Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

INTERACTION-BASED HPC MODELING OF SOCIAL, BIOLOGICAL, AND ECONOMIC CONTAGIONS OVER LARGE NETWORKS

Keith Bisset
Jiangzhuo Chen
Chris J. Kuhlman
V. S. Anil Kumar
Madhav V. Marathe

Network Dynamics & Simulation Science Laboratory
Virginia Bioinformatics Institute, Virginia Tech
Blacksburg, Virginia 24061, USA

ABSTRACT

Modeling large-scale stochastic systems of heterogeneous individuals and their interactions, where multiple behaviors and contagions co-evolve with multiple interaction networks, requires high performance computing and agent-based simulations. We present graph dynamical systems as a formalism to reason about network dynamics and list phenomena from several application domains that have been modeled as graph dynamical systems to demonstrate its wide-ranging applicability. We describe and contrast three tools developed in our laboratory that use this formalism to model these systems. Beyond evaluating system dynamics, we are interested in understanding how to control contagion processes using resources both endogenous and exogenous to the system being investigated to support public policy decision-making. We address control methods, such as interventions, and provide illustrative simulation results.

1 INTRODUCTION

Contagion is used here broadly to mean transmitted phenomena such as diseases, opinions, fads, trends, norms, packet diffusion, worm propagation in computer networks, database replication in sensor networks, spread of social movements, and influence among peers to purchase music videos or go to movies (Granovetter 1978, Halloran et al. 2008, Easley and Kleinberg 2010, Channakeshava et al. 2011). The spread of contagions over large populations includes complex problems such as: (i) pandemics, such as H1N1 and swine influenza outbreaks in recent years, in which the spread of the flu virus is often modeled by stochastic processes, such as the SIR process (Eubank et al. 2004, Germann et al. 2006), and (ii) spread of information on online social media, such as Twitter and Facebook (Easley and Kleinberg 2010), which are often modeled by stochastic and threshold based models (Kempe et al. 2005). A key observation from numerous studies shows that the underlying network structure has a significant impact on the dynamics (e.g., Gould 1993, Siegel 2009), and as in the case of the 2003 blackout, this could span multiple networks (Newman 2003).

Though these examples involve very diverse phenomena on different kinds of networks, they often involve very similar and fundamental questions from a dynamical systems perspective. In the case of epidemics, the key challenges include understanding the disease dynamics (e.g., what is the likely attack rate, and who might be vulnerable), detecting its onset and peak, and developing and evaluating strategies (e.g., vaccinations, school closures) to control the spread of the disease—these have to be translated (often in real time) into policies to be implemented by local and national public health agencies (e.g., Germann et al. 2006, Halloran et al. 2008). In the case of the spread of information and viral marketing, it is often

of interest to identify the most influential spreaders to initiate the diffusion process, so that the size of the population that gets influenced (to change and adopt a new product, for instance) is maximized (Domingos and Richardson 2001, Kempe et al. 2003). In the case of coupled infrastructures, understanding the impact of cascading failures and identifying criticality of different network elements is a very important issue. Thus, the key questions underlying diverse contagion phenomena can be related to fundamental dynamical systems problems of control and optimization via exogenous and endogenous resource allocation.

Simple contagion processes, such as percolation, have been studied extensively and are well understood on simple regular graph families (Grimmett 1989) and complete mixing models, which allow interactions among the whole population (Newman 2003). Analytical closed form results for such models have been obtained for many important questions using elegant techniques that include stochastic processes, random graphs, and probabilistic methods and techniques originating in statistical physics. However, these models do not capture aspects of realistic social networks, and are inadequate for general public health analysis. In contrast to traditional mean field analysis, individual-based models make explicit the idea of "interaction" and macro-level emergent properties are caused by the local interactions of millions of individual agents (Mitchell 2009). However, these techniques do not easily extend to large-scale realistic networks, because of the absence of independence and symmetry and the inclusion of heterogeneity. Further, many of the underlying problems related to dynamical properties become computationally intractable (e.g., the probability of a specific individual becoming infected is #P-complete, in general) – thus, high performance computing (HPC) agent-based modeling and simulation (ABMS) techniques become a natural choice. Developing computational models to reason about these systems is complicated and scientifically challenging for at least the following reasons.

First, often the size and scale of these systems is extremely large (e.g., pandemic planning at a global scale requires models with 6 billion agents). Further, the networks are highly unstructured and the computations involve complicated dependencies, leading to high communication cost and making standard techniques of load balancing and synchronization ineffective. Second, individuals are not identical – this implies that models of individual behavioral representation cannot be identical. Behavior depends on individual demographic attributes and the interactions with neighbors (Granovetter 1978, Monge and Contractor 2003). Third, the contagion, the underlying interaction network (consisting of both human and technical elements), the public policies and the individual agent behaviors co-evolve, making it nearly impossible to apply standard model reduction techniques that are successfully used to study physical systems. For instance, in the case of epidemics, as disease spreads, people reduce their interactions, thereby sparsifying the network, which in turn slows the disease dynamics. Finally, in many cases, as we discuss below, we are faced with modeling multiple networks that are coupled, with possibly multiple contagions evolving in each network.

Additionally, in contrast to many large physics simulations, the outcome of a single run of a socio-technical simulation is not interesting by itself. For example, simulations of infectious disease outbreaks are run not for their own sake, but to investigate specific questions about prevention and mitigation. Answering these questions requires analyzing the interdependent effects of many different parameters, each with many different possible settings, in a stochastic process. The stochastic nature of the simulation, the uncertainty in the initial conditions, and the variability of reactions require many replications to develop a bound on the range of results of the simulation. Also, the simulations are not meant to be predictive, but to provide comparative results over a range of simulation inputs. For example, the outcome of an infectious disease propagation experiment is not that a particular number of people will fall ill, but that intervention A is generally more effective than intervention B over the range of expected disease manifestations and initial conditions (Germann et al. 2006). Efforts to devise effective policies from simulation experiments is also gaining traction in other domains, such as financial services; e.g., (Haldane and May 2011).

Typically, overall time-to-solution is one measure of simulation effectiveness: stakeholders need answers as soon as possible. Indeed, many policymakers are forced to use inaccurate, but fast executing, tools in place of accurate but slower tools. Efficient experimental design is a crucial part of reducing the

time to solution. A complete factorial design is often infeasible, and a partial factorial design applied to such a complex, nonlinear system can miss important interdependencies. Hence, adaptive designs are usually required. Even with an adaptive design, however, there may still be thousands of experimental cells to explore. The computational burden of a simulation should be considered in the context of such an experiment, and resources should be allocated optimally over hundreds or thousands of runs, each run requiring tens or hundreds of thousands of computational cores. To this end, pending work is to exploit the close coupling of a petascale machine, as opposed to a collection of terascale machines, to obtain efficiencies by combining the calculations of multiple replicates, and, potentially, multiple cells, thereby reducing the total amount of computation needed.

