Published in Proc. Intl Conf on Computer Modeling and Simulation, Brno, Sept 2009.

HPCgen – A Fast Generator of Contact Networks of Large Urban Cities for Epidemiological Studies

Tianyou Zhang[1], Soon Hong Soh[1], Xiuju Fu[1], Kee Khoon Lee[1], Limsoon Wong[2], Stefan Ma[3], Gaoxi Xiao[4], Chee Keong Kwoh[4]

[1]Institute of High Performance Computing, Singapore [2]National University of Singapore [3]Ministry of Health, Singapore [4]Nanyang Technological University, Singapore

zhangty@ihpc.a-star.edu.sg

***Abstract*—A contact network is the well representation of heterogeneous contact behaviors within the population. Incorporating contact networks as well as community structures is important in realistic modeling and simulation for the spread of infectious diseases. We developed the "HPCgen", a fast and generic generator of contact networks of large urban cities, with the capacity of automating network re-generations for intervention studies. The produced contact networks are applicable in both analytical modeling and agent-based simulations. In this paper, we presented the design and realization of HPCgen followed by the empirical results of building Singapore contact networks with six types of community structures in the common urban settings. The results showed our 8-node parallelized HPCgen could generated a contact network of 3.4 million populations within 62.17 seconds, which is 90% reduction of runtime.**

I. INTRODUCTION

Infectious diseases are the global health threats to human beings. In epidemiology, compartmental models [1-3] have been applied to study the spread of infectious diseases for decades. Those models well portray the transmission dynamics of infectious diseases by nonlinear differential equations, but they lack network topology of the population as a result of "fully mixed" assumption of the population, i.e. anyone of the infected has an equal probability to infect the susceptible. Such assumption is not generally valid because individuals only contact a small fraction of population and they interact with each other heterogeneously [4]. To address the heterogeneity in social interactions, the most expressive approach is to form a structure of "network" by taking all individuals as vertices (or nodes) and their social connections as edges. In the context of infectious diseases, we can further specify an individual's social connections are the set of people with whom the individual may contact during the period when he or she is infective [5]. The resulted networks are called as contact networks or contact graphs [4-6].

The introduction of network topology gets rid of the assumption of "fully mixed" population and allows modeling the heterogeneous transmission patterns of infectious diseases. In the recent years, a number of researchers have pursued a mathematical theory of spreading dynamics of infectious diseases on contact networks [5,7-10]. In particular, Newman [5] formulated this problem by applying percolation theory from the domain of physics, providing a solid analytical framework to estimate the final state of the outbreaks of infectious diseases. Besides analytical modeling, network topology also provide the foundation for agent-based simulation models of infectious diseases such as [11,12]. Agent-based simulations are the common tools for studying complex systems. By representing people (or places) as "agents", the computer programs can simulate the transmission of infectious diseases by agent interactions along the edges within contact networks. Such methods are more intuitive than mathematical modeling and advantageous in visualizing the spreading dynamics of infectious diseases.

No matter mathematical modeling or agent-based simulation, the underlying contact network forms the foundation of the studies. The generation of contact network could be challenging on account of large population size and hierarchical community structures. (1) Contact networks of large urban cities are of size up to 13 million nodes [13]. How to represent and generate such large networks efficiently could be a problem. (2) A number of community structures (such as households, schools, hospitals, etc) exist in the population. Those community structures represent the social groupings which have the substantial impact to the network topology. That leads to the necessary incorporation of community structures in the contact network. However, how to tackle the following growth on the complexity of network generation would be another issue.

