Structural and Algorithmic Aspects of Massive Social Networks

Stephen Eubank* V.S. Anil Kumar* Madhav V. Marathe* Aravind Srinivasan† Nan Wang‡

Abstract

We study the algorithmic and structural properties of very large, realistic social contact networks. We consider the social network for the city of Portland, Oregon, USA, developed as a part of the TRANSIMS/EpiSims project at the Los Alamos National Laboratory. The most expressive social contact network is a bipartite graph, with two types of nodes: *people* and *locations*; edges represent people visiting locations on a typical day. Three types of results are presented. (i) Our empirical results show that many basic characteristics of the dataset are well-modeled by a random graph approach suggested by Fan Chung Graham and Lincoln Lu (the *CL-model*), with a power-law degree distribution. (ii) We obtain fast approximation algorithms for computing basic structural properties such as clustering coefficients and shortest paths distribution. We also study the dominating set problem for such networks; this problem arose in connection with optimal sensor-placement for disease-detection. We present a fast approximation algorithm for computing near-optimal dominating sets. (iii) Given the close approximations provided by the CL-model to our original dataset and the large data-volume, we investigate fast methods for generating such random graphs. We present methods that can generate such a random network in near-linear time, and show that these variants asymptotically share many key features of the CL-model, and also match the Portland social network.

The structural results have been used to study the impact of policy decisions for controlling large-scale epidemics in urban environments.

*Basic and Applied Simulation Science (CCS-5) and National Infrastructure and Analysis Center (NISAC), Los Alamos National Laboratory, MS M997, P.O. Box 1663, Los Alamos, NM 87545. Email: {eubank,anil,marathe}@lanl.gov. This work is supported by the Department of Energy under Contract W-7405-ENG-36.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Email: srin@cs.umd.edu.

‡Department of Computer Science, University of Maryland, College Park, MD 20742. Most of the work was done while visiting Los Alamos National laboratory. Email: nwang@cs.umd.edu.

1 Introduction

The rapidly expanding population of today's towns and cities has resulted in very high population densities and social connectivity. Understanding the urban social-contact structure is critical for social scientists, urban planners, infrastructure companies, and governments. For example, the spatial distribution of population in a city, people's movement-patterns and their phone-calling patterns have a direct bearing on how wire-line and wireless infrastructure providers design their networks. Similarly, the social contact network also determines the spread of an infectious disease. For instance, one of the astonishing features of the recent SARS outbreak was the speed and the extent to with which the epidemic spread: this is a demonstration of how "connected" the world has become. For many societies, this raised the important questions of how to detect the disease quickly, and how to control it (by quarantining, vaccination etc.). Because of the significant logistics involved, this needs to be done with as little cost as possible. This work does an empirical and analytical study of various structural and algorithmic issues of massive social networks.

Our work is a part of the EpiSims project. EpiSims is a simulation-based analytical tool to study the spread of infectious diseases in an Urban environment being developed at the Los Alamos National Laboratory [11]. In order to understand the spread of contagious diseases, we need a realistic representation of a social network. The Transportation Analysis and Simulation System (TRANSIMS) developed at LANL produces estimates of the social network in a large urban area based on the assumption that the transportation infrastructure constrains people's choices about what activities to perform and where to perform them [15]. TRANSIMS produces locations of all travelers on a second-by-second basis; see [15] and `http://transims.tsasa.lanl.gov` for more information. EpiSims is a tool for simulating the spread of epidemics at the level of individuals in a large urban region, taking into account realistic contact patterns and disease transmission characteristics. It integrates models for disease propagation within a host, transmission between hosts, and contact patterns among hosts to create a realistic microscopic epidemic simulation. It provides estimates of both the geographic and demographic distribution of disease as a function of time. EpiSims provides a natural representation for assessing the capability of public health infrastructure to defend urban populations against disease outbreaks. Tools such as vaccination campaigns, quarantine and isolation, and contact tracing can be modeled easily within the simulation, even if they are demographically targeted (e.g., immunizing children).

Structural analysis of social contact networks is an integral part of EpiSims and TRANSIMS. There are three key parameters to develop: (i) new graph properties that serve as structural invariants and model a given policy/design question at hand, (ii) mathematical theories that explain the evolution of these graphs and also serve as models generating random ensembles of such networks, and (iii) fast computational methods for solving problems arising in (i) and (ii). For example, recent results have shown that many real-world social and infrastructure networks cannot be modeled by the Erdős-Renyi model $G(n,p)$ [3]. New mathematical models have been proposed for generating generic "real world networks". Recent work on structural analysis of large real-world networks started with experimental work on the structure of the Internet/WWW (see, e.g., [2, 7, 18]). One of the striking results of such studies was the power-law nature of the degree distribution. Since models such as $G(n,p)$ do not have this property, these studies have sparked research on theoretical models to explain such properties. These models are broadly of two types: (i) incremental models for constructing such graphs, using primitives such as preferential attachment (see, e.g., [5, 6, 9, 20, 21]): each new vertex chooses its set of neighbors randomly according to some function of the degrees of previous vertices; and (ii) assume a given degree sequence, and generate random graphs either by putting each edge independently, with a probability dependent on the degrees of the end-points [8], or generating a graph uniformly at random from the space of graphs with the given degree-sequence [22].

Structural properties of networks[1] that have become classical include: degree distribution, clustering coefficient distributions, shortest-path distribution, and connected components. Most of the work on the above models has been toward understanding such structural properties [9, 6, 8, 22]. They serve as important invariants much like physical laws that serve as invariants for physical systems. In addition to such structural results, there has been much work in classical random graph theory on algorithms for problems that are usually NP-hard in the worst case [17]. No such studies seem to have been carried out yet for random graph models for social networks. Combinatorial problems such as dominating sets, cuts and flows are useful in this setting. One challenge here is that these graphs are typically very large (of the order of a million nodes or more), since they represent models for social interaction, and computing even simple quantities on such massive graphs is nontrivial. This raises the need for very fast algorithms. Another important problem that has not received attention in this context so far, is that of efficient generation of these graphs. For instance, in the model of [8] (the *CL-model*), implementation requires quadratic time, even when the average degree is small. For very large graphs with millions of nodes, a fast-generation mechanism that takes time of the order of the number of edges, but still produces graphs with meaningful structure, would be very useful (e.g., for sensitivity analysis). The starting point of our work is a realistic social contact structure based on data from Portland, Oregon, USA. The main component of this dataset that is relevant to us is a social contact network $G(P \cup L, E)$, which is a bipartite graph with a set P of people and a set L of locations.[2] An edge (p, ℓ) is present if person p visits location ℓ on a typical day. The dataset is massive: $|P| \sim 1.6$ million; $|L| \sim 1.8 \times 10^5$, and $|E| \sim 6$ million. Our contributions are threefold.

