One of these, the Hats Simulator, is designed to be a lightweight proxy for intelligence analysis problems. Hats models a "society in a box" consisting of many simple agents, called *hats*. Hats gets its name from the classic spaghetti western, in which the heroes and villains are known by the color of the hats they wear. The Hats society also has its heroes and villains, but the challenge is to identify which color hat they *should* be wearing based on how they behave. There are three types of hats: *benign* hats, *known* terrorists, and *covert* terrorists. Covert terrorists look just like benign hats but act like terrorists.* Population structure can make covert hat identification significantly more difficult. Investigators using the Hats Simulator must be able to control population parameters, and population generation must be fast enough to support studies that vary these parameters. This paper reports our experiences developing algorithms to generate populations whose structure is dependent on experimenter controlled parameters.

In the next section, we outline the general problem of population generation and the details of building populations for the Hats Simulator. This is followed by a brief description of our initial, brute force algorithm

Further author information: (Send correspondence to Andrew Hannon.)
Andrew Hannon.: E-mail: hannon@cs.umass.edu, Telephone: 1 413 545 1459

*The Hats Simulator is used as a test bed to study intelligence analysis tools that could be used to identify covert hats, including algorithms for finding community structure,[12,13] suspicion scoring[14,15] and reasoning about behaviors over time.

(section 3). It was sufficient for small populations, but scaled horribly. We realized that the Law of Large Numbers would allow randomized methods to generate large populations that satisfied our constraints within acceptable bounds (section 5). It was while developing this algorithm that we discovered the connection between our population generation problem and recent results from the theory of random bipartite graphs (section 4). Finally, section 6 examines the properties and performance of our randomized algorithm. Though we will describe our results in terms of the Hats Simulator, these methods apply to any modeling situation in which populations of agents share multiple, overlapping structures, each of which is independent from the others. Our hope is that our experience will benefit others facing the task of generating large populations that require similar overlapping group structure.

2. POPULATION GENERATION IN HATS

In this section we describe the parameters for Hats population generation. All hats belong to multiple organizations and their behavior is generated based on their organization membership. There are two types of organizations: terrorist organizations have only terrorist hats as members (known or covert); benign organizations may include every kind of hat. There are four parameters that control the organizational structure of a Hats population.

1. The total number of each kind of hat,
2. The number of each kind of organization,
3. The relative size (numbers of hats) of each organization for each kind of hat,
4. How often hats are in multiple organizations – the "organization overlap".

Figure 1 contains a portion of an example Hats scenario description. This scenario has 20 benign hats and 12 terrorist hats, 6 of which are covert. There are 2 terrorist organizations and 7 benign ones.[†] At the end

```
(num-benigns 20)
(num-coverts 6)
(num-terrorists 6)
(benign-org-members-ratio '(1 2 3 3 3 2 5))
(covert-org-members-ratio '(2 1))
(terrorist-org-members-ratio '(1 2))
(organization-overlap 0.3)
```

Figure 1. Scenario specification used to generate the organization graph in figure 2.

of population generation, OT0 should have a 2:1 ratio of covert to known terrorists, and OT1 should have a 1:2 ratio of covert to known terrorists. Lastly, the counts of all members (benign and terrorist) of the benign organizations should match the ratios specified in the `benign-org-members-ratio` parameter.[‡]

Figure 2 depicts the population generated from the parameters in figure 1, represented as a graph relating individual hats to organizations. Vertices along the top represent hats whereas those at the bottom represent organizations; links represent membership of a hat in an organization. Benign agents and organizations are represented as ellipses; terrorists ones as triangles; and covert agents as circles. Though we did not initially make the connection, it is clear that Hats populations can be viewed as several overlapping bipartite graphs (graphs with two distinguished kinds of vertices whose edges are only between vertexes of different types) and that the goal of population generation is to build edges between the vertex kinds such that the parameters of the scenario are matched as closely as possible. We will discuss this connection in section 4.

[†]Hats scenarios typically have many more hats and organizations!

[‡]However, this illustrative example does not satisfy those constraints, since the algorithm performs poorly at small-scales, as covered in section 6.