In this paper, we overview the mathematical concepts of graph dynamical systems (GDS) that we use to reason generally about dynamical processes. We demonstrate its versatility by itemizing practical problems that have been evaluated using GDS. We then present a suite of scalable simulation tools for these processes, including the susceptible-infectious-recovered (SIR) model of disease propagation, malware spread on communication networks, and complex social contagions. We contrast three tools with respect to their ranges of applicability and their performance in terms of execution speed. We show how tailoring a tool to a particular problem domain enables results to be generated quickly, which is crucial for short-term hard deadlines. We also present illustrative results and demonstrate how they are useful to application domain decision-makers.

Limitations and scope. While the GDS theory allows for very general dynamical processes, the following kinds of processes result in a high overhead in our current parallel framework: (i) reaction diffusion processes in which more than two agents come together (e.g., of the form "$A+B+C \rightarrow D+E+F$") – such processes arise in various applications, such as ecological modeling and chemical reaction networks (Colizza, Pastor-Satorras, and Vespignani 2007); (ii) context-dependent birth processes, in which an agent is transformed into multiple agents; and (iii) the underlying network varies dynamically at time-scales comparable to the dynamical process itself (e.g., in the case of disease transmission, edges change at rates comparable to the spread of infection). As will be discussed in Section 4, the parallel optimizations underlying two of our tools (EpiSimdemics and EpiFast) rely on semantics of the stochastic diffusion processes, and cannot be easily extended to incorporate such processes, without significant additional overheads. However, some of these can be handled in InterSim, since these can be expressed as a GDS.

2 GRAPH DYNAMICAL SYSTEMS

In this section, we present an abbreviated formal description of GDS and then itemize application domains that have been modeled as GDS in order to illustrate its utility. For a more in-depth discussion of GDS, see (Barrett et al. 2006, Mortveit and Reidys 2007).

A GDS is an abstract representation of a group of entities (agents), modeled as nodes, and abstract interactions, modeled as edges. This representation provides a sound basis to develop simulations of diffusion processes in such systems. The generality of GDS is reflected in the fact that it is a universal model of computation and can therefore simulate a Turing machine. Note that the theoretical universality of GDS does not imply that all *practical* problems can be solved expeditiously with a GDS. We present the basic elements of a GDS and then briefly discuss generalizations.

A **graph dynamical system** (GDS) $\mathscr{S}$, is a 4-tuple $\mathscr{S} = (G, \mathbb{B}, \mathscr{F}, \mathscr{R})$ consisting of a graph $G(V,E)$ whose node set V represents the collection of agents and whose edge set E represents the set of agent interactions. Let $n = |V|$ denote the number of nodes in G. Each node has a **state**, a value from a finite set $\mathbb{B}$ of all possible state values. (More complex state representations are described later.) Further, there is a family $\mathscr{F}$ of functions that describe state transitions. Specifically, each node $v_i \in V$, $1 \leq i \leq n$, has an associated **local transition function** $f_i \in \mathscr{F}$ which determines the next state of the node. In general, f_i may depend on several parameters including the history of the current and previous states of v_i and those of its neighbors in G. For example, a local transition function for the state of v_i at time $(t+1)$, $s_i(t+1)$, may depend on parameters over a time window of duration T, $s_i(t+1) = f_i(s_i(\theta), N[i](\theta), E[i](\theta)), \quad t-T+1 \leq \theta \leq t$,

where $N[i](\theta)$ represents the states of the neighbors of v_i at time θ and $E[i](\theta)$ represents the states of the edges incident on v_i at time θ. Further, each GDS has an associated **update scheme** $\mathscr{R}$ that determines the order in which the local transition functions are computed and states of nodes are updated. For example, a **synchronous** (i.e., parallel) update scheme is often utilized, where all f_i are executed in parallel, to make the best use of parallel processing. However, some applications demand other update schemes, such as **sequential** update, where a permutation of the nodes v_i is used to identify the single node whose f_i is executed at one time, in order to ensure particular algorithmic behavior, such as solution convergence (Karaoz et al. 2004). (GDS with synchronous update are often called **synchronous dynamical systems** (SyDS) and GDS with sequential update are often referred to as **sequential dynamical systems** (SDS).) At any time t, the **configuration** $\mathscr{C}(t)$ of a GDS is a vector $(s_1(t), s_2(t), \ldots, s_n(t))$, where $s_i(t)$ represents the state of node v_i at time t. The time evolution of a GDS is represented by the sequence of successive configurations of the GDS.

A concrete example of location transition functions are threshold functions (Granovetter 1978). Consider the two-state system $\mathbb{B} = \{0,1\}$ for nodes of a connected graph G, where the only permissible state transition is from state 0 to state 1. A node in state 1 remains in state 1. Given a threshold τ_i for each node, node v_i in state 0 will transition to state 1 if at least τ_i neighbors of v_i are in state 1. Otherwise it remains in state 0. Threshold models account for peer influence, which is known to be a significant factor in the spread of the three greatest public health concerns (Sturm 2002): obesity (Christakis and Fowler 2007), smoking (O'Loughlin et al. 2009), and excessive drinking (Borsari and Carey 2001). Many common "everyday" examples of threshold behavior are given in (Schelling 1978, Granovetter 1978). Moreover, (Watts 2002) argues that threshold behaviors are often used by people in place of more complicated reasoning mechanisms, and provides illustrative examples.

The basic GDS model described above can be generalized and extended in a number of ways. Among these are: (i) probabilistic state transitions; (ii) multiple contagions c_i, each with its own interaction graph G_i; (iii) evolving contagions, such as virus mutations or changes in information content such as that occurring in the childhood "telephone" game where an initial message given to the first child is substantially changed by the time it is passed to the last child; (iv) for each node, state can be represented by an arbitrary vector of different types of values (e.g., integers, doubles); (v) directed edges to model asymmetric interactions (e.g., information flow via broadcast media); (vi) time-varying existence of nodes and edges; (vii) multiple mechanisms of state transition to describe any number of mechanistically-driven state changes; $(viii)$ heterogeneity of node and edge properties and individualized evolution of properties and states, which may be history-dependent (i.e., non-Markovian); and (ix) interventions that alter the network or node or edge properties (i.e., labels) based on some criterion that is beyond the scope of local transition functions. Many of these features fall within the domain of GDS (Mortveit and Reidys 2007), but we will refer to these extensions as constituting a **generalized GDS**.

A concrete example incorporating many of the above ideas is the spread of a disease across a population. In addition to the disease contagion on a human contact network, multiple forms of information about its progression, such as television news, word-of-mouth, and online social networks—each a contagion on a possibly different network—could be propagated and affect people's actions, which in turn might affect who they contact. Finally, fear may be another contagion, to model an agent's propensity to withdraw from everyday activities, or overuse pharmaceutical interventions. Clearly, these contagions affect each other, and we have co-evolution of contagions and possibly networks.

