1-1-2014

INDEMICS: An Interactive High-Performance Computing Framework for Data Intensive Epidemic Modeling

Keith R. Bisset
Virginia Bioinformatics Institute

Jiangzhuo Chen
Virginia Bioinformatics Institute

Suruchi Deodhar
Virginia Bioinformatics Institute

Xizhou Feng
Marquette University, xizhou.feng@marquette.edu

Yifei Ma
Virginia Bioinformatics Institute

See next page for additional authors

Accepted version. *ACM Transactions on Modeling and Computer Simulation*, Vol. 24, No. 1 (January 2014). DOI. © 2014 Association for Computing Machinery. Used with permission.

Authors
Keith R. Bisset, Jiangzhuo Chen, Suruchi Deodhar, Xizhou Feng, Yifei Ma, and Madhav V. Marathe

Published in final edited form as:
ACM Trans Model Comput Simul. 2014 January ; 24(1): . doi:10.1145/2501602.

Indemics: An Interactive High-Performance Computing Framework for Data Intensive Epidemic Modeling

Keith R. Bisset, NDSSL,
Virginia Bioinformatics Institute, Virginia Tech

Jiangzhuo Chen, NDSSL,
Virginia Bioinformatics Institute, Virginia Tech

Suruchi Deodhar, NDSSL,
Virginia Bioinformatics Institute, Virginia Tech, Department of Computer Science, Virginia Tech

Xizhou Feng, NDSSL,
Virginia Bioinformatics Institute, Virginia Tech, Department of Mathematics, Statistics, and Computer Science, Marquette University

Yifei Ma, NDSSL, and
Virginia Bioinformatics Institute, Virginia Tech, Department of Computer Science, Virginia Tech

Madhav V. Marathe, NDSSL
Virginia Bioinformatics Institute, Virginia Tech, Department of Computer Science, Virginia Tech

Abstract

We describe the design and prototype implementation of Indemics (Interactive Epidemic Simulation)—a modeling environment utilizing high-performance computing technologies for supporting complex epidemic simulations. Indemics can support policy analysts and epidemiologists interested in planning and control of pandemics. Indemics goes beyond traditional epidemic simulations by providing a simple and powerful way to represent and analyze policy-based as well as individual-based *adaptive interventions*. Users can also stop the simulation at any point, assess the state of the simulated system, and add additional interventions. Indemics is available to end-users via a web-based interface.

Detailed performance analysis shows that Indemics greatly enhances the capability and productivity of simulating complex intervention strategies with a marginal decrease in performance. We also

Correspondence authors: Jiangzhuo Chen (chenj@vbi.vt.edu) and Madhav V. Marathe (mmarathe@vbi.vt.edu).

Preliminary versions of this paper appeared in *Proc. 24th International Conference on Supercomputing 2010* as “Indemics: An interactive data intensive framework for high-performance epidemic simulation” and in *Proc. Winter Simulation Conference 2011* as “Efficient implementation of complex interventions in large scale epidemic simulations”.

demonstrate how Indemics was applied in some real case studies where complex interventions were implemented.

General Terms

Design; Human Factors; Performance

Additional Key Words and Phrases

parallel computation; interactive computation; infectious disease; network dynamics; modeling and simulation

1. INTRODUCTION

Pandemics have a significant impact on public health and human society. The 2009 H1N1 influenza pandemic exemplified the scale and scope of the societal impacts of global outbreaks of infectious diseases — 60 million cases of H1N1 were reported between April 2009 and April 2010 in the United States. The 2009 H1N1 pandemic led to 274,000 hospitalizations and 12,470 deaths, according to the estimates provided by the Centers for Disease Control and Prevention [CDC 2010; Lipsitch et al. 2011]. Fortunately, H1N1 was not very virulent. This, coupled with a globally coordinated response by world and national public health authorities, reduced the overall casualties. Controlling future pandemics and reducing their economic and social burden will be challenging due to a number of societal trends. This includes increased and denser urbanization, increased local as well as global travel, and a generally older and immuno-compromised population. These trends are likely to continue in future. A recent report by McKinsey predicts that by 2025, 600 million of the world's inhabitants will live in 440 emerging cities in the developing world that are likely to lack good public health infrastructure [Dobbs et al. 2011]. As a result, a pandemic caused by a highly virulent agent can have a far-reaching impact.