In the literature of network generation, there are two types of approaches. (1) Incremental generation models [14-17] start from an initial node or small graph, gradually adding each new node and edge based on some function of preferential attachment. (2) Configuration models [18-21] construct random graphs with the prescribed degree sequences. The probability of connecting an edge is proportional to the degree of end-node. FastGen model [22], a variant of configuration models, is a fast social network generation model used in EpiSim [12]. FastGen model produces bipartite networks that comprise both people and place nodes. By introducing place nodes, the average degree of the network decreases as people-to-place degrees are generally smaller than people-to-

people degrees. Moreover, as FastGen only allow people-to-place edges, the search space of end-nodes are restrained to place nodes. With fewer edges and smaller search space, FastGen model demonstrates near-linear time complexity, compared to the quadratic time in the standard configuration models such as CL-model [22]. However, there are two drawbacks in FastGen model. (1) Place nodes in bipartite networks cannot represent the hierarchical community structures. For example, place nodes can represent individual schools or classes, but cannot reflect their hierarchical relationship in bipartite networks. (2) Bipartite networks cannot be directly used in the Newman's analytical models [5] which require people-to-people networks. The conversion from bipartite network to single-type network is not trivial because connecting the individuals visiting the same place nodes based on some degree sequences is time-consuming – equivalent to run a configuration model at each place node.

In this paper, we propose HPCgen, a fast generator of contact networks for large urban cities. HPCgen is a generic model to construct contact networks based on real-world demographics, community structures and social contact behaviors. It is comprised of multiple components; each component represents a specific type of hierarchical community structures. The output of HPCgen is a single-type undirected contact network (with size of city population) that can facilitate both analytical model and agent-based simulation models. Our contributions are threefold.

(a) Generic Model. HPCgen allow users incorporating new components as they desire. As long as providing the required parameters in the prescribed format (described in Section 4.1), adding new components in HPCgen is straightforward and transparent to the existing components.

(b) Fast Generation. Both structural optimization and high performance computing techniques are applied to improve the efficiency of HPCgen. (1) We use adjacent matrix to represent contact networks. The advantage is able to encapsulate millions of pair-wise operations during contact (edge) generation into single matrix operation. The optimization of matrix operations is a mature technique in soft computing and many existing toolkits can be used to achieve efficient and scalable computations. (2) We use sparse representation of matrices in compressed column format [23] as many "0"s (no contact edge) exists in the giant matrix of contact network. Furthermore, since HPCgen generates undirected networks, the matrices are symmetric along the diagonals. It inspires us to only use the upper triangular matrices instead of the full matrices. Other than the benefit of lower resource consumption, sparse representation of "half-matrices" also reduces the search space of indexing and searching operations in network construction. (3) We employ MPI (Message Passing Interface) [24] parallelization technique to execute multiple computations simultaneously. Each component in HPCgen is comprised of a number of compartments. Those compartments could be distributed to multiple parallel processors and carry out the network generation concurrently. Therefore, the overall runtime is reduced.

(c) Automation for Repeated Generations. HPCgen is able to automate repeated generations of contact networks, required by structural analysis especially intervention studies of infectious diseases. The common intervention strategies such as quarantine, school closure, and vaccination could effectively control the spread of infectious diseases by cutting off the contact edges between individuals. To simulate such interventions, we have to re-generate the contact networks to reflect the structural changes led by contact removal. An extensive study on interventions might require those re-generations by hundreds of times to simulate the impact of individual interventions. In this circumstance, an efficient contact network generator is far more important. HPCgen would be very helpful in such cases, as it is equipped with automation scripts to repeat the generation process as required. The accumulated gain of speedup would become even more significant than single-time generation.

II. CONCEPTUAL DESIGN

In the current presetting, HPCgen includes six types of common contacts in the urban settings: 1) contacts within households; 2) contacts within hospitals and wards; 3) contacts within schools and classes; 4) contacts within workplaces; 5) contacts within shopping malls; 6) contacts within public transport. The first four are the persistent contacts, which are repeated in the typical workdays; the last two are the transient contacts, but take place in the close settings with high population density. Therefore, we consider these six types of contacts are the dominant transmission channels of infectious diseases in the urban settings. Recall that in the HPCgen, "component" represents a particular type of community structures. Corresponding to the chosen contacts, we include six components in HPCgen presently: *households*, *hospitals*, *schools*, *workplaces*, *shopping malls* and *public transports* (illustrated in Fig 1).

