Simulating Individual-Based Models of Epidemics in Hierarchical Networks

Rick Quax, David A. Bader*, and Peter M.A. Sloot

University of Amsterdam, Faculty of Sciences,
Science Park 107 1098 XG Amsterdam, The Netherlands

Abstract. Current mathematical modeling methods for the spreading of infectious diseases are too simplified and do not scale well. We present the Simulator of Epidemic Evolution in Complex Networks (SEECN), an efficient simulator of detailed individual-based models by parameterizing separate dynamics operators, which are iteratively applied to the contact network. We reduce the network generator's computational complexity, improve cache efficiency and parallelize the simulator. To evaluate its running time we experiment with an HIV epidemic model that incorporates up to one million homosexual men in a scale-free network, including hierarchical community structure, social dynamics and multi-stage intranode progression. We find that the running times are feasible, on the order of minutes, and argue that SEECN can be used to study realistic epidemics and its properties experimentally, in contrast to defining and solving ever more complicated mathematical models as is the current practice.

1 Introduction

Faithful simulations of epidemics in population networks require explicit modeling of a large number of static and dynamic properties of the network and the epidemic. However, current research often performs rigorous mathematical studies of non-representative simplifications. Examples of such network properties are degree distribution, community structure, assortativity, node and edge types, edge weights and temporal variance; properties of epidemics include infection stage progression, infectiousness, drug efficacy and immunity. In particular, epidemics are typically simulated using standard mean-field approximations or master equations extended with one or a few of these properties, but extending these to more representative models is difficult.

It is widely accepted that current mean-field approximations and master equations of epidemics are unrealistic and that more social and epidemiological details should be considered [1,2,3,4], so various biological and social systems are thought of as networks with various complex properties [5,6,7,8,9]. At present, however, epidemic studies using such complex networks focus on only one or a few such properties, such as degree correlation [10,11,12,13,14,15,16], topology [17,16,18,19,20,21],

* College of Computing, Georgia Institute of Technology, USA.

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 725–734, 2009.

edge weights [22,21], temporal variance [23], and epidemic dynamics [24,25]. At the other extreme and similar in spirit to our work is EpiSims [26], which uses detailed infrastructure and traffic data of a single city and simulates on the scale of seconds; however it does not extend easily to other and larger populations.

In this paper we present the Simulator of Epidemic Evolution in Complex Networks, or SEECN[1], with which detailed individual-based models can be simulated experimentally, based on available data. In SEECN, nodes and edges are organized hierarchically and have arbitrary properties that dictate the temporal evolution of the network and the epidemic. This evolution is driven by a set of *dynamics operators* that can be parameterized independently and in terms of the node and edge properties; for example, sexual relationships (edges) form and break depending on both nodes' gender and age. Their hierarchical organization reduces the model complexity and reflects how network data is typically available; for example, commuter traffic statistics may be available per province within a country, and per city within a province.

A consequential concern with such an expressive approach is running time, which is a challenge to optimize since scale-free networks often exhibit low degrees of locality and memory access patterns are not known a priori. We show how SEECN reduces computational complexity by exploiting hierarchical structure, improves cache efficiency by choosing an appropriate data structure and buffering edge traversals, and achieves parallelization with load balance. Our experiments show that these improvements reduce the running time to the extent that the simulator is practical even for detailed and highly entropic models.

This paper is organized as follows. In Section 2, SEECN is defined at a high-level with a minimum of implementation details, implying its expressiveness. Then, in Section 3, we discuss its algorithms and improvements. The experiments are presented in Section 4, and their results are shown and discussed in Section 5. Finally, we conclude in Section 6.

2 The SEECN

A simulation is boot-strapped by generating a representative network $G = (V, E)$, after which temporal *dynamics* are iteratively performed that drive the epidemic or change the network. See Fig. 1. We define dynamics as processes that change the network state in some way. All dynamics (including the network generator) are implemented as separate *dynamics operators* $\mathcal{D}_i$, or operators, which can be parameterized independently or reuse existing parameters. A model is a set of parameterized operators $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_d\}$.