(a) Fast Generation of Network Models. One key idea of the CL-model for massive graphs [8], is that each node u has a given "weight" $d(u)$ corresponding to its expected degree, such that nodes tend to attract edges with probability proportional to their weight. (In the WWW context, one such situation is where popular sites tend to get even more inward links over time.) In our context, the CL model would take the sets $\{d(p) : p \in P\}$ and $\{d(\ell) : \ell \in L\}$ as input, where $\sum_p d(p) = \sum_\ell d(\ell) = \sigma$, and *independently* put the edge (p, ℓ) with probability $d(p) \cdot d(\ell)/\sigma$; note that the expected degree of any node in $P \cup L$ equals its given weight. Such a weight-based model seems plausible in the social-network case also: a popular tourist-spot or massive office-structure will likely get even more visitors, and a person such as a bus-driver making long trips is likely to visit new locations (within limits). We discuss the empirical validity of this model in (c) below. Note that this graph-generation takes time $\Theta(|P| \cdot |L|)$, which is infeasible in our case: such models typically have to be generated independently many times, e.g., for simulations. Thus, we first present two natural

[1] See `http://cnls.lanl.gov/networks` for a recent workshop on this and related topics.

[2] For diseases such as smallpox L can be restricted to buildings; the contact times in city public transport are usually inadequate.

variants of the CL-model, **FastGen-1** and **FastGen-2**, and show how they can be implemented in time nearly-linear in the number of edges. We show analytically that **FastGen-1** asymptotically matches the CL-model for various parameters of interest, and give reasons for why **FastGen-2** may not. The empirical validity of these models is discussed in (c).

(b) Fast Algorithms. Given the outbreak of a disease, one way of detecting it is to place sensors at various locations (e.g., entrances to schools and malls) so that any infected individual can be detected. Since placing and monitoring sensors is an expensive task, a policy planner would like to place as few sensors as possible, but still detect the disease (or at least a vast majority of the non-quarantined infected individuals). This is the minimum dominating set problem: find a smallest subset of L so that each person, or at least a $(1-\epsilon)$-fraction of P visits one such location. We are interested in the *fast* computation of a near-optimal-sized $L' \subseteq L$ that dominates P, or at least a $(1-\epsilon)$-fraction of P (we call it the $(1-\epsilon)$-dominating set problem), for a given constant ϵ. Dominating set cannot be approximated to within $(1-\Omega(1))\ln|P|$ [19]; even the standard greedy algorithm, which takes nearly quadratic time, is infeasible for our large networks. For the contact graph in EpiSims, we show that the following FastGreedy algorithm gives a very good approximation to the $(1-\epsilon)$-dominating set problem: select the locations in decreasing order of degrees until a $(1-\epsilon)$-fraction of the people are dominated. It turns out that this heuristic works well for us because of the high overlap ratios of the locations (defined in Section 2) and high values imply low overlaps. For our data and our models, FastGreedy takes a few seconds, as compared to a few hours for the standard greedy algorithm. We show that FastGreedy gives $(1+o(1))$–approximations (to the $(1-\epsilon)$-dominating set problem), for the CL-model and for our **FastGen-1** model. We also show that for these models, an even simpler sampling based algorithm is good: sample each person independently and choose a subset of locations visited by this sampled set having a certain bound on the sum of degrees. This is really practical for a policy planner, because we only need to know the degrees with respect to the sample. We also need fast algorithms for computing various parameters of our graphs, e.g., clustering coefficients and shortest-path distribution. It is simple to show that sampling a "small" number of vertices yields the former with high accuracy. For the latter, we show the following. Given an arbitrary graph $G=(V,E)$ with $|V|=n$ and a parameter ϵ, suppose we aim to estimate with high accuracy, the number $SPD(i)$ of pairs (u,v) that are at distance i from each other, only for those i for which $|SPD(i)| \geq \epsilon \cdot n^2$ (e.g., in practice, those i that occupy less than 0.1% of the shortest-path distribution may be irrelevant). We show that a small number of single-source shortest paths computations run from randomly generated sources, is sufficient.

(c) Empirical Evaluations. We carry out an extensive empirical analysis of our results, to show that: (i) the algorithms for computing structural properties and generating graphs run very fast and provide very good approximations, and (ii) the CL-model and the fast-generation variant(s) provide good approximations to the real social network for most of the graph parameters studied here. Thus, the CL-model and its variants can be valuable for carrying out sensitivity analysis of our methods as they relate to policy implications.

The results presented here have been combined with the simulations in EpiSims in two recently concluded studies done by Los Alamos for the Dept. of Homeland Security. Recently, there has been an active debate over adopting response strategies for controlling smallpox [14, 4, 12]. The results in [13] show the importance of early detection and aggressive quarantining/vaccination to control the spread of diseases in cities such as Portland. See [4, 12] and our website `http://www.ccs.lanl.gov/ccs5/index.shtml`.

2 Preliminaries

We will work with the bipartite graph $G(P \cup L, E)$ defined in the introduction, as well as some related graphs. As pointed out in Section 5, the degree-distribution of L is well-approximated by a power law with exponent $\beta > 2$: i.e., the number of locations in L with degree i is close to $n_i = \frac{c|L|}{i^\beta}$, where c is a normalization constant. In the actual dataset, $\beta \sim 2.8$; we will work with an arbitrary constant $\beta > 2$. We let d_0 and d_1 denote the minimum and maximum location degrees, respectively; we will have $d_0 \ll d_1$, i.e., the location degrees exhibit large variations. The people degrees are small as expected, since a person cannot visit too many locations on a typical day; they are sharply concentrated around their average value, which will throughout be denoted w_p (and is approximately 4 in our dataset). Counting the numbers of edges from P and from L, we get

$$|P| \cdot w_p \sim \frac{\beta-1}{\beta-2} \cdot |L| \cdot d_0. \tag{2.1}$$

In urban settings, it is reasonable to assume that $|L|$ grows at a much slower rate than $|P|$ (after a city becomes larger than some critical size, one starts having lots of high-rise locations). Since w_p and $\beta > 2$ are small constants, (2.1) then implies that d_0 must be large. So we will assume for our mathematical models that $d_0 =$

$\omega(1)$: i.e., a function of $|P|$ that increases unboundedly. (A typical choice could be some not-too-fast-growing function, such as polylog($|P|$).) We will also consider two other graphs induced by G. The people-people network, G_P is the projection of G onto P: the nodes of G_P are P and (p, p') is an edge if there is a common location $\ell \in L$ such that $(p, \ell), (p', \ell) \in E(G)$. G_P is a key network since it corresponds to contact between people, and hence represents the spread of epidemics through personal contact. The location-location graph G_L is also defined similarly.