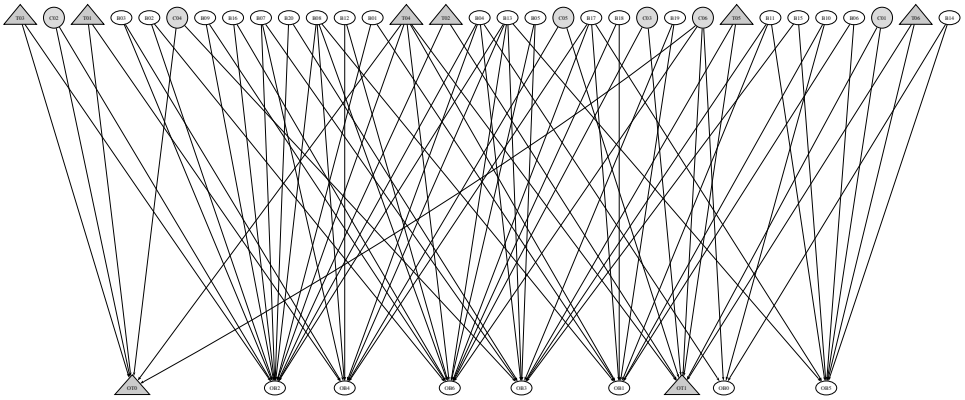

Figure 2. Representation of hat memberships in organizations as a (bipartite) graph. Agents are displayed along the top and their organizations along the bottom. Benign agents and organizations are represented as ellipses; terrorists ones as as shaded triangles; and covert hats as shaded circles.

One of the purposes of the Hats Simulator is to model terrorists embedded within a population. Terrorist organizations consist of known and covert terrorists, but terrorists also belong to benign organizations in order to blend into the population. The `organization-overlap` parameter specifies the degree of inter-organizational mixing. Increasing overlap makes it more difficult to identify organizational structure and, by extension, covert hats. All hats must be in at least two organizations; otherwise the problem of covert agent identification becomes too simple. Suppose covert agents were allowed to belong to only one organization. It would have to be a terrorist one – otherwise the agent would not *be* a terrorist! We could easily find this agent because it would only meet with other terrorists. If, on the other hand, *only* covert hats were required to be in multiple organizations, it would be easy to distinguish them because they would belong to more organizations on average than the benign ones. Thus, the `organization-overlap` parameter determines how many hats will be in *more* organizations than the minimum of two.

3. INCREMENTAL ALGORITHM

Our first population generation approach did not take the reasoning at the end of previous section into account: we required only that all hats be in at least one organization. The `organization-overlap` parameter specified the percentage of organization members that are also joined to other organizations. Each organization therefore consisted of two sets: hats that were only in the organization (*non-overlapping*) and those that were also members of other organizations (*overlapping*). We used a greedy algorithm to incrementally assign multiple organization membership to *overlapping* hats until adding more overlaps would violate the parameters of section 2.

The iterative overlapping assignments were made as follows. Given a population size N, a set of relative organization sizes $\{\rho_1, \rho_2, ...\rho_M\}$ and no overlap, we can calculate the number of hats n_i that are members of organization i as $n_i = N \cdot \rho_i$. We then calculate the number of hats in the organization's overlapping set o_i by multiplying n_i by the organization-overlap parameter. The resulting o_i for each organization is the number of members that should also be in other organizations. We refer to this as the organization's *overlap quota*. Note that assigning a hat to multiple organizations does not just fill a slot in one organization's overlap quota; it simultaneously fills a slot for each of the other organizations of which the hat is a member. However, we have only assigned organizations for a single hat. To adjust for this difference, we subtract 1 from the population and recalculate each organization's overlap quota o_i. We also update o_i by subtracting 1 for each assignment that includes that organization. The process is then repeated, generating a new assignment and updating the overlap quotas, until the difference of all overlap quotas and previous assignments are all zero or less. At this point, the algorithm stops.

Though this algorithm produced populations with correct structure, it had two major faults:

- Its continual recalculations made it much too slow – hours and hours of runtime for larger populations
- The distinction of hats into *overlapping* and *non-overlapping* sets does not generalize easily.

Fortunately, we had already moved towards the randomized approach described below.

4. RANDOM BIPARTITE GRAPHS

Population structures that have organizations can be described as bipartite graphs,[16] where two types of vertices correspond to organizations and hats, as shown in Figure 2. If there are M organizations and N hats in the population, then our bipartite graph consists of M type O vertices and N type H vertices. There is a link between an O vertex o_i and an H vertex h_j if the hat h_j belongs to the organization o_i. Clearly, all the constraints on the population can be translated into the constraints on the bipartite graph.

Let A_{ij}, $i = 1 : M$, $j = 1 : N$ be the adjacency matrix of our bipartite graph. That is, $A_{ij} = 1$ if the hat j belongs to the organization i, and $A_{ij} = 0$ otherwise. A bipartite graph is uniquely characterized by its adjacency matrix A.

Let us define $n_i = \sum_{j=1:N} A_{ij}$ and $k_j = \sum_{i=1:M} A_{ij}$. n_i is the number of edges originating from the vertex o_i, or in other words, the number of members in the organization i. Similarly, k_j is the number of organizations that the j–th agent belongs to. It is easy to see that

$$\sum_{i=1:M} n_i = \sum_{j=1:N} k_j. \tag{1}$$

.