Table 1 provides a set of models for reaction-diffusion processes in application domains ranging from epidemiology to social behavior to economics that have been modeled as generalized GDS. This list is not exhaustive; its purpose is to demonstrate the far-reaching applicability of GDS.

Beyond these models, other works describe influence networks of agents that strongly suggest GDS models could be constructed. An example is financial contagions and their effects on global markets (May et al. 2008). Furthermore, there are many models focused on determining equilibria of systems (e.g., Allen and Gale 2000). While many such models are formulated as optimization problems, it may be possible to

recast them as transient phenomena in terms of GDS; see (Epstein 2007) for an interesting discussion on modeling of dynamical and equilibrium phenomena. A short discussion of recent papers in agent-based computational economics (ACE) is provided in (Markose et al. 2007), which has a close association with game theory (e.g., Easley and Kleinberg 2010); these are possible domains for GDS modeling.

Table 1: Published diffusion models for social, epidemiological, economic, biological, and physics problems that can be modeled as generalized GDS.

Model	**Application Domains**
Collective action, ratcheted thresholds (Epstein 2002, Centola and Macy 2007, Dreyer and Roberts 2009, Siegel 2009).	Economics (e.g., purchasing high-cost items); crowd behavior (e.g., strikers, leaving a social occasion); civil disobedience; politics.
Revolutions (Kuran 1989, Gould 1993)	Revolutions come about owing to subtle, latent changes in a population; social movements.
Cascades (Goldenberg et al. 2001).	Social behavior (e.g., adoption of fads, peer pressure to join a club).
Threshold (Choi et al. 2010).	Diffusion of innovations.
Thresholds, cascades (Easley and Kleinberg 2010).	Social diffusion (e.g., fads, rumors).
Thresholds (Kempe et al. 2003, Kempe et al. 2005, Kuhlman et al. 2010).	Generic social diffusion; influence maximization.
Generalized contagion (Dodds and Watts 2005).	More sophisticated epidemiological and social behaviors, incorporating a quantitative measure of each "interaction" between two entities (i.e., a dosage).
Diffusion of warnings (Hui et al. 2010).	How alarming warnings are spread through a population.
Diffusion of information (Mertsalov et al. 2008).	Effect of (information) diffusion for dynamic networks.
Spread of information (Gruhl et al. 2004).	Spread of information via internet communication.
Diffusion of trust (Guha et al. 2004).	Trust propagation.
Spread of marketing information (Domingos and Richardson 2001, Easley and Kleinberg 2010).	How (product) marketing can be modeled and hence improved.
Spread of malware (Channakeshava et al. 2011).	Propagation of malware through a network of wireless devices.
Epidemiology (Halloran et al. 2008).	Virus diffusion in humans and other animal populations.
Blocking diffusion (Habiba et al. 2008, Kuhlman et al. 2010)	Blocking the spread of social contagions such as fads or marketing campaigns.
Statistical physics (Castellano et al. 2009)	Many models used in statistical physics, such as the Ising Model.
Language evolution (Cangelosi and Parisi 2001)	Spread and evolution of languages.
Biology (Karaoz et al. 2004)	Identifying biological functions of genes.
Economics and electric power (Atkins et al. 2009)	Market power of suppliers of electricity.

3 RELATED WORK

Modeling references are provided in Sections 1 and 2 and are not repeated. Here, we address two types of simulators: epidemiological-focused and more general simulators. We choose epidemiology as a special case because it illustrates how large scale HPC agent based simulation (ABS) methods are combined with detailed population representations and interventions, which is uncommon. Owing to space limitations, this section is not exhaustive.

There are many epidemiological ABS platforms. The system described by Ferguson et al. (2003) is implemented to be executed on a shared memory platform and is limited by the amount of available shared memory. The work of Longini et al. (2005) is a parallel simulation that uses a very simple and structured social contact network. The locations in these social contact networks are not real but simply surrogates

for simple location types such as school, home, etc. This results in a structured social contact network that is more amenable to efficient parallel computation, but which, arguably, is less representative of real-world social networks. The simulator described in (Parker 2007, Parker and Epstein 2011) is implemented in Java and involves numerous optimizations. It is a combination of spatial model (dividing the region into pixels) and agent model with randomly constructed contact networks. While it quite efficiently simulates very large populations, it currently does not support necessary interventions required in practical public health studies.

There has also been a wealth of recent work on general-purpose simulators that utilize a range of dynamics models. Among these are AnyLogic, Aurora2, BRACE, μsik, NetLogo, Repast SC++, SASSY, and Swarm (Perumalla 2005, Hybinette et al. 2006, Park and Fujimoto 2009, North and Macal 2009, Wang et al. 2010). These simulators span the range of smaller-scale ones with accompanying visual tools (e.g., NetLogo) to some of the largest simulations in terms of computing resources used (as many as 65000 processing cores (Carothers and Perumalla 2010, Perumalla and Seal 2011)). Graphics processing units (GPU) have also been used (D'Souza et al. 2007, Aaby et al. 2010, Hwu 2011), simulating hundreds of millions of agents. Two of the three codes described here, EpiSimdemics and EpiFast, also readily handle hundreds of millions of agents. We refer the interested reader to the following sources for more details and references (Perumalla 2005, Aaby et al. 2010, Kuhlman et al. 2011).

4 NDSSL TOOLS

4.1 Overview

In this section, we describe three simulation tools developed at NDSSL, starting with the most general tool that can be used for any GDS and ending with the most specific one, which is highly tuned for a class of diffusion problems. We then present comparative performance data for the three tools on four networks, showing how tailoring simulation codes for a more targeted problem space has the payoff of improved computational performance. We then present illustrative results. The first tool is a framework, and requires user-supplied functionality to quantify the population dynamics, while the latter two are self-contained simulation codes where the population dynamics are provided. Also, the first and third tools are based on agent-to-agent interaction graphs, while the second uses agent-to-location graphs where agents are located at particular places at particular times, and those agents that are co-located interact. These graphical representations are each well-suited to incorporate particular types of interventions. In particular, the latter two tools have sophisticated intervention capabilities that the first tool lacks.

4.2 InterSim

InterSim (Kuhlman et al. 2011) is a framework that can simulate any GDS. It is driven by two needs: (i) expressiveness in agent behaviors and (ii) fast turn-around time. The approach taken is to provide a software framework that handles all aspects of distributed simulation except agent behaviors and to enable users to write and plug in software called **node interaction models** (NIM) for these behaviors. Agent behavior is implemented in software that contains the local transition functions and aspects of the GDS extensions listed in Section 2. The second need, fast turn-around time, measures the time from problem specification until useful simulation results are being generated, and is the sum of times for software design, construction, and verification of the NIM (i.e., the dynamics model). Turn-around times on the order of 1 to 4 hours are routinely realized. This compares favorably with the several weeks (if not months) generally required to alter and validate a focused simulator for a different diffusion process. The susceptible-exposed-infected-recovered model (SEIR Newman 2003), for example, was implemented with a four-hour turn-around time. The two needs, taken together, illustrate the motivation for InterSim: we want a user to be able to implement *quickly* any GDS-based network dynamics that they desire.