When developing random graph models of $G(P \cup L, E)$ (related to the CL-model) given by the degree-sequences of P and L, we will make the assumptions of the above paragraph, for some arbitrary constant $\beta > 2$. Recall from Section 1 that the CL-model will take such a degree-sequence and then generate a random graph with edges put independently, with probability proportional to the product of the expected degrees. For a node $v \in P \cup L$, $N(v)$ is the set of nodes adjacent to v. For a set $V \subseteq P \cup L$, $N(V)$ is defined as $N(V) = \bigcup_{v \in V} N(v)$; also, $d(V)$ is defined to be $d(V) = \sum_{v \in V} d(v)$. For a set $L' \subseteq L$, the overlap ratio of L' is defined to be $|N(L')|$ divided by the sum of degrees of nodes in L'. The overlap ratio is a quantity in $(0, 1]$ and a larger value indicates lower overlap. The clustering coefficient [25] is a measure popular with social scientists to study interaction graphs. The clustering coefficient $C(v)$ for node v is defined to be the number of edges (u, w) such that $u, w \in N(v)$, divided by $\binom{deg(v)}{2}$; $deg(v)$ denotes the degree of v. For any integer i, $C(i)$ is the average clustering coefficient of vertices of degree i. The clustering coefficient is 0 for every node in the bipartite graph, and therefore, we consider it only for G_P.

3 Fast Generation of our Massive Graphs

As explained in Section 1, a naive generation of random instances using the model of [8] takes $|P| \cdot |L|$ steps, because we have to try out each tuple $(p, \ell) \in P \times L$. The number of edges, on the other hand, is $O(|P| \cdot w_p)$, which is essentially $O(|P|)$. This is a lot smaller than $|P| \cdot |L|$ for the scale of data sets we are looking at, and naturally leads to the question whether instances can be generated in nearly $O(|P| \cdot w_p)$ time. We describe a method to do this in this section.

The Fast-Generation Models. Formally, we are given two disjoint sets P and L, and two positive integral degree sequences $D(P) = \{d(p) : p \in P\}$ and $D(L) = \{d(\ell) : \ell \in L\}$ such that $\sigma = \sum_{p \in P} d(p) = \sum_{\ell \in L} d(\ell)$ and $\max_{p \in P} d(p) \cdot \max_{\ell \in L} d(\ell) \leq \sigma$. Let the random variable $X_{p,\ell}$ denote the event that there is an edge between $p \in P$ and $\ell \in L$. **FastGen-1** will be the following model, where the sequence of location-degrees equals $D(L)$ with probability 1. We will guarantee the following properties, by a careful implementation of the approach of [24]: (a) $\forall p \in P, \forall \ell \in L$, $\Pr[X_{p,\ell} = 1] = \frac{d(p) \cdot d(\ell)}{\sigma}$; (b) $\forall \ell \in L$, $\Pr[|\{p : X_{p,\ell} = 1\}| = d(\ell)] = 1$; (c) $\forall \ell \in L$, the following "negative correlation" properties hold for all subsets $P' \subseteq P$:

$$\Pr[\bigwedge_{p \in P'} (X_{p,\ell} = 0)] \leq \prod_{p \in P'} [X_{p,\ell} = 0],$$
$$\Pr[\bigwedge_{p \in P'} (X_{p,\ell} = 1)] \leq \prod_{p \in P'} [X_{p,\ell} = 1].$$

The first property implies that each edge is put in with the right probability. The second property ensures that the degrees of nodes in L are equal to the expected degrees. (Our algorithm can be trivially modified so that degrees in P are satisfied exactly, instead: that model is called **FastGen-2**.) The third property allows us to use Chernoff-like bounds, and we will use it later to show that several measures like overlap ratios, clustering coefficients etc. are preserved in this model.

The FastGen Algorithm. The discussion here is for model **FastGen-1**. Algorithm FastGen is a careful implementation of the approach of [24] and is described in Figure 1.

Lemma 3.1. *For any positive integer q and any index k, $\frac{\lfloor q/2 \rfloor}{q} < \sum_{i=0}^{q-1} \frac{w(block_{k+i})}{\sigma} \cdot \frac{\max_{\ell \in L} d(\ell)}{q} \leq 1$. Also, the total number of blocks is at most $2(\max_{\ell \in L} d(\ell) + 1)$.*

Lemma 3.2. *For each $\ell \in L$, the number of super blocks in step 4(a) of* FastGen *is at most $2(d(\ell) + 1)$.*

Lemma 3.3. *Algorithm* FastGen *satisfies the requirements of the fast generation model and runs in time $O(\sigma \cdot \log |P|)$.*

3.1 Analytical Validation of FastGen-1

We now briefly explain why model **FastGen-1** is a close approximation to that of [8] for the degree-distribution of the people-people graph, clustering coefficient, overlap ratios, etc. Similar remarks also hold for the distribution of shortest-path lengths, and we will provide detailed proofs in the full version.