To differentiate dynamics in a heterogeneous population, we augment each node $x \in [0, \|V\|)$ and edge $(x, y) \in E$ with a property vector $v_x \in \mathcal{V}$ or $w_{(x,y)} \in \mathcal{W}$, respectively, and parameterize operators in terms of these vectors. Each value $v_x[i]$ (or $w_{(x,y)}[j]$) of such a vector is one of a predefined set $\mathcal{V}[i]$ ($\mathcal{W}[j]$) of possible values for a specific node property i, such as gender or infection stage, or edge property j, such as type of relationship. Then, an operator implements a

[1] Pronounce as "season".

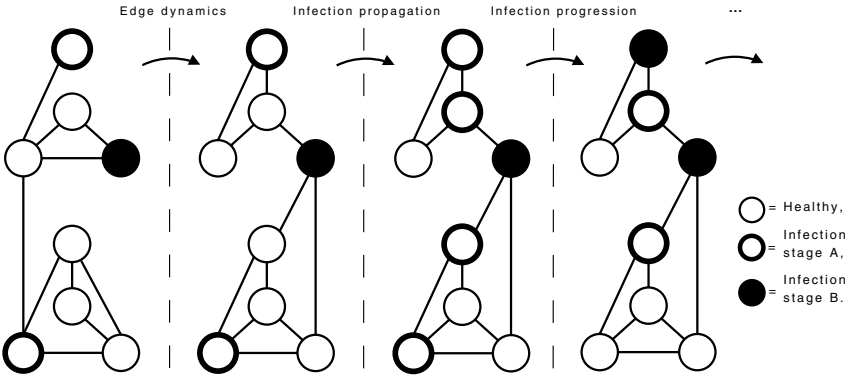

Fig. 1. Dynamics operators are applied sequentially to a network in one time step. The network has two communities, each with more internal than external edges.

transition probability matrix of $\|\mathcal{V}\| \times \|\mathcal{V}\|$ or $\|\mathcal{V}\|^2 \cdot \|\mathcal{W}\| \times \|\mathcal{V}\|^2 \cdot \|\mathcal{W}\|$ elements, essentially defining first-order stochastic differential equations for all network state[2]. Adding and removing nodes[3] and edges are special cases.

We use a separate operator for each type of dynamics, simplifying model specification. In this way, semantically different dynamics can be independently parameterized and turned on or off. In practice, each operator usually 'ignores' most of the parameter space; for instance, infection propagation changes only the receiving node's state, not that of the edge or the originating node.

In particular, the network generator $\mathcal{G}(h, p, q, N)$ is a special operator that is executed first and once. The first parameter is a network *recipe* $h : [0, N)^2 \mapsto [0, 1]$, which specifies the probability of each possible edge to be generated. Second, $p : \mathcal{V} \mapsto [0, 1]$ and $q : \mathcal{W} \mapsto [0, 1]$ specify the prior probability of any node or edge having a particular state. Lastly, N is the number of nodes.

The recipe is constrained as a modular network with hierarchical organization, or *hierarchy*, where the network is iteratively partitioned into subhierarchies. All nodes within a particular subhierarchy have equal connection probabilities to nodes of all other subhierarchies not contained by or containing the former. In particular, we define a *community* such that node x and y belong to the same community iff $\forall_z \; P((x, z) \in E) = P((y, z) \in E)$. In the worst case, the communities are precisely the lowest subhierarchies (which are not partitioned further), however in some cases multiple subhierarchies may be combined. Ultimately, a typical network's recipe is simplified with additional a priori statistical structure that we can exploit in our algorithms and analyses. Note that a single partition into N subhierarchies yields a recipe with no additional structure.

In addition to the benefit of a simplified recipe, we use hierarchical organization because many population networks appear hierarchical, and data is

[2] A *state* of a node or edge is a specific instance of its property vector; the network state is the combined state of its nodes and edges.

[3] Currently we only *replace* nodes, keeping network size constant.

often available in this way. For example, one may use travel statistics between provinces in a country, and between cities in a province. In this way, in the absence of individual data, each contact's probability can be estimated by traversing the hierarchy and multiplying the appropriate connection probabilities.