While a generic A describes a population without any explicit structure, we are interested in populations that are subject to certain structural constraints. Below we list examples of constraints, and demonstrate how these are translated into the constraints on the adjacency matrix A.

Each hat should belong to at least m organizations: This is simply translated to

$$k_j = \sum_{i=1:M} A_{ij} > K_{min}, \forall j = 1 : N. \tag{2}$$

Constraints on organization sizes: One of the natural constraints on the population is to require that each organization has a pre–specified number of members:

$$n_i = \sum_{i=1:M} A_{ij} = n_i^0, \forall i = 1 : M. \tag{3}$$

Sometimes it is useful to constrain not the absolute number of hats in an organization, but to require that the relative sizes of organizations scale as specified, $\{\rho_1 : \rho_2 : ... : \rho_M\}$

$$\frac{n_i}{\sum_j n_j} = \frac{\rho_i}{\sum_j \rho_j}. \tag{4}$$

We will assume from now on that the sum is normalized, $\sum_k \rho_k = 1$. The constraint Eq. 4 states that the fraction ρ_i of all the edges in the graph originate from the vertex o_i. We also note that this constraint itself does not specify the total number of edges in the graph (i.e., the total number of membership assignments).

Constraints on Overlap: While the constraints specified above were concerned with the organization sizes, they did not contain any information about the population mixing. On the other hand, in simulations, it is important to control the degree to which various organizations are mixed. For instance, we would like to constrain the number of hats that belong to more than K_{min} organizations.

This can be done by specifying a distribution $P(k)$ for the number of links originating from each hat. Indeed, if $P(k) = \delta_{k,1}$ then each hat belongs to only one organization so the organizational overlap is zero.[§] On the other hand, if we start to shift the distribution function $P(k)$ towards higher values of k, then more hats will belong to more than one organization, hence creating an overlap across the organizations. Note that the distribution $P(k)$ also determines the absolute sizes of the organizations. The expected number of total edges is $N \sum_k kP(k)$ and the expected number of hats in each organization i is $n_i = N\rho_i \sum_k kP(k)$.

To summarize, we want to have an algorithm that generates random bipartite graphs with the following properties:

- Number of O–vertices (organizations): M
- Number of H–vertices (hats): N
- Satisfies the size–ratio constraints Eq. 4
- H–vertices have a specified degree distribution $P(k)$

An algorithm for accomplishing this is listed below. Note that in the actual implementation in Hats, we must deal with collisions that occur when an organization is sampled that the hat already belongs to. Also, we must deal with multiple types of hat and organization vertexes; this is covered in section 5.

GENERATE-RANDOM-BIPARTITE-GRAPH($N, M, \rho_1, \rho_2, ..., \rho_M, P(k)$)

```
1  for i = 1 : N
2      do choose k from distribution P(k)
3         for count = 1 : k
4             do o ← sample from ρ_{1..M}
5                add edge between H_i and o
```

5. RANDOMIZED ALGORITHM

The randomized algorithm, while independently developed, is conceptually the same as the bipartite graph algorithm. The organization size ratios are used to build a lookup table that is then uniformly sampled in order to add edges between agents and organizations. Figure 3 demonstrates how the desired degrees of the organization vertices are used to generate a probability distribution. This lookup table provides a fast method of sampling organizations. Figure 4 shows how a sample from the uniform distribution [0, 1) is used to select an organization.

In Hats, we have two sets of organizations, benign and terrorist, with 3 different probability distributions for the three types of hats (benign, known terrorist and covert terrorist). As described earlier, initially all benign hats must belong to two benign organizations, and all terrorists must belong to one benign and one terrorist organization. When any hat belongs to more than two organizations, it is considered an *overlapping* hat. Unlike GENERATE-RANDOM-BIPARTITE-GRAPH, we treat the process of adding the initial organizations and the overlaps as two separate steps. We do a first pass through the entire population to ensure that all hats belong to the minimum number of required organizations. We then select a subset of the population (as specified by the scenario's `organization-overlap` parameter) and make a second pass to add additional edges between these hats and organizations.

In the heterogeneous scenario of hats, the relationship between hat vertices and organization vertices is equivalent to multiple independent bipartite graph problems. That is, the population of hats is mapped to the different organizations, which have independent probability distributions (see figure 5). The hats will then sample from each set of organizations separately, if applicable (e.g., benign hats will never sample from the set

[§]The δ function is zero everywhere except at k where it is 1.