The InterSim framework is implemented in C++ and uses the Message Passing Interface (MPI) for distributed processing. An instance of InterSim is run on each **processing element** (PE) (e.g., core) of a

compute node. InterSim contains data structures for storing and manipulating node and edge properties. There can be any number of integer, double, and character variables that describe node and edge properties, including state. Control flow is managed in InterSim as described below. It also provides communication patterns among worker PE, and between worker PE and the outputting PE. Finally, it provides interfaces for connectors, described next.

The most important connector from a user standpoint is the *interaction model connector*. This connector provides a completely general interface for agent behaviors, implemented through NIM. (Note that a NIM does not require an explicit FSM or PTTS for state transitions, permitting more varied system dynamics.) A NIM is implemented by the user and is compiled and linked with the framework. Each agent is associated with a unique instance of a NIM so that internal state and intermediate quantities can be evolved individually (edge properties and other local variables can also be evolved). This versatility comes at the price of a greater memory footprint. We have implemented various NIM for the SEIR model, many different threshold models, generalized cellular automata (Mortveit and Reidys 2007), and computer network communication algorithms, some of which contain multiple contagions and networks, as well as multiple dynamics mechanisms. The framework also provides a set of methods and objects that the NIM uses to retrieve properties for nodes and edges and to set the NIM results that will in turn be used by the framework to update new states of nodes across PE and to output quantities to file.

At a high level, the code operates as follows. Initially, the set of networks, agent and edge properties, dynamics model parameters, the number of diffusion instances to run (a diffusion instance is called an **iteration**), the maximum time for each diffusion instance, and base states of all agents and edges are read. For each iteration, all variables associated with the dynamics models and node and edge states are reset, and then states of selected seed agents, NIM properties, and other parameters are read in (e.g., seed states overwrite the base states of the agents, and enable variations among iterations). Time marches forward in discrete increments until the maximum time is reached. At each time, depending on update scheme, one or more agents (and one or more diffusion mechanisms and contagions) are evaluated for state update by MPI PE. Note that internal variables may be updated for an agent without a state change resulting. State updates are passed among PE and new state values are written to file. If a state update occurs, the parameters of the dynamics models for the new state of that agent are reset in a user-specified manner because the FSM may have cycles, so states can be entered multiple times. There are stopping criteria in the framework that permit early termination of an iteration; e.g., if no more state transitions can take place.

Two illustrative cases studies that use the framework are presented in (Kuhlman et al. 2011). The first examines multiple information-based contagions and multiple networks to model individuals divesting themselves of stocks during a financial downturn. Network structure effects are also examined and we show that as the fraction of network edges that are randomly rewired increases, the probability of a cascade (i.e., widespread divestiture) decreases, but the size of the cascade, if it occurs, increases. The second case study examines how members of a population change their minds back and forth between two alternatives. The two choices represent adoption of competing ideologies or opinions, and multiple states for each choice represent different degrees of conviction. For each state, there are three different mechanisms for state transition: (i) increase conviction, (ii) decrease conviction, and (iii) change opinion. We show how different prioritizations of the three mechanisms and how deterministic and probabilistic diffusion yield vastly different behaviors.

InterSim has a number of limitations. First, this implementation lacks the sophisticated intervention capabilities of EpiSimdemics and EpiFast discussed later. Local interventions, such as an agent extricating itself from a network based on its local neighborhood conditions, can be employed, but interventions based on global system state are not currently implemented. For example, an intervention such as "when the disease spreads through 5% of the population, all children stay home from school" is currently not possible. Also, because of the generality of the local transition functions, a change of state can require a lot of data to be passed among worker PE, degrading performance. Since each agent has its own instance of a C++

object to evolve its state (including the local transition function), a large memory footprint may result. Finally, if a dynamics model is very complicated, then longer turn-around times will result.

4.3 EpiSimdemics

EpiSimdemics (Barrett et al. 2008) is an interaction-based, HPC, highly resolved modeling and simulation system for representing and reasoning about large scale epidemics (Barrett et al. 2008, Bisset et al. 2009, Bisset et al. 2011). We describe programming abstractions and models that allow us to extend EpiSimdemics to better reason about real-world situations and to study the joint evolution of policy, disease dynamics, human behavior and social networks as an epidemic progresses.

At a high level, the simulator sends agents to locations based on their **schedules** (i.e., activity lists), and agents that are co-located interact at these locations. The set of all locations occupied by an agent in one time increment is called a set of **visits**. A time increment, often a day, is called a **phase**. Hence, we see that the implicit person-to-person interaction network changes by the second (if we are simulating by days) according to agents' schedules.

The computation structure of this implementation consists of three main components: persons, locations, and message brokers. We assume a parallel system consisting of N cores, or PE. Processing proceeds in the following manner:

1. Partitioning: Persons and locations are partitioned into N groups denoted by $P_1, P_2, \ldots, P_N$ and $L_1, L_2, \ldots, L_N$ respectively. Each PE then executes the EpiSimdemics algorithm on its local data set (P_i, L_i).
2. Computing Visit Data: The first step consists of computing a set of visits for each individual, P_i for the current day (or phase). This also involves computing any health state changes and applying events such as infections and interventions. A light-weight "copy" of each person (called a *visit message*) is then sent to each location (which may be on a different PE).
3. Computing Infections: Each location receives the visit messages and forms a discrete event simulation (DES) by collecting the messages into a time-ordered list of arrive and depart events. Using this data, each location computes contagion transmission for each individual at that location. Outcomes of these computations are then sent back to the "home" PE of each person.
4. Collecting Infection Messages: Infection messages for each person on a PE are merged and processed, and the resulting health state of each infected person is updated.

The EpiSimdemics model is used to explore the impact of agent behavior and public policy mitigation strategies on the spread of contagion over extremely large interaction networks (10^6–10^9 agents). To determine their local behavior, individual agents may consider their individual state (the contagion model), their demographics, and the global state of the simulation. Contagion and behavior are modeled as coupled **probabilistic timed transition systems** (PTTS), an extension of the well known finite state machine with two additional features: the state transitions are probabilistic and timed. A PTTS is a set of states. Each state has an id, a set of attribute values (the same attributes for each state of a given PTTS), a dwell time distribution, and one or more labeled sets of weighted transitions to other states. The label on the transition sets is used to select the appropriate set of transitions. The attributes of a state describe features possessed by an agent when in that state (e.g., infectivity, symptoms exhibited, or level of fear). Once an individual enters a state, the amount of time that they will remain in that state is drawn from the dwell time distribution. There may be several independent PTTS. An agent has a separate state and dwell time for each PTTS. As an optimization, there is only one copy of each PTTS in each parallel process. Each agent carries the state id and next transition time for each PTTS.