As a component refers to a collection of community structures of the same type, we call an individual community structure as "compartment". A compartment might have the hierarchical groupings internally, which is referred as "multi-layers" in the context. For an instance, *schools component* is a set of two-layer compartments which includes top-layer *school* and sub-layer *class*. For *schools component*, HPCgen firstly selects all the students from the entire population, and distributes them to the individual schools. In each school, the students are further distributed to the individual classes. According to social

groupings, we can classify the contacts between students to be cross-school, cross-class (equivalent to within-school) and within-class. HPCgen allows specifying the different contact rates for individual type of contacts.

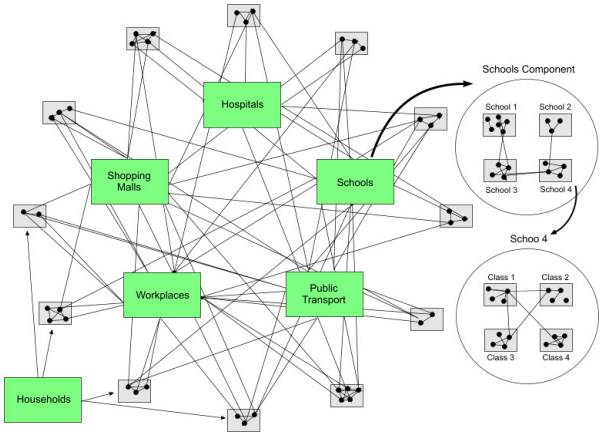

Figure 1. Conceptual design of HPCgen: six types of community structures with hierarchical groupings

III. DESIGN REALIZATION

A. *Input Data*

The input data of HPCgen are comprised of three types: demographic data, structure descriptors, and contact behavior profiles. Demographic data such as population size and age distribution are used in population initialization, people selection and distribution to the respective compartments. Structure descriptors carry the realistic information about community type, hierarchical layers, compartment count and size, and eligibility criteria about the people involved. Contact behavior profiles describe both cross-compartment and within-compartment contacts in the individual layers of each compartments. The profiles can be represented in form of arbitrary degree distributions. In the current presetting, we selected Poisson distribution as a representative, using the distinct mean values (average numbers of contacts) to reflect different types of contacts included in HPCgen.

B. *Generation Procedure*

Recall that one of our contributions is the generic design, – generation procedure is generalizable to every component in HPCgen. By taking in the specific input data from individual components, the same piece of codes will generate "sub" contact networks respectively. Integrating all these "sub" contact networks will finally form a complete contact network of our interest. We call this piece of codes as "generic engine", meaning a central engine driving the generations in all the components.

Recall that the compartments in HPCgen have multi-layer groupings. As there are some dependencies between the layers (e.g. compartment size in the top-layer determines the total population size in the sub-layer), we use a recursive approach to implement them. At each layer, there are three stages in the generic engine: structure construction, contact generation, and population mapping. Fig.2 shows the pseudo codes of contact network generation.

```
1  Function GenericEngine(m):    m is compartment at the current layer
2
3    network = empty contact matrix;
4
5    size[] = Compute size distribution of  individual compartments in m;
6    unit[] = Construct all compartments based on size[];
7
8    subnetwork = Generate cross-compartment contacts in unit[];
9    network = Integrate network and subnetwork;
10
11   for i in each unit[]
12          if i is bottom-layer compartment
13                 subnetwork = Generate within-compartment contacts in i;
14                 network = Integrate network and subnetwork;
15          else
16                 subnetwork = GenericEngine (i);
17                 network = Integrate network and subnetwork;
18          end if
19   end for
20
21   if m is top-layer
22          criteria = obtain selection criteria of m;
23          selected = select the eligible from the population based on criteria;
24          selected = filter out the exempted in the selected;
25          selected = match the size between selected and network;
26          network = map the indices of network to the indices of selected;
27          [network = rewire multi-edge in network;]   % optional operation
28   end if
29
30   return network;
```