Consider the degree-distribution of the people-people graph. Recall that a general model we are employing for the bipartite graph (P, L) is where: (a) the weight of any person is bounded by a constant C, and (b) the weight of the locations follow a power-law with exponent $\beta > 2$ with weights running in the range $[d_0, d_1]$, where d_0 is a slowly-growing function of $|P|$ such as polylog($|P|$). Since the number of locations with

Algorithm FastGen

1. Order the elements in P in an arbitrary order, $(p_1, p_2, \cdots)$.
2. Compute prefix sums $ps_i = \sum_{j=1}^{i} d(p_j)$ for $1 \le i \le |P|$, and define $ps_0 = 0$.
3. Partition the elements in P into a sequence of z blocks $block_1, block_2, \cdots block_z$ satisfying the following properties.
   (a) Each block consists of a (contiguous) subsequence of P, e.g., $block_k = (p_{j_k}, p_{j_k+1}, \cdots, p_{j_{k+1}-1})$.
   (b) Let $w(block_k) = \sum_{p_j \in block_k} d(p_j)$. The last block satisfies $0 < w(block_z) \cdot \max_{\ell \in L} d(\ell) \le \sigma$.
   (c) For each $k < z$, let p_{j_k} be the first element in $block_k$, the k-th block satisfies
   $$\sigma - d(p_{j_{k+1}}) \cdot \max_{\ell \in L} d(\ell) < w(block_k) \cdot \max_{\ell \in L} d(\ell) \le \sigma; \tag{3.2}$$
4. For each $\ell \in L$, choose $d(\ell)$ edges incident on ℓ by the following steps.
   (a) Let $q_\ell = \lfloor \frac{\max_{\ell' \in L} d(\ell')}{d(\ell)} \rfloor \ge 1$. Group the blocks into super-blocks:
   $$\begin{aligned} Block_1 &= (block_1, block_2, \cdots, block_{q_\ell}), \\ Block_2 &= (block_{q_\ell+1}, block_{q_\ell+2}, \cdots, block_{2 \cdot q_\ell}), \\ &\cdots \end{aligned}$$
   and so on. By Lemma 3.2, there are at most $2(d(\ell)+1)$ super-blocks.
   (b) Let the random variable Y_i denote the event that there is an edge between ℓ and the super-block $Block_i$. Run the algorithm of [24] to determine if $Y_i = 1$ or 0.
   (c) Suppose $Block_i = (p_{t_i}, p_{t_i+1}, \cdots, p_{t_i+j})$ which is a subsequence of the ordered people. If $Y_i = 1$, choose *one* $p \in Block_i$ with probability proportional to its weight $d(p)$ and add the edge (ℓ, p). This can be done by first choosing an integer r uniformly at random from the interval $[ps_{t_i-1}, ps_{t_i+j})$, locating $p_x \in Block_i$ by running binary search on the (increasing) sequence $(ps_{t_i}, ps_{t_i+1}, \cdots, ps_{t_i+j})$ and finding the index x such that $ps_{x-1} \le r < ps_x$.

Figure 1: Description of the algorithm for fast generation of random graphs.

degree d_1 must be nonzero, it can be shown using the definition of power-law that

$$d_1 \le O(d_0 \cdot |L|^{1/\beta}) \le O(|P|^{1/2-\Omega(1)}).$$

Consider a person p. Its number of neighbors is distributed asymptotically as Poisson$(d(p))$, both in the model of [8] and in **FastGen-1**. Since $d(p) \le C$, we have in particular that with high probability, the number of neighbors is at most $2 \ln |P| / (\ln \ln |P|)$. So, since each of these neighbors has degree at most $|P|^{1/2-\Omega(1)}$ as seen above, the total degree of these neighbors is at most $|P|^{1/2-\Omega(1)}$; thus, these neighboring locations of p will, in turn, have almost-disjoint neighborhoods in the set P, with high probability. Thus, the number of neighbors of p in the people-people graph is essentially the sum of the degrees of its neighbors, measured in the graph (P, L). This sum is sharply concentrated around its mean, as it is a sum of independent random variables with mean $\omega(1)$ in the model of [8]; in **FastGen-1**, the concentration holds trivially with probability 1 (since the location-degrees equal the prespecified degrees with probability 1).

A similar argument holds for the overlap ratios. As for the clustering coefficient of the people-people graph, recall that it equals $(\sum_{p \in P} C(p))/|P|$, where $C(p)$ is the number of edges $NE(p)$ in the neighborhood $N(p)$ of p in the people-people graph, divided by $\binom{N(p)}{2}$. Consider the random variable $C(p)$ for any fixed p, in both of our models of interest. As sketched above, the denominator $\binom{N(p)}{2}$ is highly concentrated around a value a; thus, $\mathbf{E}[C(p)]$ is essentially $(1/a) \cdot \mathbf{E}[NE(p)]$. Now, it can be shown that the main type of conditioning in the calculation of $\mathbf{E}[NE(p)]$, involves conditioning on a small constant number of edges incident on each of a set of locations L'. This conditioning poses no problems in the CL-model, due to its full independence. It essentially poses no problem in model **FastGen-1** either, since the locations make their edge-choices independently, and have expected degree $\omega(1)$. It can also be shown that **FastGen-2** is in general *not* a good approximation to [8] in the context of the above-seen parameters, basically because the weights $d(p)$ of the people are all small. In particular, conditional on the presence of an edge (p, l) in (P, L), the distribution of other edges incident on p becomes altered significantly in **FastGen-2**. We will present the details in the full version.

4 Fast Algorithms

4.1 Dominating Set The FASTGREEDY heuristic is the following simple algorithm for computing a dominating set. Consider the locations in L in nonincreasing order of $d(\cdot)$: $\ell_1, \ell_2, \ldots, \ell_{|L|}$ with $d(\ell_1) \geq d(\ell_2) \geq \ldots \geq d(\ell_{|L|})$, where $d(\cdot)$ is the given degree-sequence. Pick the smallest i such that $|\bigcup_{j \leq i} N(\ell_j)| \geq (1-\epsilon)|P|$ and take the subset $\{\ell_1, \ldots, \ell_i\}$ to be the dominating set (for dominating $(1-\epsilon)$-fraction of the people). For any $L' \subseteq L$, define $d(L') = \sum_{\ell \in L'} d(\ell)$. Consider first the case where $d(p)$ is the same for all $p \in P$ (recall that the people-degrees are highly concentrated around the mean). A key lemma that we can show for both the CL-model and the **FastGen-1**, is:

LEMMA 4.1. *There is a certain parameter $\delta = o(1)$ such that the following holds simultaneously for all $L' \subseteq L$ with high probability: if $d(L') \leq |P| \cdot \frac{\ln(1/\epsilon)}{1+\delta}$, then $|N(L')| < (1-\epsilon) \cdot |P|$, and if $d(L') \geq |P| \cdot \ln(1/\epsilon)$, then $|N(L')| \geq (1-\epsilon) \cdot |P|$.*

Using this, we can show that FASTGREEDY is a $(1 + o(1))$-approximation algorithm to the $(1-\epsilon)$-dominating set problem; by a similar argument, we get that it is an $O(1)$-approximation algorithm (with a small constant hidden in the $O(1)$) when the people-degrees vary somewhat.