Finally, if network G_0 is an instance of a recipe, it changes over time step $i \geq 0$ as G_i when iteratively applying d dynamics operators, as

$$G_{i+1} = \mathcal{D}_1 \left(\mathcal{D}_2 \left(\ldots \mathcal{D}_d \left(G_i \right) \ldots \right) \right) ,$$
$$G_0 = \mathcal{G} \left(h, p, q, N \right) .$$

(Here, $\mathcal{D}(G_i)$ is shorthand for applying the operator to all existing nodes and edges at time step i.) Any statistic can then be calculated from the succession $(G_i)_i$, e.g., infection incidence, prevalence or treatment uptake.

SEECN's current implementation has some limitations. Most prominently, the hierarchy is constrained to that of "Kronecker graphs" [27] because of its good qualitative correspondence to real networks in absence of more detailed parameters that enable defining a recipe. This results in a recursive partitioning into two subhierarchies with equal connection probabilities at all levels. Further, edge probabilities are independent of property vectors; although this shortcoming can be significant, it can be partly overcome by changing the parameters for disease propagation.[4] Lastly, nodes are only removed or added, but do not migrate between communities.

3 Algorithmic Improvements

A primary concern with detailed individual-based simulations is running time. Many real networks are approximately scale-free [28], but such networks are notoriously difficult to partition and access patterns are highly irregular. Moreover, multiple simulation runs must be combined to obtain statistical significance, and typical applications include parameter-searching and impact evaluation of parameters. Therefore, a simulator should run fast in order to be useful.

For ease of presentation, we characterize operators as one of the following *primitives* that compute in $\mathcal{O}(N)$, $\mathcal{O}(|E|)$, and $\mathcal{O}(N^2)$ instructions[5], respectively.

Node operators visit all nodes, e.g., progression and removal.
Edge operators visit all existing edges, e.g, infection propagation.
Recipe operators visit all possible edges, e.g., network generation.

Our algorithmic improvements focus on computational complexity, cache efficiency and parallelization. Firstly, recipe operators dominate computation time and become a bottleneck for larger graphs even though such operators are cache efficient. The second bottleneck is due to cache misses, because the simulator performs many random memory accesses per little computation. Finally, an obvious but non-trivial improvement is parallelization.

[4] For instance, if unsafe sex with an HIV-infected person is less frequent, its infection probabilities are reduced.

[5] We distinguish computational complexity, in terms of instruction count, from running time complexity, which includes communication and memory access overhead.

3.1 Improving Computational Complexity

For recipe operators, we can exploit regularity imposed by hierarchical recipes. In contrast, the complexities of node and edge operators cannot be improved, assuming that every existing node and edge is relevant for the evolution of the epidemic. The following observation holds even if edge probabilities depend on property vectors since there is a constant number of possible states.

Many ($\omega(1)$) entries of a recipe are equal if $C \in o(N)$, where C is the number of communities. Since edge probabilities are equal for all nodes within a community, recipe operators compute in terms of communities instead of nodes. For example, to generate edges for a node its degree is drawn from a suitable distribution, since each community's contribution to a node's degree is binomially distributed. For larger communities these approach Poisson distributions that sum up. Then, each edge's neighbor node id is selected by first selecting a community id $[0, C)$ and then a random node id within that community.

Thus, recipe operators require $\mathcal{O}(NC)$ instructions. In worst-case, though, $C \propto N$ in which case it remains $\mathcal{O}(N^2)$; for Kronecker graphs, $C = \log_2 N$ [29].

3.2 Improving Cache Efficiency

Random memory access dominates running time due to visiting and traversing edges, both of which require special attention [30] and are discussed in turn. Node and recipe operators can visit node structures and random number generators in order, so this section focuses on edge operators.

Network dynamism further constrains the choice of data structure and suggests the classic adjacency matrix (see left of Fig. 2). To remain efficient, all memory should be pre-allocated and not be (de)allocated constantly, which would fragment memory. The adjacency matrix stores edges contiguously per node and reserves equal capacity $0 < D \leq N$ for each node. Visiting all edge sides of one node is in cache. However, in scale-free networks only a small fraction of nodes uses its full capacity, in which case visiting the first edge side of the next node is (almost) always a cache miss. Thus, the adjacency matrix incurs an expected $\langle M \rangle = N$ cache misses per node.