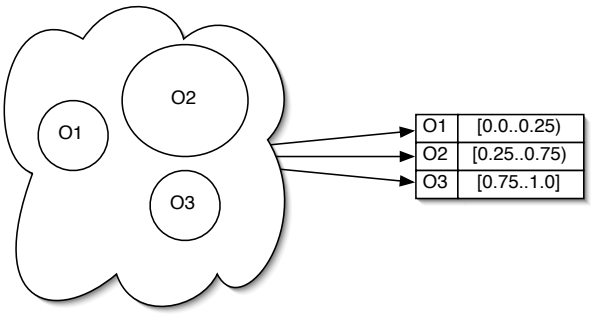

Figure 3. A table of intervals representing the distribution is used to select organizations. Here, the radii of the circles represents the proportion of the population that should belong to each organization (O2 is twice as big as both O1 and O3). The table generated represents the probability distribution for a single set of organizations (a simulation may use multiple organization sets to build complex structures).

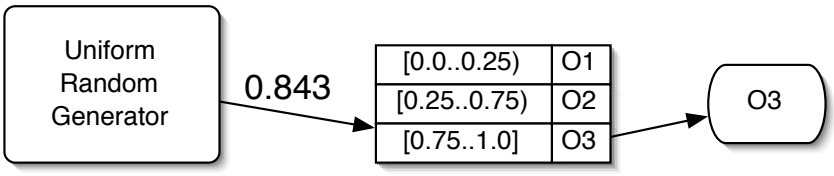

Figure 4. An organization being selected. The uniform random generator produces a value between 0.0 and 1.0, which is then used to look up an organization in the table.

of terrorist organizations), thereby treating each set as an independent bipartite graph solution. This creates the need for an additional probability distribution that is specific to each *type* of hat, which allows it to select which set of organizations it will sample from when adding a new edge. In Hats, the probability of sampling from the set of terrorist organizations is

$$P(terrorist) = \frac{\sum_j n(terrorist)_j}{\sum_j n(terrorist)_j + \sum_j n(benign)_j}, \tag{5}$$

and $P(benign) = 1.0 - P(terrorist)$. Covert terrorists will use a similar function to compute $P(covert)$.

When adding overlaps, we iterate over a subset of the entire Hats population; This subset is of size $N' = N * organization - overlap$. We select the subset uniformly from all possible subsets of size N' so that every hat has an equal chance of being selected.

Given the probability of selecting an organization set to sample from, we can now generate an entire population. The revised algorithm for Hats is therefore broken into two – one for seeding the population, and one for adding overlap:

SEED-POPULATION$(N, M, \rho_{terrorist}, \rho_{benign})$

```
1  for i = 1 : N
2      do if terrorist(H_i)
3          then o ← sample from ρ_terrorist
4               add edge between H_i and o
5          else o ← sample from ρ_benign
6               add edge between H_i and o
7        o ← sample from ρ_benign
8        add edge between H_i and o
```

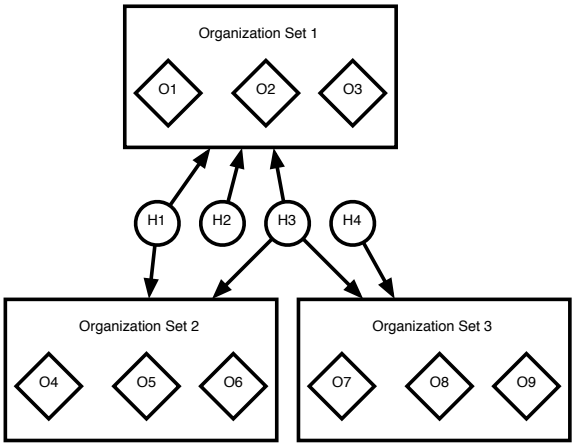

Figure 5. An example of a population of hats (H1..H4) that can join different sets of organizations. Hat H1 can have edges to the vertices in organization sets 1 & 2. Each selection is treated as an independent bipartite graph problem. However, if H1 adds additional edges, then for each additional edge it adds, it needs to select which organization set (of 1 & 2) to sample from.

The procedure SEED-POPULATION connects the initial two organization edges to the hat. If the hat is a terrorist (the process for adding coverts is the same and has been suppressed for clarity), it adds an edge between the hat and a randomly sampled terrorist organization; otherwise, it adds a benign organization. The last two lines add one benign organization so that all hats have two organizations, and at least one is a benign organization.