Figure 2. Pseudo Codes of Contact Network Generation

Structure construction (lines 5-6) parses the parameters from structure descriptors and construct a set of compartments *unit[]* conforming to the prescribed statistics. We have not taken the actual indices of people (indices of individual persons in the entire population or in the complete contact network) into account yet at this stage. Instead, we assume that there are sufficient people filling up all the compartments in component m, with the "local indices" from 1 to total population within m. The advantage is twofold: (a) a compartment can be represented by two numbers (starting index and compartment size) instead of a rectangular matrix of size equal to population in the compartment. It is not only resource-efficient, but also reduces the communication cost in the parallelized HPCgen. (b) The local indices are in fact the row/column indices of matrix. Using the native matrix indices could simplify the operations of connecting edges between individual elements of matrix.

After constructing the compartments, the next stage is **contact generation** (lines 11-19). In this stage, HPCgen generates the contact edges between the individual nodes based on some degree sequences. Given the degree sequences computed from Poisson distribution (or any distribution else), HPCgen connects the edges stochastically with the probability based on the degree of end-nodes. For within-compartment

contacts, such implementation is straightforward. For cross-compartment contacts, an additional procedure is taken to ensure start-nodes and end-nodes of any contact edges are from the distinct compartments.

Once contact generation is over, HPCgen steps to the final stage of population mapping (lines 21-28). Recall that we do not consider the actual indices of people during the stage of structure construction. Without the indices in the entire population, there is no way to integrate those "sub" contact networks together. Population mapping is to complement the lack of uniform indices by mapping the local indices in the "sub" contact networks to the global indices in the "complete" contact network.

Population mapping can be further divided into four steps. 1) **Population selection** is to select the eligible persons for a particular component. The selection criteria are a set of conditional rules in the structure descriptors, which define the eligibility. For example, age-based criteria can be used to define the enrollment to schools. 2) **Exclusion filtering** is to filter out the exempted persons from the above selection. For example, if a person is selected to be a patient, he/she will be assumed to stay in the hospital ward and not to participate in any of schools, workplaces, shopping malls and public transport components. However, his/her contacts within household are retained because the visits from family members are expected. 3) **Mapping** is to map the index of every node in the "sub" contact network to the global population. Such mapping could be one-to-one (e.g. a student can enroll to only one school) or many-to-one (e.g. a shopper can visit multiple shopping malls during his/her infective period). To ensure the mapping is stochastic, index shuffling is implemented. 4) **Multi-edge rewiring** is an "optional" step to rewire the extra edges connecting to the same pair of nodes. A pair of nodes might be connected by multiple edges, which are accumulated in the contact generation of different components. However, percolation-based analytical modeling, one of the major applications of contact networks, cannot handle those multiple edges in its formulation [4]. Thus, we have to rewire the extra edges to other nodes before applying the analytical models. The rewiring is done by iterative re-mapping between the network nodes and the global populations until all multiple edges are eliminated. During the process, the original degree distribution of contact network remains constant.

C. *Parallelization*

Recall the goal of our HPCgen is to automate the repeated generations of contact networks of large urban cities. As it is a time-consuming task, building an efficient contact network generator is far more important. In the particular work, we leverage on MPI techniques to speed up the generation process.

In HPCgen, contact generation in any compartment is peer-independent and parent-dependent. For example, contact generation in a class compartment is independent of other class compartments, and dependent of its upper-layer school compartment which determines the size of individual class compartments. Therefore, we can distribute the top-layer compartments to the parallel processor nodes and compute concurrently. At the end of operations, we collect back all "sub" contact networks from all the processor nodes and form the complete contact networks by integration. This design can scale the parallelization up to the number of top-layer compartments, meaning if we have k schools in the model, maximally we can assign every school to a distinct processor node; so totally we can use up to k processor nodes for parallelization.