An alternative is the Jagged Diagonal Storage (JDS) [31] which stores node x's ith edge in the ith of D arrays, at element x of N (see right of Fig. 2). In JDS, all edge sides are visited per such array, and a cache miss occurs only when some node's ith edge is accessed while $B \geq 1$ previous nodes had less than i edges. To minimize this probability we renumber nodes on expected degree.

We can show that, for *visiting* all edge sides, JDS incurs fewer expected cache misses $\langle M \rangle$ than the adjacency array, as follows. We assume that a cache miss occurs if two memory accesses are separated by B or more edge structures. Node x has degree d with probability $f_x(d)$, and $F_x(d) = \sum_{i=0}^{d} f_x(i)$. The probability $P(M_x \geq m)$ of node x incurring at least $0 < m \leq D$ cache misses is then

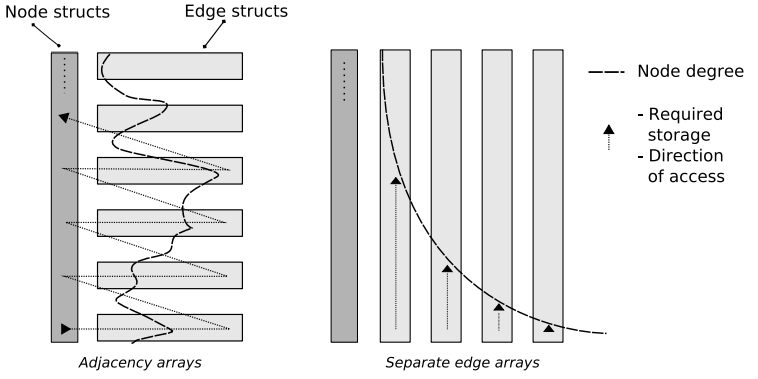

Fig. 2. A comparison of memory access patterns between the adjacency array and JDS for a scale-free network

$$
\begin{aligned}
P(M_x \geq m) &= \sum_{d=1}^{D} f_x(d) \cdot \prod_{y=1}^{L} F_{x-y}(d-m) \\
&\leq \sum_{d=1}^{D} f_x(d) \cdot F_{x-1}(d-m) && (L=1) \\
&\leq \sum_{d=1}^{D} f_x(d) \cdot F_x(d-m) \ < \frac{1}{2} && (\forall_{d'}\ F_x(d') \geq F_{x-1}(d'))
\end{aligned}
$$

(For binomial f_x.) In other words, most nodes will incur no cache miss at all. Although this does not prove that $\langle M \rangle < 1$ for all possible sets of parameters, we have been unable to find such sets experimentally. Computed using the above, $\langle M \rangle \approx 0.6$ for the simulations in the next section.

The second source of cache misses is due to *traversing* edges, which we reduce by queuing and sorting *updates*. Operators in SEECN queue updates, which dictate state changes for a neighboring node. Eventually the updates are sorted on node identifier and performed in order, splitting the traversal problem into a sorting part and an edge visiting part. For improved efficiency, the updates of multiple independent operators may be combined.

3.3 Parallelization

Because edge traversals dominate running time for large graphs, and usually $|E| \in \Omega(N)$, we partition the network with respect to edges. In contrast, node operators are relatively inexpensive because they access all nodes in order. Moreover, node structures must already be accessed for edge operators, e.g., to update a node's degree after edge removal or addition.

Therefore we partition the network at the granularity of nodes, in contiguous ranges, while balancing the expected total number of edges to be handled by each

process. Although SEECN could exploit the hierarchical community structure to partition the network more optimally, the current partitioning algorithm is simplified by assuming no assortativity beyond that in expected degree; in fact, Kronecker is such a generator [29]. This assumption implies that a node x with expected degree $\langle d_x \rangle$ expectedly incurs $\langle d_x \rangle$ / $\langle d_y \rangle$ more interprocess edges than node y, and therefore the optimal partition is to balance the expected number of edges to handle by any process i out of p. The probability that an edge connects two nodes of different processes is $(p-1)/p$.