ADD-ORGANIZATION-OVERLAP($N', M, \rho_{terrorist}, \rho_{benign}, P(k)$)

```
1  for i = 1 : N'
2        do choose k from distribution P(k)
3           for count = 1 : k
4                 do if terrorist(H_i) and sample(P(terrorist)) = true
5                       then o ← sample from distribution ρ_terrorist
6                       else  o ← sample from distribution ρ_benign
7                    add edge between H_i and o
```

In ADD-ORGANIZATION-OVERLAP, we iterate over the random subset of the population, adding new organization edges[¶]. Each type of population is handled separately, so that overlap is distributed evenly across the entire population (i.e., the benign population is processed separately from the terrorist population). Line 2 randomly samples the number of organizations to link the hat to[‖]. Line 4 checks to see if the hat is a terrorist, and, if so, it samples a weighted boolean distribution to determine whether or not to add a terrorist or benign organization. Lines 5-7 sample and add the appropriate edge.

5.1. Summary

The approach of the randomized algorithm is to build a population using a simple sampling mechanism that will cause large populations to converge on the parameters of the scenario. Generating a distribution table in the form of a lookup table provides a fast mechanism for returning sampled data. It is similar to the bipartite graph solution since the hats and the organizations can be seen as the appropriate vertices in a bipartite graph.

[¶]While the algorithms are written for only two types of organizations, this can easily be extended to any number of organization types, as each set of organizations form an independent bipartite relationship with the population.

[‖]In Hats, a poisson distribution is used.

Some details were left out: the actual algorithm needs to handle collisions, and build the distribution table. They were excluded since the algorithms for dealing with them are trivial and they don't add anything to algorithm's description. A poor implementation of the distribution table can greatly reduce the speed of the algorithm, since it is performing a search across M organizations. For this reason, we used a red-black tree to represent the table, as it provides sub-linear search.

6. EXPERIMENTS

Below, we evaluate the speed of the randomized algorithm, and how well the structure matches the parameters. We provide empirical evidence that with large populations, the generated structure very closely resembles the parameters; however, for smaller populations, there is significant error in the final population, and a more precise approach would be better. Also, the speed of the algorithm scales linearly with the size of the population. Thus, large populations can be generated accurately and quickly.

6.1. Speed

To measure speed, we ran two experiments, one varying the size of the population along with the number of organizations, and the other varying the amount of overlap. For each experiment we ran 20 trials at each parameter setting. For the first experiment, we ran 20 trials, varying population sizes from 100 to 100,000 benign hats and 60 to 10,000 covert and terrorist hats. The number of organizations was increased proportionally to the number of agents (up to 5,000 benign organizations for more than 50,000 benign agents). For a given number of organizations, we generated a set of random organization ratios, and used that ratio for each of the 20 trials at that population size. The organization overlap was set to 0.8 throughout the first experiment. Figure 6 is a plot of the mean run times for the 20 trials at each population size. The graph provides empirical evidence that the run time is linearly proportional to the size of the population.