Computational models play an important role in elucidating the space-time dynamics of epidemics. The H1N1 pandemic reaffirmed the need for developing analytical tools and methods to detect, assess, and respond to future pandemics; see [Lipsitch et al. 2011; Kerkhove and Ferguson 2012; Wu and Cowling 2011; Fineberg and Wilson 2009]. The role of computational models is all the more important due to ethical reasons and lack of data. Computational models can assist in evaluating a diverse set of interventions aimed at preventing and controlling pandemics. Computational models provide a powerful tool to study the role of individual behavior and public policies in containing the pandemics.

2. RELATED WORK

Traditionally, mathematical and computational modeling of epidemics has focused on aggregate models using coupled rate equations. See [Anderson and May 1991; Kermack and McKendrick 1927; Bailey 1975; Hethcote 2000; Vynnycky and White 2010] for comprehensive reviews of this approach. In this approach, a population is divided into subgroups (compartments) according to an individual's health state (e.g., susceptible, exposed, infected, and recovered) and demographics. The evolution of the infectious disease is characterized by ordinary differential equations. Over the years, aggregate differential equation-based models have been employed to analyze patterns of epidemic spread and corresponding mitigation strategies [Rvachev and Longini 1985; Hufnagel et al. 2004]. The spatio-temporal epidemiological modeler (STEM) [Edlund et al. 2010] is an open source disease simulation system using this approach. Aggregate differential equation-based models have proved to be immensely successful. They yield analytical expressions for a number of important epidemic parameters, including number of infected individuals, mortality rate, etc. Extensions of the models to represent more complicated compartments that capture either the disease history or a specific group of individuals in a population have also been studied. An important assumption in all aggregate differential equation-based models is homogeneous mixing. This limits use of these models for spatially sensitive processes.

Another methodology used to model the epidemic spread process is to use spatially explicit models developed using cellular automata [Sirakoulis et al. 2000; Meyers et al. 2006; White et al. 2007a]. Cellular automata based models allow one to represent individual level interactions. Nevertheless, these interactions are highly regular and hence do not capture the complicated social interaction patterns in an urban region.

In recent years, high-resolution individual-based computational models have been developed to support planning, control and response to epidemics. These models support networked epidemiology—the study of epidemic processes over explicit social contact networks. Research in this area can be divided into three distinct subareas.

The first subarea aims to develop analytical techniques and computer simulations over classes of progressively sophisticated random graphs; see [Meyers 2007; Meyers and Dimitrov 2010; Pastor-Satorras and Vespignani 2002; Barrat et al. 2008; Newman et al. 2002] for an extensive and recent discussion on this topic. These models relax the mean field assumption to some extent but still use the inherent symmetries in random graphs to analytically compute important epidemic quantities of interest. The primary goal of these results is to obtain closed-form analytical results.

The second subarea aims to develop individual-based models using important statistics of a region. The two important statistics used are: (*i*) density (which is usually obtained using LandScan data) and (*ii*) basic census information that provides the demographic distribution of individuals within a population. A simple template is used to represent a community, and these communities are joined hierarchically to obtain larger regions. See [Germann et al. 2006; Ferguson et al. 2006; Ferguson et al. 2003] for examples of this approach. These models can be extended to obtain hybrid models as well. In a hybrid model, counties are represented as nodes and edges are added between counties to capture the movement of individuals — see [Colizza et al. 2007; Merler and Ajelli 2010; Ajelli et al. 2010; Colizza et al. 2005] for a comparative study. Epidemic dynamics within a county are computed using an individual-based model. The dynamics over a network of counties are captured using coupled rate equations. The Global-Scale Agent Model (GSAM) in [Parker and Epstein 2012] is a high-performance agent-based epidemic model capable of simulating a disease outbreak in a population of several billion agents. FluTE in [Chao et al. 2010] is a publicly available parallel simulation engine that also falls in this broad category of models.