Domination Using Sampling. FASTGREEDY and the usual greedy algorithm, require knowing the structure of the graph, and the degrees of locations. Once this data is available, these algorithms are easy to implement, but collecting such data is a non-trivial task. In practice, any simulation (such as EpiSims) builds models based on a small sample of data from the whole population. This motivates the following question: *can one get a good solution to the domination problem by only sampling a fraction of the data?* We show here that this is possible, using the following SAMPLEDGREEDY algorithm. Choose a set $S \subset P$ by sampling each person independently with probability α. Compute the set $N(S)$ of locations visited by S and choose the smallest set $L' \subset N(S)$ whose sum of degrees (restricted in S) is at least $\alpha|P| \cdot \ln(1/\epsilon)$. L' is output as the solution for the whole graph.

LEMMA 4.2. *Assume $d_0 \geq a \log|P|$, where a is a certain constant. For any sampling probability $\alpha \geq 2(\ln|P|)/d_0$, the solution L' chosen by the* SAMPLED-GREEDY *algorithm is also a $(1 + o(1))$-approximation solution to the $(1-\epsilon)$-dominating set problem of the whole graph, with high probability.*

4.2 Shortest Path and Clustering Coefficient Distribution Two basic statistics related to the people-people graph G_P are the average clustering coefficient (c.c.) $(\sum_{p \in P} C(p))/|P|$, and the distribution of shortest paths. The former is easy to estimate by standard sampling, since each $C(p)$ lies in $[0, 1]$. Choosing $\Theta(\log|P|)$ vertices at random and outputting their mean c.c., is easily shown via a Chernoff bound to give an accurate estimate. For shortest paths, consider estimating the values $SPD(\cdot)$ defined in Section 1, for the graph G_P. Suppose we only wish to do so (with some relative error δ) for all $|SPD(i)| \geq \epsilon|P|^2$. Simple sampling works again: choose each vertex-pair in an i.i.d. manner, so that the expected number of choices is of the form $\Theta((\epsilon\delta^2)^{-1} \cdot \ln(1/\epsilon))$, find the distance between each chosen pair, and output the distribution obtained. A similar sample complexity, using single-source shortest paths from random sources, can be shown using a tail bound from [23].

5 Experimental Analysis and Validation

We now present empirical results to strengthen the claims in Section 3.1. We also compare the results of the random graph models with the original social contact network for Portland. For this section we denote this network as the *Real-Network*. Recall, that the degree sequence of this network is used as input by the random graph models. All the experiments were done on a shared machine, Sun UltraSPARC-III, 750 MHz CPU, 8.0 GB main memory. In all our experiments, we ignored isolated vertices, i.e., vertices with degree 0. For example, in Table 1, the percentages of locations and people for the three models are the percentages of how many non-isolated locations and people are in the generated graph, compared to the Real-Network. In all random graphs that we generate, there is a giant connected component consisting of almost all vertices. By projecting the people-location bipartite graph into the people-people graph and the location-location graph, we examine the sizes of the giant components in these two graphs separately. Since the CL-model is expensive to run, we only generated a few instances by this model. For all the three random graph models we found that the values are sharply concentrated. Hence instead of taking averages, the data in Table 1 is for only one instance from each model.

We study the following measures for these random graph models and the EpiSims data. (i) *Degree distribution in the bipartite graph and in the induced people-people graph;* (ii) *Overlap ratios;* (iii) *Size of the giant component in the people-people graph and the location-location graph;* (iv) *Shortest paths and clustering coefficients;* (v) *Quality of the* FastGreedy *algorithm for domination.*

Table 1 compares some basic parameters between the Real-Network and the graphs generated by the random models. The size of giant indicates the number of people (locations) in the giant connected component of the people-people (location-location) graph. Note that **FastGen-1** preserves the number of locations and **FastGen-2** preserves the number of people. Both of them preserves the number of edges.

(a) Degree Distributions. Figure 2 compares the degree distributions of locations in the Real-Network and the random graphs generated by CL- and **FastGen-2** models. One can see from the figure that the three distributions are very close to each other. Note that **FastGen-1** preserves location degrees. Also note that a large part of the degree distribution of locations exhibits a power-law. This part starts at degree approximately 20 and ends at degree about 200. For locations of degree k in $[20, 200]$, the number of them is proportional to $\frac{|L|}{k^\beta}$, where $\beta \simeq 2.8$, i.e., $|L_k| = \frac{c|L|}{k^{2.8}}$, where $L_k = \{l \in L : deg(l) = k\}$, $20 \leq k \leq 200$, and $c \simeq 200$.

Next consider the degree distribution of the giant component of the people-people graph G_P. The experimental results and the theoretical analysis (Section 3.1) together confirm that the CL- and **FastGen-1** models match the Real-Network much more closely than **FastGen-2**.

(b) Overlap Ratios. We now study two variants of overlap ratios for locations in the bipartite graph. For any positive integer d, the *point_overlap_ratio*(d) is the overlap ratio for the set $L' = \{\ell \in L : d(\ell) = d\}$ and the *cumulative_overlap_ratio*(d) is the overlap ratio of the set $L' = \{\ell \in L : d(\ell) \geq d\}$. Note that all the overlap ratios are in $(0, 1]$. The higher the overlap ratio is, the better the FastGreedy algorithm performs for the dominating set problem. The results are shown in Figure 3.

From both experiments (Figure 3) and the theoretical analysis (Section 3.1) we can conclude that the overlap ratios of graphs generated by **FastGen-1** are close to the graphs generated by CL-model and both of them are close to the Real-Network. On the other hand, the overlap ratios of graphs generated by **FastGen-2** are much higher than the corresponding overlap ratios in graphs generated by the other two models and the real-Network. Thus we can expect that FastGreedy should perform much better for the graphs generated by **FastGen-2** than by the others.