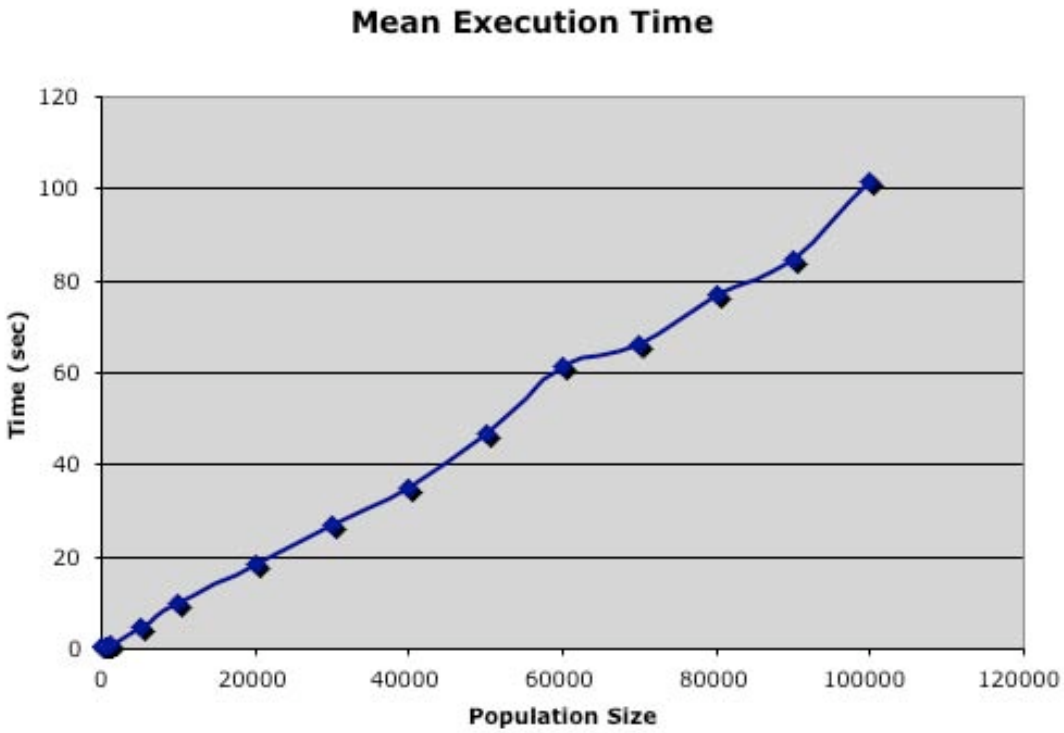

Figure 6. Run-time of the randomized algorithm. The Run-time increases linearly as the size of the population is increased to 100,000 agents.

The second experiment was run with a varying overlap, from 0.0 to 1.0, maintaining a fixed total population size of 70,000 agents with 5,500 organizations. Figure 7 shows that the computation time again linearly increases. The linear relationship suggests that large amounts of overlap will not grossly hinder the speed of the algorithm.

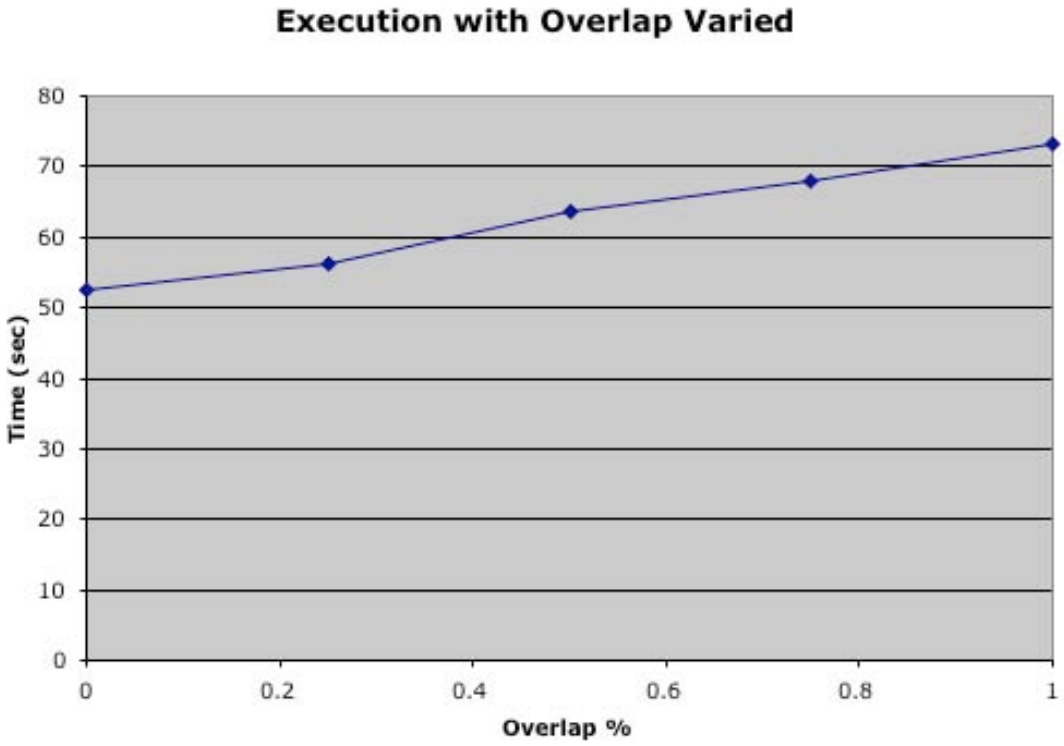

Figure 7. Run-time of the randomized algorithm when population is fixed at 70,000 agents and overlap is varied from 0.0 to 1.0. Run-time increases linearly.

6.2. Parameter Matching

The populations and underlying social structure generated by the Hats simulator are best measured by looking at how well they fit the given distribution. To measure this, we compare the degrees of the organization vertices to that of what we would expect to see. In figure 8, a mean of 1.0 indicates that the generated population met the specified *member-ratio* parameters exactly. As you can see, the confidence intervals converge to a minor amount of error as the populations get larger. What should be inferred is that the randomized algorithm is not well suited for small-scale simulations, and realistically, should only be used on populations with more than 5000 agents, at the very least.

The population parameters in figure 9 were used to generate the organizational structure shown in figures 10 and 11. The first graph was created with a `organization-overlap` of 0.1 whereas the second was created with an `organization-overlap` of 0.9. The increased organizational mixing is shown clearly as many more hats belong to multiple organizations in figure 11 than do in figure 10.