The PTTS and the interaction network are co-evolving, as the progression of each one potentially affects the other. In simple terms, who you meet determines whether you fall sick, and the progression of the disease may change who you meet (e.g., you stay home because you are sick). The co-evolution

can be much more complex, as an individual's schedule may change depending on information exchanged with others, the health state of people they contact even if no infection takes place (e.g., more people than usual are sick at work), or even expected contacts that do not happen (e.g., coworkers who are absent from work). Activities may also be affected by an individual's demographics (e.g., a person's income affects their decision to stay home from work). The interaction network can represent different types of interactions such as physical proximity or telephone communication.

Each individual agent has a set of schedules for different purposes. Examples include a normative schedule, one for school closures, one to use while staying home when sick, etc. Currently, all of the schedules must be precomputed. Ongoing work will add the capability to generate schedules on demand. Some schedules, such as self isolation at home, can be dynamically created as needed to save memory and decrease startup time. Others, such as going to the nearest medical facility, depend on an individual's location when the schedule change is invoked, and need to be computed during execution.

The **scenario** specifies the behavior of individuals (e.g., stay home when sick), as well as public policy (e.g., school closure when a specific proportion of the students are sick). There are two fundamental changes that can be made that will affect the spread of a contagion in a social network. All behavior and public policy interventions can be implemented through these changes. First, the probability of transmission of a contagion can be changed by changing the infectivity or susceptibility of one or more individuals. Second, edges can be added, removed, or altered in the social network, resulting in different individuals coming into contact for different amounts of time. The individual behaviors and public policy interventions in EpiSimdemics, collectively referred to as the scenario, expose these two changes in a way that is flexible, easy to understand for the modeler, and computationally efficient.

There are several restrictions on the type of problems that EpiSimdemics can efficiently handle. First, individuals can only affect other individuals through interactions that occur when they are co-located in space and time, and the interactions can be computed pair-wise. The results of all interactions involving an agent during a visit can be applied to that agent in any order. Second, an individual's PTTS state changes can be precomputed for an iteration. Third, changes to the network or individual agent state must be based on either local (to the agent) information, or global measures (e.g., the total number of infected agents in the system). Measures based on the local network (e.g., state of neighbors in the network) cannot be computed efficiently. Fourth, there is a minimum latent period, D_{min}. This is the amount of time that must pass between an agent's state changing due to an interaction, and that state change affecting other agents. Problems where the minimum latent period approaches 0 cannot be computed efficiently. Examples of such problems are computer malware, where an infected device is immediately able to infect other devices, and the physical spread of contamination, such as a virus, where one agent contaminates a surface that can then spread the virus to other agents.

4.4 EpiFast

EpiFast (Bisset et al. 2009) is a high-performance implementation of a subset of GDS $\mathscr{S} = (G, \mathbb{B}, \mathscr{F}, \mathscr{R})$ extended with *interventions* $\mathscr{I}$, with the following differences from the other two tools:

1. The graph G is explicitly given (as in InterSim), unlike EpiSimdemics, which builds this implicitly.
2. EpiFast implements a simple SEIR disease model for epidemic simulations.
3. The state transition function $\mathscr{F}$ is simple aggregation of multiple functions $f_{u,v}$, each corresponding to an edge incident on the node v. Function $f_{u,v}$ depends on nodes u, v and edge (u,v) and labels on $u, v, (u,v)$; and it is dynamic – *interventions* on u and/or v change $f_{u,v}$.
4. The update scheme $\mathscr{R}$ in EpiFast is synchronous. The GDS synchronizes all nodes' state changes and changes to G due to interventions in each time step.
5. $\mathscr{I}$: Interventions in EpiFast change network G, including its structure (e.g., in the case of non-pharmaceutical interventions, or NPI) and nodes' properties regarding their infectivities and vulnerabilities. EpiFast can handle a class of interventions, which is of form: *when predefined global*

conditions $g_1, g_2, \ldots$ and predefined local conditions $\ell_1, \ell_2, \ldots$ are satisfied for node v, change v's node properties and/or labels of a predefined subset of edges incident on v.

EpiFast is also implemented in C++/MPI. It is similar to InterSim and Episimdemics in that it runs on almost any distributed memory system, so long as the total of all available memory can hold the whole network and modeling parameters. In EpiFast there is one master PE and multiple worker PE. The master PE is responsible for passing messages and data, and coordinating synchronizations. The major computations for a diffusion process are undertaken by the worker PE. The network is partitioned online and stored on worker PE such that each node has a unique "owner" PE and all edges incident on this node "reside" together on the same PE. A worker PE can distinguish "local" nodes and "foreign" nodes and the master PE knows the "owner" PE of every node. This way, all data about foreign nodes are relayed by the master PE. For the problems we have considered to date, the master PE does not appear to be a bottleneck.

In each time step of a simulation, computations consist of interventions and diffusion. For interventions, the master PE and worker PE collaborate to evaluate global conditions. Then each worker PE evaluates local conditions for each node. Corresponding changes are applied to nodes that satisfy all conditions and to edges incident on them. For diffusion, each worker PE identifies its infectious nodes. For each such node, a PE computes adjacent nodes that it infects and sends messages about infected foreign nodes to their owner PE via the master PE. Then the system synchronizes for updating states of nodes.

Besides the limitations of EpiSimdemics described in Section 4.3, the following restrictions also apply to EpiFast. First, its local transition function implements a simple PTTS, commonly known as an SEIR model in epidemiology. It is still realistic and is extendible to cover a large set of diffusion models, including social models such as the spread of opinions. This model: (i) is predetermined for each node (although possibly heterogeneous) and static; (ii) provides for one way transitions: a node in S state can only stay in S or change to E state, a node in E can only stay in E or change to I, a node in I can only stay in I or change to R, a node in R can only stay in R; (iii) has a single probabilistic transition, occurring between S and E states. Second, the network is not completely dynamic. The network is stored in memory in a continuous and compact data structure. EpiFast allows labeling and relabeling the existing edges. Therefore, removing edges is easy by marking them as "inactive." However, adding new edges is difficult. For this reason, EpiFast requires that each node be adjacent to a predetermined set of other nodes during the simulation; i.e., the initial graph must contain all person-person contacts, and the average degree of the network needs to be $O(1)$ (so the contact graph cannot be a complete graph). Third, network edges have *quantity* labels (for example, duration of contact or probability of transmission) but no *ordering* labels. Therefore, it is impossible to order all transmissions computed in one time step. For example, suppose a simulation computes that two nodes u, w transmit the disease to node v. If we had known u meets v in the morning and w meets v in the afternoon, then we would know u is the true infector. Since we do not know which of (u,v) and (w,v) occurs first, EpiFast can only pick one randomly.