(c) Shortest Paths and Clustering Coefficients. Two popular measures people study on social networks are the shortest path distribution and the clustering coefficient distribution. Often, these give important connectivity information for the contact networks. Both of these can be computed easily in polynomial time, but usual algorithms take $\Omega(|P|^2)$ time, which is infeasible for such large graphs. This motivates faster methods to approximate these distributions. We show simple random sampling based algorithms for these two problems, and show their empirical performance. For practical reasons, we examine the shortest paths and the clustering coefficients only in the giant component of the people-people graph.

(c1) Shortest paths. Since $\log_2 |P| \simeq 20$, we did seven experiments each by uniformly and independently sampling about a hundred vertices (i.e., people) in the EpiSims graph and computed the shortest-path spanning tree for each sampled vertex. The fraction of shortest paths of distance i in the giant-component is estimated by the fraction of paths of distance i in all the shortest-path spanning trees. Similarly we did the same experiments for the CL-model, **FastGen-1** and **FastGen-2**. For each model, nine experiments were done and in each experiment about a hundred vertices were sampled. The means and the standard deviations of the experiments for the EpiSims data and the three models are presented in Table 2. From the table we can see that the distance between most pairs of people in the giant connected component is 2 or 3.

(c2) Clustering coefficients. By the distribution of the clustering coefficients, we examine for each degree $i \geq 2$, the average clustering coefficient $C(i)$ in the giant component of the people-people graph. We present two fast algorithms to approximate the distribution of the clustering coefficients:

Fast-algorithm-1 provides a lower bound to the distribution of the clustering coefficients. For a $p \in P$, arbitrarily order the locations in $N(p)$ as $l_1, \cdots, l_d$, where $d = |N(p)|$ is the degree of p in the bipartite graph. Let $\hat{N}(l_1) = N(l_1) \backslash \{p\}$, and for $i = 2$ to d, let $\hat{N}(l_i) = N(l_i) \backslash \{p\} \backslash \bigcup_{j=1}^{i-1} \hat{N}(l_j)$. The lower bound of the clustering coefficient for p is $\frac{\sum_{i=1}^{d} |\hat{N}(l_i)|(|\hat{N}(l_i)|-1)}{(|N(N(p))|-1)(|N(N(p))|-2)}$, where $N(N(p))$ denotes the set of neighbors (in people) of the set $N(p)$ (in locations). Due to the high overlap ratio of the set $N(p) \subset L$, we can approximate people-people degree of p by $|E(N(p))| - |N(p)|$, where $E(N(p))$ is the set of edges incident on the set $N(p)$. Fast-algorithm-2 exploits these two properties.

Fast-algorithm-2 approximates the lower bound given

	Real-Network	CL	FastGen-1	FastGen-2
number of locations	181230	178746 (98.63%)	181230 (100%)	178668 (98.59%)
size of giant(locations)	181192	178571	181088	178611
number of people	1615860	1507234 (93.28%)	1507291 (93.28%)	1615860 (100%)
size of giant(people)	1615813	1507054	1507148	1615803
number of edges	6060679	6065637 (100.08%)	6060679 (100%)	6060679 (100%)
average degree (people)	3.7507	4.0227	4.0209	3.7507
time for generating a graph		$>$ 10 hours	$<$ 40 seconds	$<$ 30 seconds

Table 1: Comparing the basic graph theoretic parameters for the Real-Network and randomly generated graphs. The last row compares the time it requires to generate such graphs.

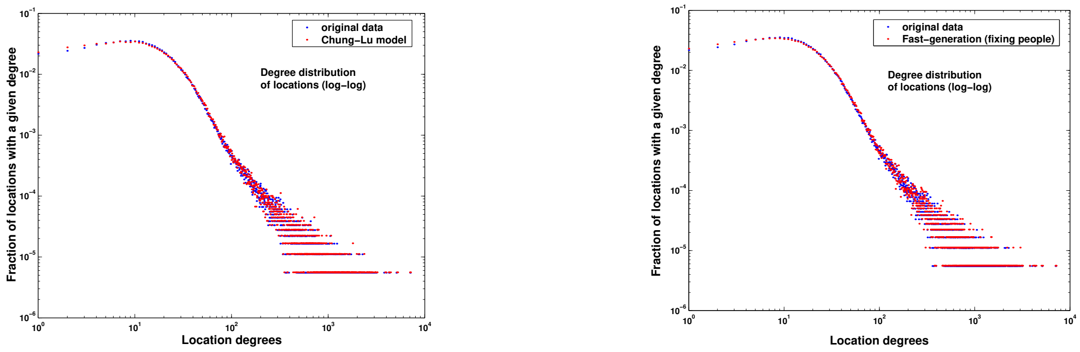

Figure 2: Degree distribution of locations.

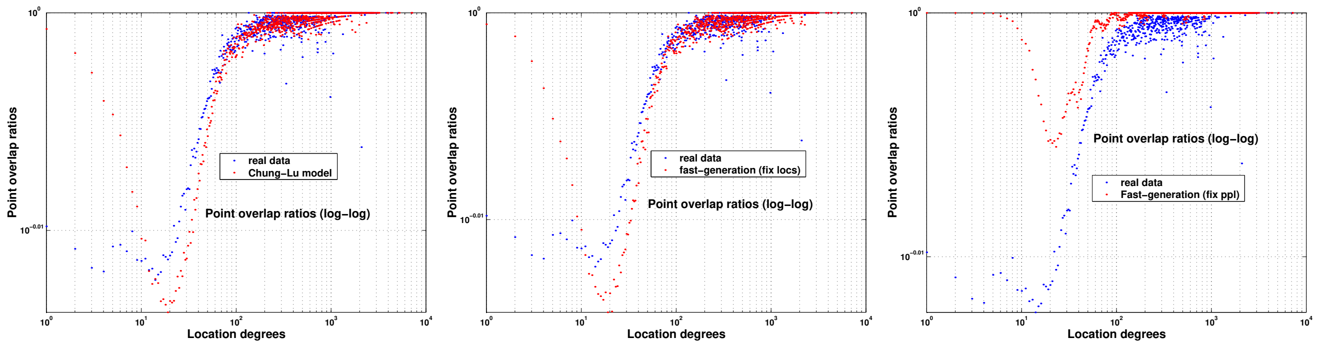

Figure 3: Point overlap ratios of the Real-Network and the three models.