However, using more processor nodes does not necessarily lead to higher speedup. Parallelization could introduce the extra overheads in coordination and communication, which offset some fraction of speedup. In order to reduce the communication overheads, besides local indexing technique introduced in the previous section, we design a relay-like mechanism to achieve faster collection of network matrices from individual processor nodes. Instead all child processor nodes (child-nodes) send their matrices to the master processor node (master-nodes) at the same time (causing the bottleneck at the master-node), half of active processor nodes are nominated as sub-master-nodes. Each of sub-master-nodes manages one distinct child-node. During the collection process, every sub-master-node collects the network matrix from its child-node and integrates the matrix with its own. Then all child-nodes are deactivated. The above process is repeated until only master-node is active. By such relay-like collection, we reduce the time complexity to $O(\log_2 n)$, where n is the number of parallel processor nodes.

IV. EMPIRICAL RESULTS

We implemented the HPCgen in Octave [25] with support of MPITB [26] library. For efficiency purpose, some of non-matrix-related modules were coded in C++ and dynamically linked to Octave. Our experimental platform is an Intel 3GHz 2x quad-core machine with total 32 GB memory.

A. *Contact Network of Singapore*

The experiments were to build contact networks for Singapore city of 3.4-million population (residential population in Singapore, year 2000). We obtained the demographic data from the Singapore census of year 2000 [27], acquired the structure descriptors about schools, hospitals, workplaces, public transport and shopping malls from either government authorities or private organizations. Additionally, we conducted 1040 pieces of public surveys to collect the contact behavior profiles of Singapore residents.

Fig 3 plots the degree distribution of a Singapore contact network generated by HPCgen (recall that six components are included in the current setting: *households*, *schools*, *hospitals*, *workplaces*, *shopping malls* and *public transport*). The average degree of contacts is 24.1875. The two peaks occurs at degree 14 and 55, representing the typical degrees of general public and students (students are the most densely connected group of people in the Singapore contact network of current settings) respectively.

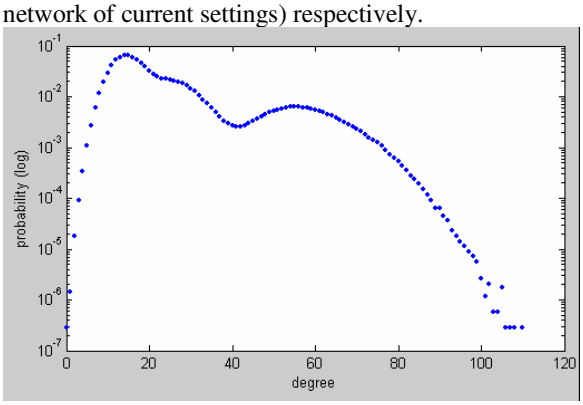

Figure 3. Degree distribution of contact network of Singapore (semi-log)

As contact generation is a stochastic process, we repeated the generations by 100 times with the same input data. Fig 4 plots the degree distributions of five contact networks out of 100 generations. We observed the curves of degree distributions remained stable from degrees 4 to 92 among the five generations. The fluctuations happened in less than degree 4 or greater than degree 92, especially for the degrees greater than 100. That matches with the intuition that both extremely small and large degrees are the mostly affected by stochastic events. The rest of generations confirm such observations too. We also observed that the degree distribution remained stable when compartment size varied by small fraction, showing the structural robustness.

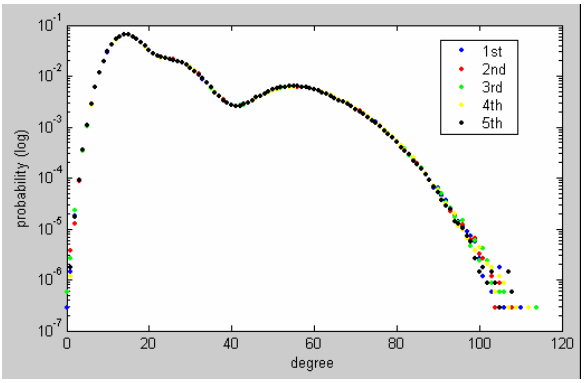

Figure 4. Degree distributions of multiple generations of contact networks

B. *Time Efficiency*

With the current settings, it requires around ten minutes to generate a Singapore contact network. That might not seem substantial if the generation is once only. However, when repeating the generations by hundreds of times (as the previous section did), it could cost nearly a day to complete. Such considerations motivated us to parallelize HPCgen for speedup.