In terms of implementation, few changes need be made. Most prominently, each process' update queue (Section 3.2) must be split into p queues, one for each process; at intermediate steps the queues can easily be transferred at once, benefiting from high communication bandwidth. Exploiting high bandwidth is important because the size of a queues scales as $\mathcal{O}(|E|)$ for increasing N.

4 Experiments

In this section we perform benchmark experiments using one, four and sixteen processes. The epidemic model is fairly complex and represents HIV in a population of homosexual men. This model assumes a hierarchical network with power-law exponent 1.6, and classifies nodes as healthy, acute, (un)treated asymptomatic, or (un)treated AIDS, each of which have distinct infectiousness, life expectency and duration of relationships. The details are presented elsewhere [29], where good qualitative correspondence with historical data was found.

We experiment with three variants of SEECN to evaluate the impact of cache efficiency and load balance. The first algorithm (`cached` or `C`) queues updates but does not renumber the nodes on descending degree (Section 3.2) and does not sort the queues. As variations to `cached`, the second algorithm (`load-balanced` or `LB`) renumbers the nodes and the third (`sorted` or `S`) also sorts the queues.[6] The latter two algorithms partition the graph into equal expected edge counts per process, whereas the `cached` algorithm must resort to balancing the number of nodes due to the irregular distribution of edges over nodes.

The experiments are run on a Linux cluster of 680 dual-core Intel Xeon nodes at 3.4 GHz. We allocate one process per compute node (homogenizing communication overhead) and use TCP/IP over Infiniband. We implement SEECN in C++, parallelize it with MPI, and compile it with GCC 4.1.2 and MPICH2 1.0.8. SEECN's code is not hand tuned and uses the standard STL sorting algorithm.

5 Results

The sequential experiments show the impact of cache efficiency of both visiting (`load-balanced`) and traversing (`sorted`) edges, for increasing N. The results are shown in Fig. 3. We could not experiment with network sizes of over 2^{20} because a single process would run out of memory.

[6] Particularly, we do not analyze the impact of using the JDS instead of the adjacency array, because only JDS is implemented.

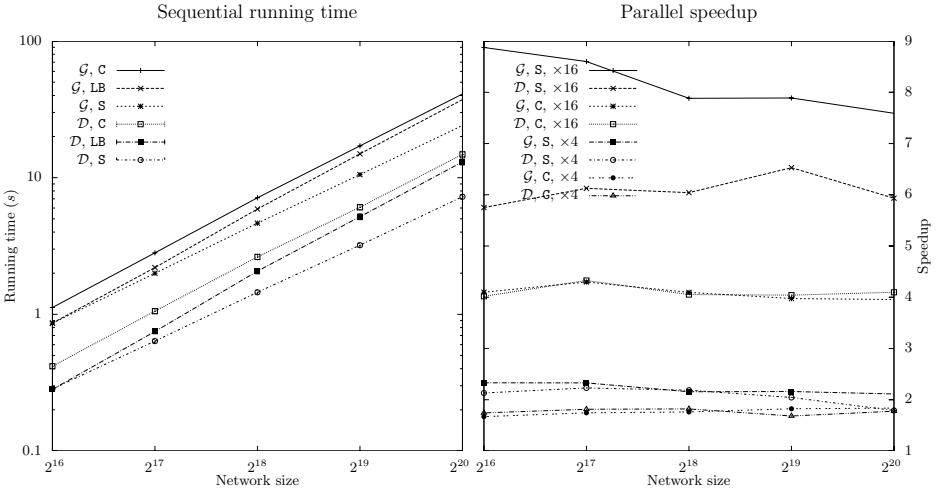

Fig. 3. The sequential running times (on logarithmic scale) of generating the network ($\mathcal{G}$) and performing all other dynamics in a single time step ($\mathcal{D}$), and the parallel speedups of both using 4 and 16 processes. All data points are averages of five runs, and each run had 100 time steps of dynamics (standard error is too small to show).