7. SUMMARY AND FUTURE WORK

Many large-scale simulations require that unique populations be generated quickly. The algorithm we have presented is based on bipartite graph generation techniques and allows large populations to be generated in times that in practice scale linearly with the size of the population. Population generation remains an area of open research. The more we learn about the nature of real-world networks, the more we understand how simple random graph models have failed to capture the necessary structure. Interesting real-world networks tend to

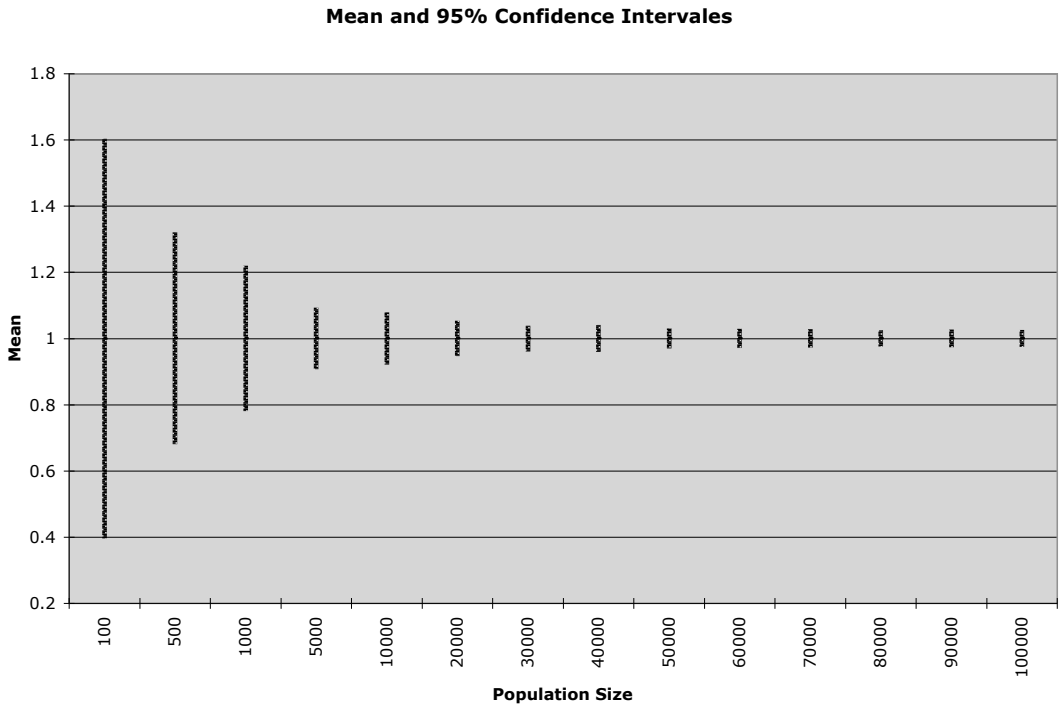

Figure 8. Here, you can see the confidence intervals of the generated distribution converging to the desired distribution as the population gets larger. With smaller populations, there is a significant amount of variation in what was expected and what occurred.

```
(num-benigns 50)
(num-coverts 6)
(num-terrorists 12)
(benign-org-members-ratio '(1 2 3 2 5))
(covert-org-members-ratio '(3 2 4))
(terrorist-org-members-ratio '(4 2 3))
(benign-organizations (create-benign-organization-ids 5))
(terrorist-organizations (create-terrorist-organization-ids 3))
```

Figure 9. Scenario specification used to generate the organization graphs in figures 10 and 11.

be scale-free, have high clustering (friends of a friend are often friends) and other complex structure.[7,8,17] The relatively simple bipartite graphs discussed in this paper could be used to extend population generation to more accurately reflect those richly structured real-world networks.

ACKNOWLEDGMENTS

Work on this project was funded by the Air Force Research Laboratory, account numbers 53-4540-0588 and F30602-01-2-0580. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the Defense Advanced Research Projects Agency/Air Force Materiel Command or the U.S. Government.