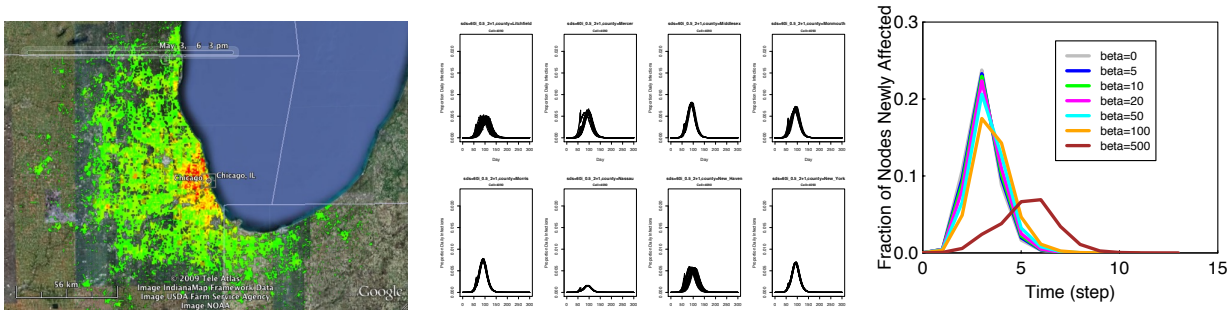
4.5 Performance Comparison Among Tools

An SEIR epidemiological model was used with each simulation tool to evaluate disease diffusion through four large U. S. cities. Each time step in a simulation represented one day, and each diffusion instance was run for 300 time steps. Given that an agent u is infectious, the probability of agent v contracting the disease is given by $p(v|u) = 1-(1-r)^{w(u,v)}$ where $w(u,v)$ is the edge weight and represents the duration of contact between u and v in minutes, and r is the probability of disease transmission for a contact of one unit time. In InterSim simulations, $r = 0.0005$, while values for EpiSimdemics and EpiFast are determined from distributions.

Simulation conditions were not identical across platforms. For example, InterSim used an exposed duration of zero, and a constant infectious duration of 4 time units for all agents. EpiFast, in contrast, used random assignments of exposed and infectious durations from distributions for each agent. Disease spread reached all agents in the InterSim simulations while the spread in EpiFast was roughly 50% of agents.

Table 2: Average execution times for one diffusion instance for each of the simulation codes for four networks. Averages were computed over 50 diffusion instances per network. There are 300 time steps per diffusion instance. The number of PE used with each tool differs because of different memory requirements.

Region	Population (million)	Edges (billion)	Average Degree	InterSim		EpiSimdemics		EpiFast	
				PEs	Time (s)	PEs	Time (s)	PEs	Time (s)
Miami	2.09	0.1	49	80	131.0	8	875	8	18.09
DC	3.75	0.2	54	80	247.9	16	812	16	22.53
Chicago	9.04	0.5	58	160	635.5	40	852	40	44.31
NYC	17.88	0.9	53	–	–	72	1224	72	81.54



(a) (b) (c)

Figure 1: Results generated from NDSSL tools. (a) shows the spread of influenza in the Chicago, IL area, with green showing low rates and red high rates of infection. (b) shows the proportion of infections by day for several counties in the New York City area. (c) shows how methods to inhibit the spread of contagions can be used to slow down their progression.

Numbers of PE and execution durations for one diffusion instance, averaged over 50 iterations with 300 time steps per iteration, are provided in Table 2 for each tool and population. Because InterSim instantiates a unique NIM object for each agent and provides individualized expressive node and edge labels, it requires more memory, so the number of PE needed is larger than that for EpiSimdemics and EpiFast. The larger number of PE requires more communication among them, which contributes to the greater execution times. Another contributor to the differences in execution times is the optimized EpiFast code for SEIR dynamics. Even with the differences in simulation conditions noted above, we clearly see the trade-off (i.e., inverse relationship) between simulator versatility and performance. These results serve to drive home the point that different types of simulators are useful, depending on context.

4.6 Illustrative Results

We provide representative results from simulations and briefly describe their value. Maps of high-intensity infectious regions like that of Figure 1(a) can provide decision-makers with information regarding where to focus resources. Likewise, Figure 1(b) shows outbreak sizes over time for various counties in the vicinity of New York City and provides information regarding where vaccines might be efficaciously delivered. Figure 1(c) illustrates that the progression of contagions like unfounded rumors, malware in computer networks, or financial contagions can be mitigated by finding a minimum set of critical nodes that do not pass on the contagion or are isolated from their neighbors. In this plot, we show that if only a small number $\beta = 500$ of critical nodes is available, which is less than optimal so that the contagion cannot be completely halted, then it can be slowed down and attenuated to give decision makers more time to identify other solutions. Simulations such as these enable us to evaluate protocols for controlling diffusion mechanisms.

5 CONCLUSIONS

In this survey, we motivated the use of GDS to reason about ABMS for systems that may consist of multiple networks, contagions, and heterogeneous agent behaviors that co-evolve. Our goals in simulation include understanding baseline system dynamics, but more importantly, understanding how to control and optimize dynamics. This is because, beyond simulation, an objective is to provide domain experts, or even the public at large, with information to improve public health and welfare. To this end, control often takes the form of exogenous interventions; e.g., reducing the spread of a human or computer virus, or of a malicious rumor by, respectively, providing antiviral drugs and software patches, and identifying critical agents that must be convinced to not pass on a rumor. Large networks, sophisticated agent behaviors, and large experimental designs to cover multi-dimensional parameter spaces of inputs, including interventions, require HPC simulation methods. We discussed three tools that we continue to develop and use. We believe that no single tool can satisfy all simulation needs. Some tools are highly optimized for a particular application space because they are used frequently and/or domain experts need data—and possible several sets of data over time—quickly to plan and to incorporate newly arriving field data during a crisis. For other tools, versatility is key in being able to simulate a wide range of dynamics without starting from scratch by developing new simulation capabilities.

ACKNOWLEDGMENTS

We thank our external collaborators and members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their suggestions and comments. This work has been partially supported by NSF Nets Grant CNS- 0626964, NSF HSD Grant SES-0729441, NIH MIDAS project 2U01GM070694-7, NSF PetaApps Grant OCI-0904844, DTRA R&D Grant HDTRA1-0901-0017, DTRA CNIMS Grant HDTRA1-07-C-0113, NSF NETS CNS-0831633, DHS 4112-31805, DOE DE-SC0003957, NSF CNS-0845700, NSF Netse CNS-1011769 and NSF SDCI OCI-1032677.