Firstly, the results suggest indeed that the running time scales as $|E|$. The lorarithmic running time plots of the `sorted` generator and dynamics operators have slopes of 2.26 and 2.30. The theoretically expected slope for the logarithmic edge count is 2.3 (not shown) because $|E| \propto 2.3^{\log_2 N}$ [29], in the absence of a maximum degree. It is also interesting that the fraction of edge visits and traversals that are cache efficient without sorting queues decreases with increasing network size, because both slopes for the `LB` algorithm are about 2.45.

Secondly, the performance benefit of renumbering nodes and visiting them in order is significant for smaller graphs ($N \leq 2^{16}$), for which sorting queues has no benefit. However, the number of updates per time step grows as $|E|$, so the average number of updates per node increases, whereas the number of initial visits (i.e., not caused by an update) remains constant. Consequently, for larger graphs ($2^{16} < N \leq 2^{20}$) the benefit of cache efficient initial visits diminish while the benefit of sorting updates and performing them in order increases.

For the parallel case we show the speedup of the `cached` and `sorted` algorithms over the same range as for the sequential case in Fig. 3. Clearly, balancing the number of nodes is less efficient than balancing edge counts. For an increasing number of processes, `sorted`'s efficiency curves are about 0.5 and almost constant; in contrast, `cached`'s efficiency curves drop from 0.4 to 0.25.

The speedups remain roughly constant over our range of N, which would be the case if both processing and communication time would be dominated by the number of updates and therefore by $|E|$. Communication overhead that is sublinear in $|E|$ in the absence of additional assortativity is not possible (without load imbalance), so we consider our parallelization efforts successful.

6 Conclusion

In this paper we show that detailed, individual-based models of epidemics can indeed be simulated in reasonable running time; in particular, we discuss and implement algorithmic improvements and evaluate the impact of buffering edge traversals, sorting these traversals, and parallelizing with balanced numbers of nodes or edges. For a representative epidemic model we find a constant sequential running time reduction of almost factor 2, and a constant parallel eficiency of roughly 0.5. As a result, a detailed simulation of HIV among one million persons in a hierarchical and scale-free network over 25 years (100 time steps) takes two minutes using 16 processes. We conclude that for simulating epidemics, experimentation can be an expressive and convenient alternative to mathematical modeling, and that parameter searching and impact evaluation are feasible.

Acknowledgments. We thank Breanndán Ó Nualláin and Dung Manh Chu for their help with the cache efficiency derivations, and Richard Vuduc for pointing out the existence of the Jagged Diagonal Storage. Bader's research is supported in part by NSF Grant CNS-0614915, Pacific Northwest National Laboratory's Center for Adaptive Supercomputing Software and by MIT Lincoln Laboratory. This research is supported by European ViroLab Grant INFSO-IST-027446.

References

1. Keeling, M.: The implications of network structure for epidemic dynamics. Theoretical Population Biology 67(1), 1–8 (2005)
2. Pastor-Satorras, R., Vespignani, A.: Epidemic dynamics in finite size scale-free networks. Phys. Rev. E 65(3), 035108 (2002)
3. Keeling, M., Eames, K.: Networks and epidemic models. Journal of the Royal Society Interface 2 (November 2005)
4. Parham, P.E., Ferguson, N.M.: Space and contact networks: capturing the locality of disease transmission. Journal of The Royal Society Interface 3, 483–493 (2006)
5. Strogatz, S.H.: Exploring complex networks. Nature 410(6825), 268–276 (2001)
6. Wuchty, S., Ravasz, E.: lászló Barabási, A.: The architecture of biological networks. In: Complex Systems in Biomedicine. Kluwer Academic Publishers, Dordrecht (2003)
7. Barabási, A.L., Oltvai, Z.N.: Network biology: understanding the cell's functional organization. Nature Reviews Genetics 5(2), 101–113 (2004)
8. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: Structure and dynamics. Physics Reports 424(4-5), 175–308 (2006)
9. da Costa, F.L., Rodrigues, F.A., Travieso, G., Boas, P.R.V.: Characterization of complex networks: A survey of measurements. Advance. Advances In Physics 56, 167 (2007)
10. Sloot, P.M.A., Ivanov, S.V., Boukhanovsky, A.V., van de Vijver, D.A.M.C., Boucher, C.A.B.: Stochastic simulation of HIV population dynamics through complex network modelling. Int. J. Comput. Math. 85(8), 1175–1187 (2008)
11. Pastor-Satorras, R., Vespignani, A.: Epidemics and immunization in scale-free networks. In: Bornholdt, S., Schuster, H.G. (eds.) Contribution to Handbook of Graphs and Networks: From the Genome to the Internet. Wiley-VCH, Berlin (2002)