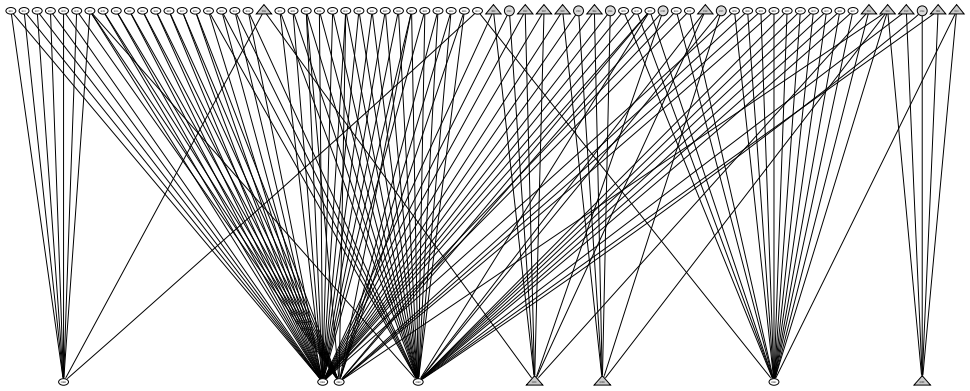

Figure 10. Bipartite hat organization graph for the scenario in figure 9. The organization-overlap parameter is set to 0.1.

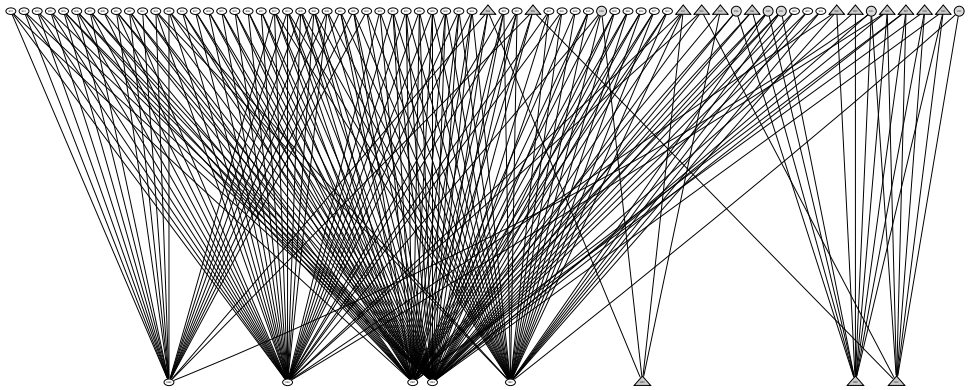

Figure 11. Bipartite hat organization graph for the scenario in figure 9. The organization-overlap parameter is set to 0.9.

REFERENCES

1. W.-M. Suen, E. Seidel, I. Foster, M. Norman, J. Shalf, and M. Parashar, "Astrophysics Simulation Collaboratory."
2. "See Population Biology Simulations at http://darwin.eeb.uconn.edu/simulations/simulations.html," January 2004.
3. M. S. Atkin, D. L. Westbrook, and P. R. Cohen, "Capture the Flag: Military simulation meets computer games," in *Proceedings of AAAI Spring Symposium Series on AI and Computer Games*, pp. 1–5, 1999.
4. A. Joshi and T. Drashansky and J. Rice and S. Weerawarana and E. Houstis, "Multiagent simulation of complex heterogeneous models in scientific computing," 1997.
5. S. Calderoni and P. Marcenac, "Emergence of Earthquakes by Multi-agent Simulation," 1997.
6. R. H. Crites and A. Barto, "Elevator Group Control Using Multiple Reinforcement Learning Agents," *Machine Learning* **33**, pp. 235–262, 1998.
7. M. E. J. Newman, "The Structure and Function of Complex Networks," *SIAM Review* **45**, pp. 167–256, 2003.
8. J. Scott, *Social Network Analysis: A Handbook*, Sage Publications, 2 ed., 2000.
9. P. R. Cohen and C. T. Morrison, "The Hats Simulator," in *Proceedings of the 2004 Winter Simulation Conference*, 2004.

10. C. T. Morrison, P. R. Cohen, G. W. King, J. J. Moody, and A. Hannon, "Simulating Terrorist Threat with the Hats Simulator," in *Under Review for the First International Conference on Intelligence Analysis*, 2005.
11. G. W. King, M. Schmill, A. Hannon, and P. R. Cohen, "Asymmetric Threat Assessment Tool (ATAT)," in *Under Review for BRIMS*, 2005.
12. J. Adibi, P. R. Cohen, and C. T. Morrison, "Measuring confidence intervals in link discovery: A bootstrap approach," in *Proceedings of the ACM Special Interest Group on Knowledge Discovery and Data Mining (ACM-SIGKDD-04)*, 2004.
13. M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Condensed Matter Abstracts* **69**, 2003.
14. A. Galstyan and P. R. Cohen, "Identifying Covert Sub-Networks Through Iterative Node Classification," in *Under Review for the First International Conference on Intelligence Analysis*, 2005.
15. S. A. Macskassy and F. Provost, "Simple Models and Classification in Networked Data," 2004.
16. M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review* **64**, 2001.
17. M. Sageman, *Understanding Terror Networks*, University of Pennsylvania Press, 1 ed., April 2004.