The final class of models use the most realistic representation of social contact networks; see [Keeling and Eames 2005; Meyers 2007; Barrett et al. 2008; Eubank 2002]. EpiSims [Eubank 2002], EpiSimdemics [Barrett et al. 2008; Bisset and Feng et al. 2009], and EpiFast [Bisset and Chen et al. 2009] model each individual in the United States with detailed demographic profiles and daily activities. They are interaction-based simulation systems, where disease spreads via social interactions between individuals. Perumalla and Seal [2011] build on the work in [Barrett et al. 2008] to obtain a highly scalable simulation system; the paper contains a number of interesting technical ideas that lead to substantially improved scaling.

The primary focus of the above models is on simulating disease transmission. As a result, these simulations do not have well defined modules to represent policies and interventions — interventions are programmed in an ad hoc manner and are not transparent to the user. Moreover, none of the simulations support *adaptive interventions* — interventions in which a user formulates new interventions based on assessing the current system state.

Finally, our work is also related to recent work on building scalable database-oriented methods to support massively multi-player online games (MMOGs) [White et al. 2007b; Sowell et al. 2009; Wang et al. 2010]. The authors explore the use of databases to support real-time strategies for such games. The real-time strategies are formulated using relational algebra, and data computation is implemented by relational databases. In a recent paper, [Wang et al. 2010] have extended their work by showing how certain social simulations can be implemented using a MapReduce framework.

3. OUR CONTRIBUTIONS

Here, we present Indemics (*Interactive Epidemic Simulation*) — a high-performance computing (HPC) oriented modeling environment to support networked epidemiology. Indemics is designed to support real-time planning, situation assessment, and course of action analysis, as it pertains to public health epidemiology. It extends the current computational modeling environments in several ways. This includes: (*i*) providing a computational framework that allows a user to easily represent a broad range of policy-based as well as individual-based interventions; (*ii*) allowing a user to stop the simulation to assess the system state and to formulate interventions based on the system state; in other words policies and individual behaviors that are adaptive and sensitive to a system state; and (*iii*) allowing a user to access the system via a web-based interface; this makes Indemics accessible to public health analysts and policy makers who are not necessarily computing experts. Indemics is designed to improve the overall human productivity and ease of use of epidemic modeling environments without sacrificing computational efficiency. Current simulation frameworks primarily focus on the epidemic transmission process; as a result, interventions and policies are implemented in an ad hoc manner when using these simulations. Additionally, in many simulations, these components are intertwined with the epidemic transmission process. As a result, (*i*) changes to the system and scenario descriptions have to be done by the software developers who are intimately familiar with the code and (*ii*) software maintenance and reuse becomes cumbersome and error prone. In designing Indemics, care has been taken to address these shortcomings. Indemics is based on three important technical ideas.

First, we develop a set of abstractions that allow us to *decouple* three primary components of an epidemic simulation: (*i*) the data intensive and complicated intervention and behavioral adaptation component; (*ii*) the data-intensive state assessment; and (*iii*) the relatively generic but computationally intensive disease transmission component. The first two components often demand flexibility while the last requires substantial computational speed. Existing HPC-based simulation systems do not explicitly decouple them. This limits the ways in which the state of the system can be interrogated. It also requires additional effort from an expert and thus limits the possible interventions that can be simulated. The decoupled system architecture allows us to independently optimize the intervention and behavioral adaptation simulations and the disease transmission simulation. The abstractions formalize the communication and data transfer that takes place when these modules interact. It requires a delicate balance between the amount of data that is exchanged and the frequency of this exchange. This turned out to be an extremely important aspect of Indemics and ultimately led to a simulation environment that had an acceptable execution time and substantially improved human productivity. Our abstractions are based on a model called the co-evolving graphical discrete dynamical system (CGDDS). CGDDS is used to model the problem of interaction based epidemic propagation and corresponding intervention simulation. In addition, we overlay a Partially Observable Markov Decision Process (POMDP) over CGDDS to model intervention policies.