		len=1	len=2	len=3	len=4	len=5	len=6
Real-Network	mean	8.0806×10^{-4}	0.2666	0.7126	0.0200	4.8687×10^{-5}	5.6341×10^{-9}
	std.	9.1226×10^{-5}	0.0149	0.0138	0.0049	8.8549×10^{-5}	9.8875×10^{-9}
CL	mean	9.7835×10^{-4}	0.4264	0.5665	0.0061	1.0460×10^{-6}	
	std.	1.1151×10^{-4}	0.0241	0.0223	0.0025	5.1511×10^{-7}	
FastGen-1	mean	0.0010	0.4161	0.5756	0.0072	4.6721×10^{-7}	
	std.	1.5816×10^{-4}	0.0300	0.0282	0.0028	2.0310×10^{-7}	
FastGen-2	mean	8.4945×10^{-4}	0.3622	0.6344	0.0026	2.9173×10^{-7}	
	std.	6.9460×10^{-5}	0.0177	0.0181	0.0019	5.3671×10^{-7}	

Table 2: Means and standard deviations of fractions of shortest paths of different lengths in the single-source shortest-path spanning trees sampled from the giant component of the people-people graphs.

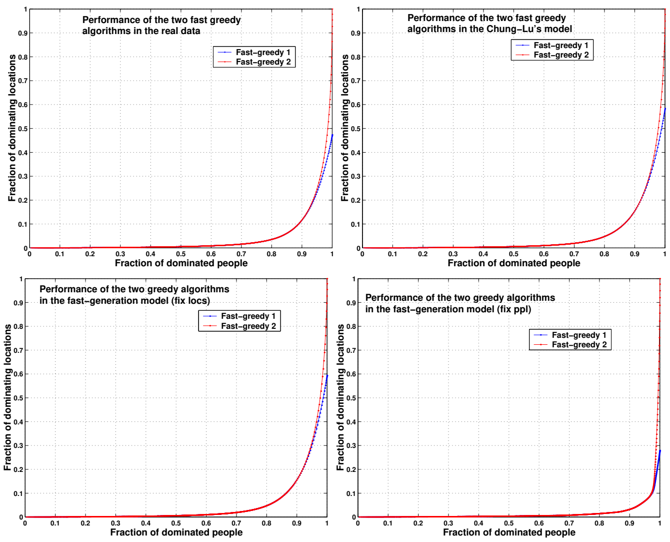

Figure 4: Performances of the two fast greedy algorithms in the Real-Network and the three models.

in **Fast-algorithm-1**. Instead of computing the lower bound of the clustering coefficient for p, we approximate it by $\frac{\sum_{i=1}^{d}|N(l_i)-1|(|N(l_i)|-2)}{(|E(N(p))|-|N(p)|)(|E(N(p))|-|N(p)|-1)}$. Without computing $\hat{N}(l_i)$ and $N(N(p))$, this algorithm is much faster than the first one. Both the algorithms yield a very good approximation to the distribution of the clustering coefficients in the people-people graph.

We also estimate the average clustering coefficient (c.c.) of the giant component of the people-people graph by sampling about a hundred vertices in each experiment. We did seven experiments respectively for the Real-network and the graphs generated by the three models. We calculated the mean and the standard deviation of the seven experiments. These values are presented in Table 3 along with the lower bound (l.b.) and the approximated lower bound (a.l.b.) of the clustering coefficients according to the above algorithms. Note that l.b. and a.l.b. are computed for all vertices of the graph, while c.c. is computed only for the sampled vertices, hence it is possible that l.b. and a.l.b. are higher than the *sampled* c.c. value.

(d) Dominating Sets. We considered four versions of greedy algorithms here: **RegularGreedy**, **VRegularGreedy**, **FastGreedy-1** and **FastGreedy-2**. **RegularGreedy** is the standard greedy algorithm that chooses a location that covers the maximum number of currently uncovered people. **VRegularGreedy** algorithm, works as follows: first find the set of all people of degree exactly *one*, call it P_1, then find the set of locations adjacent to P_1, i.e. $N(P_1)$. Obviously, any dominating set must contain $N(P_1)$. Then we find the set of people adjacent to $N(P_1)$ (including P_1), i.e. $N(N(P_1))$. Obviously $N(P_1)$ dominates $N(N(P_1))$. At last, we run the **RegularGreedy** algorithm on the subgraph $(P \backslash N(N(P_1)), L \backslash N(P_1))$. **FastGreedy-1** is similar to the FastGreedy algorithm but we do not take those l_i for which $N(l_i) \subseteq \bigcup_{j<i} N(l_j)$. **FastGreedy-2** is a variant of the FastGreedy algorithm where locations are grouped by degrees, and the groups of the largest degrees that cover a $(1-\epsilon)$-fraction of people are chosen. Obviously, **FastGreedy-2** is less accurate than **FastGreedy-1**, but it is easier to implement.

We study the performance of these four algorithms for the EpiSims graph and the graphs generated by the three models. Table 4 compares the **FastGreedy-1** heuristics with the **RegularGreedy** and the **VRegularGreedy** algorithms that dominate all people. The percentages in the table indicate the size of the dominating set compared to all locations. The time in Table 4 denotes the approximate experimental running time. Figure 4 compares the performances of **FastGreedy-1** and **FastGreedy-2**. As expected, **FastGreedy-1** runs much better on graphs generated by **FastGen-2** than on graphs generated by the other two models due to the higher values of overlap ratios. Meanwhile, the similarity of performance of **FastGreedy-1** on graphs generated by the CL-model and by the **FastGen-1** model is determined by the similarity of these two models. One can see that in Figure 4, in order to dominate a large fraction (say $\sim 95\%$) but not all of the people, the sizes of dominating sets by **FastGreedy-1** and by **FastGreedy-2** are almost the same.