With the same inputs, we measured the runtime of the parallelized HPCgen for 1, 2, 4, 6 and 8 processor nodes; and at each setting, 20 generations were repeated to take the average runtime. As shown in Fig 5, we observed the average runtime of HPCgen fell in a power function from 10.22 minutes to 62.17 seconds (~9.87 time speedup) as the number of processor nodes increases.

Our results outperformed the theoretical limit of linear speedup (Amdahl's law [28]), suggesting the performance gain might not be gained from the parallelization only. One possible explanation is the impact of runtime memory management. During the contact network generation in HPCgen, the frequent operations of matrix expansion and concatenation trigger the intensive memory reallocation in size of gigabytes. It might lead to a significant overhead by triggering swapping during the heavy usage of memory. By distributing top-layers compartments to different processor nodes, the reduction of resource consumption in individual processor nodes lower the memory usage. As a result, the chance of swapping and the overhead reduce accordingly.

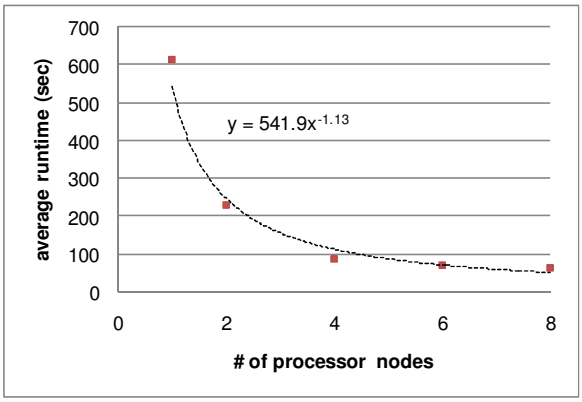

Figure 5. Average runtime decreases in a power function when number of processor nodes increases

V. CONCLUSION

Modeling and simulation allow us to have quantitative understanding about spreading dynamics of infectious diseases. The recent emerging contact network based modeling addresses the lack of network topology in the traditional mathematical modeling, leading to a better representation of our heterogeneous population. Moreover, the inclusion of community

structures in contact networks approximates the hierarchical social groupings in the real world, which might exert direct impact to clustering property of network structure.

In the literature, there are relevant works about generate social networks such as random networks and bipartite networks. However, there is no particular work about how to efficiently generate large people-to-people contact networks with hierarchical community structures for epidemiological studies. Our HPCgen fills this blank as a fast and generic contact network generator for large urban cities. By introducing parallelization techniques, we are able to reduce the generation time by 90%. To maximize the benefit brought by speedup, HPCgen is further equipped with automation scripts to repeat network generations as users command. This feature is especially useful in intervention studies, which require hundreds of contact networks in order to reflect the respective structural changes as a result of individual interventions. The overall saving of computation time could be far more substantial.

The contact networks generated from HPCgen can be used in both analytical models and agent-based simulations. The analytical models could extract the topological properties of networks and produce the statistical risk assessments to the epidemics in an instant manner; and the agent-based simulations could run on top of contact networks to simulate the time evolution of disease spreading with easy-interpretable visualization and easiness of manipulation. The versatile applications of contact networks ensure our HPCgen to be a valuable tool in the epidemiological studies of infectious disease outbreaks in large urban cities.

ACKNOWLEDGEMENT

This research work is supported by Biomedical Research Council of Singapore Grant (grant no. 06/1/21/19/457).

REFERENCE

[1] R. May and R.M. Anderson, *Infectious Diseases of Humans: Dynamics and Control*, Oxford University Press, 1992.

[2] O. Diekmann and J.A.P. Heesterbeek, *Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation*, Wiley, 2000.