The second technical idea is the use of a relational database management system (RDBMS) to store data that drives the simulation as well as the data that is produced by the simulation as it progresses in time. Our work is motivated by a similar approach advocated by Heber and Gray in the context of finite element mesh simulations [Heber and Gray 2007a; 2007b]. Extending the work of Heber and Gray, we show how an RDBMS can be used not only to store the input data and the output data produced by the simulations, but also as a part of the simulation engine itself. The data stored in the database consists of four important components: (*i*) data corresponding to the demographic attributes associated with each individual, (*ii*) a dynamic social contact network that represents the set of proximity relationships between individuals (who meets whom at what location and when), (*iii*) a temporal data set that captures the disease transmission network (also called a dendogram), and (*iv*) data summarizing the interventions—when they are applied, to whom they are applied, and how they are applied. The dendogramis comprised of a time series representing when an individual was infected and the individual who transmitted the infection. These four datasets capture the basic elements of an epidemic simulation. Each dataset is conveniently stored in a relational database (other kinds of databases can also be considered). Situational assessment questions as well as interventions and behavioral adaptations that a policy maker may wish to implement are implemented using a relational database query language such as SQL (Structured Query Language). SQL is a natural fit to represent interventions and behavioral adaptations. Using SQL a user can specify the specific interventions naturally without worrying about how they are implemented.

Moreover, we can leverage built-in query optimization engines and automatic query parallelization available as a part of today's database systems to improve computational efficiency. To intuitively see why SQL would be a natural fit for this domain, note that a situation assessment query typically consists of finding the health state of a set of individuals in a spatio-temporal region. SQL is an appropriate language to specify this, since SQL specifies queries using set operations. SQL is more high level than C++ or many other programming languages. It is functional as opposed to imperative—users specify what is wanted rather than how to compute it [Heber and Gray 2007a; 2007b]. Implementing this within existing simulation codes would require one to write potentially complicated codes to select such sets. SQL-based queries can be created or modified without modifying the HPC-code that handles the disease transmission component. This greatly improves the human productivity and ease of use.

Our final technical idea is a set of software techniques to start and stop the disease transmission simulation EpiFast that is used within the Indemics framework. EpiFast is a highly efficient high performance computing simulation environment [Bisset and Chen et al. 2009] designed specifically to simulate disease transmission over large social networks. See [Bisset et al. 2011] for the description of the basic algorithm, its performance analysis and a comparison with other known simulations. Indemics was designed to be interactive — a user can stop the simulation and interrogate the system state before deciding the next set of actions. EpiFast was not originally designed to support this requirement. We modified EpiFast so that the simulation can be stopped after any simulation day, new interventions added and existing interventions modified, and then resumed. The stopping criteria can be a fixed number (e.g., day 30) or a function of the three data types discussed earlier. When EpiFast is stopped, the incremental changes in the system state are transferred to the database. The system state relevant to EpiFast is also held in memory. The abstraction and the data type discussed earlier prove useful—it turns out that the relevant system state needed to re-start EpiFast from the point at which it was stopped is quite modest.

Implementation

An initial version of Indemics has been implemented using the software services paradigm and is currently in use. It is accessible to a user over the Internet via the world wide web. This makes Indemics accessible pervasively. We have carried out an initial performance analysis of Indemics — we can show that the system scales easily to support realistic case studies over social contact networks with millions of agents. We have recently used Indemics for two interesting case studies; our results appear in [Marathe et al. 2011; Liao et al. 2012].

Paper organization

In Section 4, we formalize our simulation framework as an interactive CGDDS. In Section 5, we formalize the concept of interventions in epidemics. In Section 6, we introduce Indemics, an interactive epidemic simulation framework. We evaluate its performance in Section 7. The capabilities of Indemics are demonstrated in Section 8 using a case study that was difficult to carry out with other simulation tools. Finally, we conclude in Section 9. Basic material covering within and across host disease models, epidemic propagation over a network, and interventions is described and illustrated in the Appendix. Specifically we refer the reader to

Figure 14 in the Appendix for a concrete example of stochastic disease transmission in a social contact network with and without interventions. We also refer the interested reader to [Barrett et al. 2008; Bisset and Chen et al. 2009] for more details.