12. Pastor-Satorras, R., Vespignani, A.: Epidemic dynamics and endemic states in complex networks. Physical Review E 63, 66117 (2001)
13. Moreno, Y., Pastor-Satorras, R., Vespignani, A.: Epidemic outbreaks in complex heterogeneous networks. The European Physical Journal B - Condensed Matter and Complex Systems 26(4), 521–529 (2002)
14. Zhou, C., Kurths, J.: Hierarchical synchronization in complex networks with heterogeneous degrees. Chaos: An Interdisciplinary Journal of Nonlinear Science 16(1), 015104 (2006)
15. Barthelemy, M., Barrat, A., Pastor-Satorras, R., Vespignani, A.: Dynamical patterns of epidemic outbreaks in complex heterogeneous networks. Journal of Theoretical Biology 235, 275 (2005)
16. Sorrentino, F.: Effects of the network structural properties on its controllability. Chaos 17(3), 033101 (2007)
17. Petermann, T., Rios, P.D.L.: The role of clustering and gridlike ordering in epidemic spreading. Physical Review E 69, 066116 (2004)
18. Grabowski, A., Kosiński, R.A.: Epidemic spreading in a hierarchical social network. Physical Review E 70(3), 031908 (2004)
19. Grabowski, A., Kosiński, R.A.: The SIS Model of Epidemic Spreading in a Hierarchical Social Network. Acta Physica Polonica B 36, 1579 (2005)
20. Verdasca, J., da Gama, M.M.T., Nunes, A., Bernardino, N.R., Pacheco, J.M., Gomes, M.C.: Recurrent epidemics in small world networks. Journal of Theoretical Biology 233(4), 553–561 (2004)
21. Barthélemy, M., Barrat, A., Vespignani, A.: The role of geography and traffic in the structure of complex networks. Advances in Complex Systems (ACS) 10(01), 5–28 (2007)
22. Barrat, A., Barthelemy, M., Pastor-Satorras, R., Vespignani, A.: The architecture of complex weighted networks. Proc. Natl. Acad. Sci. U.S.A. 101(11), 3747–3752 (2003)
23. Restrepo, J.G., Ott, E., Hunt, B.R.: Characterizing the dynamical importance of network nodes and links. Physical Review Letters 97(9), 094102 (2006)
24. Masuda, N., Konno, N.: Multi-state epidemic processes on complex networks. Journal of Theoretical Biology 243, 64 (2006)
25. Aalen, O., Farewell, V., De Angelis, D., Day, N., Gill, O.: New therapy explains the fall in AIDS incidence with a substantial rise in number of persons on treatment expected. AIDS 13(1), 103–108 (1999)
26. Eubank, S., Guclu, H., Anil, M.M.V., Srinivasan, A., Toroczkai, Z., Wang, N.: Modelling disease outbreaks in realistic urban social networks. Nature 429(6988), 180–184 (2004)
27. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C.: Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS, vol. 3721, pp. 133–145. Springer, Heidelberg (2005)
28. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509 (1999)
29. Quax, R.: Modeling and simulating the propagation of infectious diseases using complex networks. Master's thesis, Georgia Institute of Technology (2008)
30. Madduri, K.: A High-Performance Framework for Analyzing Massive Complex Networks. PhD thesis, Georgia Institute of Technology (2008)
31. Saad, Y.: Krylov subspace methods on supercomputers. SIAM J. Sci. Stat. Comput. 10, 1200–1232 (1989)