REFERENCES

Aaby, B., K. Perumalla, and S. Seal. 2010. "Efficient Simulation of Agent-Based Models on Multi-GPU and Multi-CoreClusters". In *Proceedings of the 3rd International ICST Conference on SimulationTools and Techniques*, SIMUTools '10, 29:1–29:10.

Allen, F., and D. Gale. 2000. "Financial Contagion". *The Journal of Political Economy* 108:1–33.

Atkins, K., J. Chen, V. Kumar, M. Macauley, and A. Marathe. 2009. "Locational Market Power in Network Constrained Markets". *Journal of Economic Behavior and Organization* 70:416–430.

Barrett, C. L., K. R. Bisset, X. Eubank, Stephenand Feng, and M. V. Marathe. 2008. "EpiSimdemics: An efficient algorithm for simulating the spreadof infectious disease over large realistic social networks". In *Proceedings of the ACM/IEEE Conference on High Performance Computing(SC)*, 37. IEEE Press.

Barrett, C. L., H. B. Hunt III, M. V. Marathe, S. S. R. D. J. Rosenkrantz, and R. E. Stearns. 2006. "Complexity of Reachability Problems for Finite Discrete DynamicalSystems". *J. Comput. Syst. Sci.* 72 (8): 1317–1345.

Bisset, K., J. Chen, X. Feng, and A. V. M. Marathe. 2009, June. "EpiFast: A fast algorithm for large scale realistic epidemic simulationson distributed memory systems". In *Proceedings of 23rd ACM International Conference on Supercomputing(ICS'09)*, 430–439. ACM Press. NYC, NY.

Bisset, K., X. Feng, M. Marathe, and S. Yardi. 2009. "Modeling interaction between individuals, social networks and publicpolicy to support public health epidemiology". In *Proceedings of the 2009 Winter Simulation Conference*, 2020 –2031. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bisset, K. R., A. M. Aji, M. V. Marathe, and Wu-chunFeng. 2011. "High-performance biocomputing for simulating the spread of contagionover large contact networks". In *Computational Advances in Bio and*

Medical Sciences, IEEE InternationalConference on, 26–32. Los Alamitos, CA, USA: IEEE Computer Society.

Borsari, B., and K. Carey. 2001. "Peer Influences of College Drinking: A Review of the Research". *Journal of Substance Abuse* 13:391–424.

Cangelosi, A., and D. Parisi. 2001. "Computer Simulation: A New Scientific Approach to the Study of LanguageEvolution". In *Simulating the Evolution of Language*, edited by A. Cangelosi and D. Parisi, 3–28. Springer.

Carothers, C., and K. Perumalla. 2010. "On Deciding Between Conservative and Optimistic Approaches on MassivelyParallel Platforms". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 678–687. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Castellano, C., S. Fortunato, and V. Loreto. 2009. "Statistical Physics of Social Dynamics". *Rev. Mod. Phys.* 81 (2): 591–646.

Centola, D., and M. Macy. 2007. "Complex Contagions and the Weakness of Long Ties". *American J. Sociology* 113 (3): 702–734.

Channakeshava, K., K. Bisset, M. Marathe, and A. V. andS. Yardi. 2011. "High performance scalable and expressive modeling environment tostudy mobile malware in large dynamic networks". In *Proceedings of 25th IEEE International Parallel & Distributed ProcessingSymposium*, 770–781.

Choi, H., S.-H. Kim, and J. Lee. 2010. "Role of network structure and network effects in diffusion of innovations". *Industrial Marketing Management* 39:170–177.

Christakis, N., and J. Fowler. 2007. "The Spread of Obesity in a Large Social Network Over 32 Years". *N. Engl. J. Med.* 357:370–379.

Colizza, V., R. Pastor-Satorras, and A. Vespignani. 2007. "Reactiondiffusion processes and metapopulation models in heterogeneousnetworks". *Nature Physics* 3:276–282.

Dodds, P., and D. Watts. 2005. "A Generalized Model of Social and Biological Contagion". *J. Theo. Biology* 232 (4): 587–604.

Domingos, P., and M. Richardson. 2001. "Mining the Network Value of Customers". In *Proc. ACM KDD*, 57–61.

Dreyer, P., and F. Roberts. 2009. "Irreversible k-Threshold Processes: Graph-Theoretical ThresholdModels of the Spread of Disease and Opinion". *Discr. Appl. Math.* 157:1615–1627.

D'Souza, R., M. Lysenko, and K. Rahmani. 2007. "SugarScape on Steroids: Simulating over a Million Agents at InteractiveRates". In *Proceedings of the Proceedings of Agent2007 Conference*.

Easley, D., and J. Kleinberg. 2010. *Networks, Crowds and Markets: Reasoning About A Highly ConnectedWorld*. New York, NY: Cambridge University Press.

Epstein, J. 2002. "Modeling Civil Violence: An Agent-Based Computational Approach". *PNAS* 99:7243–7250.

Epstein, J. 2007. *Generative Social Science: Studies in Agent-Based ComputationalModeling*. Princeton University Press.

Eubank, S., H. Guclu, V. S. A. Kumar, M. Marathe, A.Srinivasan, Z. Toroczkai, and N. Wang. 2004. "Modelling disease outbreaks in realistic urban social networks". *Nature* 429:180–184.

Ferguson, N. M., M. J. Keeling et al. 2003. "Planning for smallpox outbreaks". *Nature* 425 (6959): 681–685.

Germann, T. C., K. Kadau, I. M. Longini, Jr., and C. A. Macken. 2006, April11. "Mitigation strategies for pandemic influenza in the United States". *Proc. of National Academy of Sciences* 103 (15): 5935–5940.

Goldenberg, J., B. Libai, and E. Muller. 2001. "Talk of the Network: A Complex Systems Look at the Underlying Processof Word of Mouth". *Marketing Letters* 12:211–223.

Gould, R. 1993. "Collective Action and Network Structure". *American Sociological Review* 58:182–196.

Granovetter, M. 1978. "Threshold Models of Collective Behavior". *American J. Sociology* 83 (6): 1420–1443.

Grimmett, G. 1989. *Percolation*. Springer.

Gruhl, D., R. Guha, D. Liben-Nowell, and A. Tomkins. 2004. "Information Diffusion Through Blogspace". In *Proc. WWW*, 491–501.

Guha, R., R. Kumar, P. Raghavan, and A. Tomkins. 2004. "Propagation of Trust and Distrust". In *Proc. WWW*, 403–412.

Habiba, Y. Yu, T. Berger-Wolf, and J. Saia. 2008. "Finding Spread Blockers in Dynamic Networks". In *Proc. SNA-KDD Workshop*, 55–76.

Haldane, A., and R. May. 2011. "Systemic Risk in Banking Ecosystems". *Nature* 469:351–355.