Although **FastGreedy-1** is more efficient and accurate than **FastGreedy-2** (see Table 4), **FastGreedy-2** is potentially useful from the standpoint of implementation in the real world. To see this, note that in order to find a dominating set covering $(1-\epsilon)$-fraction of people, one does not need to run the code of **FastGreedy-2**, and even does not need to generate the edges of the bipartite graph. Given the description of the model (i.e. CL-model, **FastGen-1** model,

	Real-Network	CL	FastGen-1	FastGen-2
c.c. mean	0.6376	0.6161	0.6235	0.7021
c.c. std.	0.0167	0.0315	0.0236	0.0201
c.c l.b.	0.6286	0.6220	0.6241	0.7061
c.c. a.l.b.	0.6262	0.6220	0.6240	0.7061

Table 3: Means and standard deviations of clustering coefficients of sampled vertices in the giant components of people-people graphs. Lower bounds and approximated lower bounds to the clustering coefficents of giant components of people-people graphs.

	Real-Network	CL	FastGen-1	FastGen-2
FG1	47.35%	58.37%	59.60%	27.90%
RG	41.23%	54.47%	55.78%	27.54%
VRG	40.62%	54%	55.30%	27.54%
TFG1	$<$ 15 sec.	$<$ 15 sec.	$<$ 15 sec.	$<$ 15 sec.
TRG	$>$ 4 hr.	$>$ 5 hr.	$>$ 7 hr.	$>$ 2 hr.
TVRG	$\sim$ 2.5 hr.	$\sim$ 0.5 hr.	$\sim$ 0.5 hr.	$\sim$ 0.5 hr.

Table 4: Performance of three dominating-set algorithms for the Real-Network and graphs generated by the three models. Percentages denote the sizes of the dominating sets (compared to the whole set of locations) found by different algorithms. Seconds and hours denote the time needed to run these algorithms. FG1 = FastGreedy-1, RG = RegularGreedy, VRG = VRegularGreedy, TFG1 = Time of FastGreedy-1, TRG = Time of RegularGreedy, TVRG = Time of VRegularGreedy, sec. = seconds, hr. = hours.

or **FastGen-2** model) and the (input) degree sequence, one can figure out (with high confidence) the degree (say d_s) where **FastGreedy-2** should stop. Since actual degrees of locations are concentrated sharply around their input-degrees, one can take as the dominating set, those locations with input-degrees at least d_s. Additionally, Table 4 shows that although **VRegularGreedy** is much faster than **RegularGreedy** and provides the best solution among all the four algorithms, it is still much slower than the FastGreedy algorithms. Also, as we have seen, the FastGreedy algorithms have their own merits in the real world.

Acknowledgments. We thank the SODA 2004 reviewers for their helpful comments.

References

[1] R. Albert and A. Barabási. *Statistical Mechanics of Complex Networks.* Rev. Mod. Phys., 74 (2002), 47-97.

[2] R. Albert, A. Barabasi, and H. Jeong. *Diameter of the World Wide Web.* Nature, 401 (1999), 103-131.

[3] N. Alon and J. Spencer. *The Probabilistic Method, Second Edition.* Wiley, 2000.

[4] C. Barrett. *Simulation Science: The key to understanding National Infrastructures.* Los Alamos Research Quarterly, Spring 2003, available at `http://www.ccs.lanl.gov/ccs5/hottop.shtml`.

[5] A. Barabasi and R. Albert. *Emergence of scaling in random networks.* Science, 286 (1999), 509-512.

[6] B. Bollobas and O. Riordan. *Mathematical results on scale free graphs*, in *Handbook of graphs and networks*, S. Bornholdt and H. Schester (ed). Wiley-VCH, Berlin, 2002.

[7] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. *Graph Structure in the Web.* Proc. 9th World Wide Web Conf (2000), 309-320.

[8] F. Chung-Graham and L. Lu. *Connected components in random graphs with given degree sequences.* Annals of Combinatorics, 6 (2002), 125-145.

[9] C. Cooper and A. Frieze. *A general model of Web Graphs.* Random Structures and Algorithms, 22 (2003), 311-335.

[10] H. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. *The Origin of Power-Laws in Internet Topologies Revisited.* INFOCOM, 2002.

[11] S. Eubank and J. Smith. *Scalable, Efficient Epidemiological Simulation.* Proc. Symposium on Applied Computing, Madrid, 2002.

[12] S. Eubank, et al. *EpiSims Assessment of Responses to Smallpox Attack.* Report to the Office of Homeland Security, June 2002, Los Alamos Technical Report, LA-CP-02-254, 2002.

[13] S. Eubank, H. Guclu, V.S. Anil Kumar, A. Srinivasan, Z. Toroczkai, and N. Wang. *Controlling Smallpox Epidemics: Strategies drawn from an Empirical Data Instantiated Virtual City.* Technical Report, Los Alamos National Laboratory, 2003.

[14] E. Kaplan, D. Craft, and L. Wein. *Emergency response to a smallpox attack: the case for mass vaccination.* Proc. Natl. Acad. Sci. USA, 99 (2002), 10935-10940.

[15] C. Barrett et al. *TRANSIMS (TRansportation ANalysis SIMulation System): Volume 0-1: Overview, LA-UR-99-1658, Volume 2-6: LA-UR-99-2574-2580.* Los Alamos National Laboratory, 1999.

[16] C. Barrett, R. Jacob, and M. Marathe. *Formal Language Constrained Path Problems.* SIAM J. Computing, 30 (3), 809-837, June 2001.

[17] A. Frieze and C. McDiarmid. *Algorithmic theory of random graphs.* Random Structures and Algorithms, 10 (1997), 5-42.

[18] M. Faloutsos, P. Faloutsos, and C. Faloutsos. *On the power law relationships of the Internet topology.* Comput. Comm. Rev., 29 (1999), p. 251.

[19] U. Feige. *A Threshold of ln n for Approximating Set Cover.* JACM, 45(4), 634-652, July 1998.

[20] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. *The web as a graph.* Proc 19th ACM Symp. Principles of Database Systems, (2000), 1-10.

[21] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. *Stochastic models for the web graph.* Proc 41st IEEE Symp. Foundations of Computer Science, (2000), 57-65.

[22] M. Molloy and B. Reed. *A critical point for random graphs with a given degree sequence.* Random Structures and Algorithms, 6 (1995), 161-179.

[23] S. V. Pemmaraju. *Equitable colorings extend Chernoff-Hoeffding bounds.* SODA, (2001), 924-925.

[24] A. Srinivasan. *Distributions on Level-Sets with Applications to Approximation Algorithms.* Proc. IEEE Symposium on Foundations of Computer Science (FOCS 2001), 588-597.

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