[3] F. Brauer and C. Castillo-Chavez, *Mathematical Models in Population Biology and Epidemiology*, Springer, 2001.

[4] L.A. Meyers, B. Pourbohloul, M. Newman, D.M. Skowronski, and R.C. Brunham, "Network theory and SARS: predicting outbreak diversity," *Journal of Theoretical Biology*, vol. 232, Jan. 2005, pp. 71-81.

[5] M.E.J. Newman, "Spread of epidemic disease on networks," *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 66, Jul. 2002, p. 016128.

[6] A. Vazquez, "Spreading dynamics on heterogeneous populations: multitype network approach," *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 74, Dec. 2006, p. 066114.

[7] Y. Moreno, R. Pastor-Satorras, and A. Vespignani, "Epidemic outbreaks in complex heterogeneous networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 26, Apr. 2002, pp. 521-529.

[8] R. Pastor-Satorras and A. Vespignani, "Epidemic Spreading in Scale-Free Networks," *Physical Review Letters*, vol. 86, Apr. 2001, p. 3200.

[9] C.P. Warren, L.M. Sander, and I.M. Sokolov, "Firewalls, Disorder, and Percolation in Epidemics," *cond-mat/0106450*, Jun. 2001.

[10] G. Abramson and M. Kuperman, "Social games in a social network," *Physical Review E*, vol. 63, Feb. 2001, p. 030901.

[11] F.C. Coelho, O.G. Cruz, and C.T. Codeço, "Epigrass: a tool to study disease spread in complex networks," *Source Code for Biology and Medicine*, vol. 3, 2008, p. 3.

[12] S.Y. Del Valle, P.D. Stroud, J.P. Smith, S.M. Mniszewski, J.M. Riese, S.J. Sydoriak, and D.A. Kubicek, "EpiSimS: epidemic simulation system," *Los Alamos, NM: Los Alamos National Laboratory*, 2006.

[13] "List of cities proper by population."

[14] J.M. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A.S. Tomkins, "The web as a graph: Measurements, models and methods," *Lecture Notes in Computer Science*, 1999, pp. 1-17.

[15] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, E. Upfal, I. Center, and C.A. San Jose, "Stochastic models for the web graph," *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, 2000, pp. 57-65.

[16] C. Cooper and A. Frieze, "A general model of web graphs," *Random Struct. Algorithms*, vol. 22, 2003, pp. 311-335.

[17] A.L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, 1999, p. 509.

[18] F. Chung and L. Lu, "Connected Components in Random Graphs with Given Expected Degree Sequences," *Annals of Combinatorics*, vol. 6, Nov. 2002, pp. 125-145.

[19] E.A. Bender, E.R. Canfield, and B.D. McKay, "The asymptotic number of labeled connected graphs with a given number of vertices and edges," *Random structures and algorithms*, vol. 1, 1990.

[20] B. Bollobas, *Random graphs*, Cambridge University Press, 2001.

[21] M. Molloy, B. Reed, and E. Combinatoire, "A critical point for random graphs with a given degree sequence," *Random Structures and Algorithms*, 1995.

[22] S. Eubank, V.S.A. Kumar, M.V. Marathe, A. Srinivasan, and N. Wang, "Structural and algorithmic aspects of massive social networks," *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2004, pp. 718-727.

[23] Y. Saad, "SPARSKIT: a basic tool kit for sparse matrix computations."

[24] P.S. Pacheco, *Parallel Programming with MPI*, 1997.

[25] J.W.Eaton,"Octave", *http://www.gnu.org/software/octave/*.

[26] J. Fernández, M. Anguita, E. Ros, and J. Bernier, "SCE Toolboxes for the Development of High-Level Parallel Applications," *Computational Science – ICCS 2006*, 2006, pp. 518-525.

[27] "Singapore Census of Population 2000 Publications," *http://www.singstat.gov.sg/pubn/census.html*.

[28] G.M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Readings in computer architecture*, Morgan Kaufmann Publishers Inc., 2000, pp. 79-81.