Halloran, M., N. Ferguson, I. L. S. Eubank, D. C. B. Lewis, S. Xu, C. Fraser, A. Vullikanti, T. Germann, D. Wagener, R. Beckman, K. Kadau, C. Barrett, C. Macken, D. Burke, and P. Cooley. 2008. "Modeling Targeted Layered Containment of an Influenza Pandemic inthe United States". *PNAS* 105 (12): 4639–4644.

Hui, C., M. Goldberg, M. Magdon-Ismail, and W. A.Wallace. 2010. "Agent-Based Simulation of the Diffusion of Warnings". In *Agent-Directed Simulation Symposium (ADS '10) as part of the 2010Spring Simulation MultiConference (SpringSim '10)*, 9.

Hwu, W. 2011. *GPU Computing Gems*. Elsevier Morgan Kaufmannnce. See "Chapter 21. Template-Driven Agen-Based Modeling and Simulationwith CUDA" by P. Richmond and D. Romano.

Hybinette, M., E. Kraemer, Y. Xiong, G. Matthews, and J. Ahmed. 2006. "SASSY: A Design for Scalable Agent-Basd Simulation System Usinga Distributed Discrete Event Infrastructure". In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 926–933. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Karaoz, U., T. Murali, S. Letovsky, Y. Zheng, C. D. C. Cantor, and S. Kasif. 2004. "Whole-Genome Annotation By Using Evidence Integration in Functional-LinkageNetworks". *Proceedings of the National Academy of Sciences* 101 (9): 2888–2893.

Kempe, D., J. Kleinberg, and E. Tardos. 2003. "Maximizing the Spread of Influence Through a Social Network". In *Proc. ACM KDD*, 137–146.

Kempe, D., J. Kleinberg, and E. Tardos. 2005. "Influential Nodes in a Diffusion Model for Social Networks". In *Proc. ICALP*, 1127–1138.

Kuhlman, C., A. Kumar, M. Marathe, H. Mortveit, S. S. G. Tuli, S. Ravi, and D. Rosenkrantz. 2011. "A General-Purpose Graph Dynamical System Modeling Framework". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kuhlman, C., V. Kumar, M. Marathe, S. Ravi, and D. Rosenkrantz. 2010. "Finding Critical Nodes for Inhibiting Diffusion of Complex Contagionsin Social Networks". In *Proceedings of ECML PKDD*, 111–127.

Kuhlman, C., M. Marathe, S. Ravi, and D. Rosenkrantz. 2010. "Exploiting Network Structure in Enhancing Diffusion of Complex Contagions". In *Proceedings of the Analysis of Complex Networks (ACNE) Workshop ofECML PKDD*, 20–34.

Kuran, T. 1989. "Sparks and Prairie Fires: A Theory of Unanticipated Political Revolution". *Public Choice* 61:41–74.

Longini, I., A. Nizam et al. 2005. "Containing Pandemic Influenza at the Source". *Science* 309 (5737): 1083–1087.

Markose, S., J. Arifovic, and S. Sunder. 2007. "Advances in Experimental and Agent-Based Modelling: Asset Markets,Economic Networks, Computational Mechanism Design, and Evolutionary Game Dynamics". *Journal of Economic Dynamics and Control* 31:1801–1807.

May, R., S. Levin, and G. Sugihara. 2008. "Ecology for Bankers". *Nature* 451:893–895.

Mertsalov, K., M. Magdon-Ismail, and M. Goldberg. 2008. "Models of Communication Dynamics for Simulation of Information Diffusion". In *Proceedings of Advances in Social Networks Analysis and Mining (ASONAM2009)*, 44–49.

Mitchell, M. 2009. *Complexity: A Guided Tour*. Oxford University Press.
Monge, P. R., and N. S. Contractor. 2003. *Theories of Communication Networks*. Oxford University Press.
Mortveit, H., and C. Reidys. 2007. *An Introduction to Sequential Dynamical Systems*. Springer.
Newman, M. 2003. "The structure and function of complex networks". *SIAM Review* 45 (2): 167–256.
North, M., and C. Macal. 2009. "Foundations of and Recent Advances in Artificial Life Modeling withRepast 3 and Repast Simphony". In *Artificial Life Models in Software*, 37–60. Springer.
O'Loughlin, J., I. Karp, T. Koulis, G. Paradis, and J. DiFranza. 2009. "Determinants of First Puff and Daily Cigarette Smoking in Adolescents". *American J. Epidemiology* 170 (5): 585–597.
Park, A., and R. Fujimoto. 2009. "Efficient Master/Worker Parallel Discrete Event Simulation". In *Proc. of the 23rd Workshop on Principles of Advanced and DistributedSimulation*, 145–152.
Parker, J. 2007. "A Flexible, Large-Scale, Distributed Agent Based Epidemic Model". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1543–1547. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
Parker, J., and J. Epstein. 2011. "A Distributed Platform for Global-Scale Agent-Based Models of Disease-Transmission". *ACM Transactions on Modeling and Computer Simulation* 22.
Perumalla, K. 2005. "μsik: A Micro-Kernel for Parallel/Distributed Simulation Systems". In *Proceedings of the 19th Workshop on Principles of Advanced and DistributedSimulation*, 185–192.
Perumalla, K., and S. Seal. 2011. "Discrete Event Modeling and Massively Parallel Execution of EpidemicOutbreak Phenomena". *SIMULATION, to appear*.
Schelling, T. 1978. *Micromotives and Macrobehavior*. W. W. Norton and Company.
Siegel, D. 2009. "Social Networks and Collective Action". *Americal Journal of Political Science* 53:122–138.
Sturm, R. 2002. "The Effects Of Obesity, Smoking, And Drinking On Medical ProblemsAnd Costs". *Health Affairs* 21 (2): 245–253.
Wang, G., M. Salles, B. Sowell, X. Wang, T. Cao, A.Demers, J. Gehrke, and W. White. 2010. "Behavioral Simulations in MapReduce". *Proceedings of the VLDB Endowment* 3 (1): 952–963.
Watts, D. 2002. "A Simple Model of Global Cascades on Random Networks". *PNAS* 99 (9): 5766–5771.

AUTHOR BIOGRAPHIES

KEITH BISSET is a Senior Research Associate in the Virginia Bioinformatics Institute at Virginia Tech. His email address is kbisset@vbi.vt.edu.

JIANGZHUO CHEN is a Senior Research Associate in the Virginia Bioinformatics Institute at Virginia Tech. His email address is chenj@vbi.vt.edu.

CHRIS J. KUHLMAN is a graduate student in the Computer Science Department at Virginia Tech. His email address is ckuhlman@vbi.vt.edu.

VULLIKANTI S. ANIL KUMAR is an Associate Professor in the Computer Science Department and the Virginia Bioinformatics Institute at Virginia Tech. His email address is akumar@vbi.vt.edu.

MADHAV V. MARATHE is a Professor in the Computer Science Department and the Virginia Bioinformatics Institute at Virginia Tech. His email address is mmarathe@vbi.vt.edu.