4. FORMALIZATION OF INTERACTIVE EPIDEMIC SIMULATION

The formal mathematical model consists of two parts: (*i*) a CGDDS framework that captures the co-evolution of disease dynamics, social network, and individual behavior, and (*ii*) a POMDP that captures various control and optimization problems formulated on the phase space of this dynamical system. We will first extend the CGDDS described in [Barrett et al. 2009] in Section 4.1 so that (*i*) it covers normal pharmaceutical and non-pharmaceutical interventions; (*ii*) the system progresses in a synchronized manner; and (*iii*) the vertex state modification functions are separated into propagation functions, which change the vertex health state based on information of its neighborhood, and intervention functions, which change other states of a vertex based on global information. Later in Section 4.2, we introduce an extended POMDP to model the intervention decision making process as a dynamic mapping from the system states to the intervention actions. Finally in Section 4.3, we overlay the extended POMDP over the extended CGDDS to form an interactive CGDDS, which formally describes our Indemics framework. We elaborate on the extended CGDDS using EpiFast [Bisset and Chen et al. 2009], and the interactive CGDDS using Indemics.

4.1. Extended CGDDS

An extended CGDDS, represented by symbol $\mathscr{S}$ over a given domain $\mathbb{D}$of state values and a given domain $\mathbb{L}$of label values is a triple (G, $\mathscr{F}$, W), whose components are as follows.

1. Graph $G(V,E)$:

   Let the vertex set $V = \{v_1,v_2,\ldots v_n\}$ represent the set of $n \geq 1$ agents (individuals). For each vertex v_i, let vector s_i denote its states $s_i=(s_i^1, s_i^2, \cdots s_i^k) \in \mathbb{D}=(\mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_k)$, where k is the number of states of vertex v_i. Intuitively, the states comprise the agent's health state, behavioral state (e.g., level of fear, risk aversion, etc.) as well as static demographic attributes.

   Let the edge set $E = \{e_1,e_2,\ldots e_m\} \subseteq (V \times V)$ represent the contacts between agents. For any edge $e \in E$, let vector ℓ_e denote its labels $\ell_e=(\ell_e^1, \ell_e^2, \ldots, \ell_e^h) \in \mathbb{L}=\mathbb{L}_1 \times \mathbb{L}_2 \times \cdots \times \mathbb{L}_h)$, where h denotes the number of labels. In our social contact network, the edge labels include the contact duration and the contact type (home, school, work, shopping, or others).

2. Functions $\mathscr{F} = (f,g^V,g^E)$, where f is a set of local transition functions; g^V is a set of vertex modification functions; and g^E is a set of edge modification functions.

   For each vertex v_i, let $f_i : \mathbb{D} \times \mathbb{D}^{V_i} \times \mathbb{L}^{E_i} \mapsto \mathbb{D}$be its local state transition function, where V_i and E_i are the neighboring vertices and edges of v_i. Normally, V_i are vertices adjacent to v_i and E_i are edges incident on v_i. The f_i function corresponds to the propagation process which changes the states of an agent based on (*i*) the current states of the agent, (*ii*) the states of all its neighboring agents, and (*iii*) the current labels on the contact edges with its neighboring agents. These variables determine a distribution over $\mathbb{D}$ then a state is chosen from the distribution as the output of f_i. So f_i is a random function.

Let $g^V = \{g_1^V, g_2^V, \ldots g_{k_V}^V\}$ be a set of k_V vertex modification functions, where each $g_j^V : \mathbb{D}^V \times \mathbb{L}^E \mapsto \mathbb{D}^V$ directly changes states of vertices based on the current state of the whole graph. We assume *V* is constant.

Let $g^E = \{g_1^E, g_2^E, \ldots g_{k_E}^E\}$ be a set of k_E edge modification functions, where each $g_j^E : \mathbb{D}^V \times \mathbb{L}^E \mapsto \mathbb{L}^{V \times V}$ changes the set of edges and the edge labels based on the current state of the whole graph. Note that g^E functions may add new edges to *G*; and that we abuse the notation a little with the assumption that for vertex-pair *u*, *v* with no contact, the labels on them are null.

Note that g^V and g^E can be random functions, too.

3. String *W* over the alphabet $g^V \cup g^E$:

   Let $W = w_1^1 w_1^2 \ldots w_1^{j_1} \ldots w_t^1 w_t^2 \ldots w_t^{j_t} \ldots w_T^1 w_T^2 \ldots w_T^{j_T}$ be a schedule of modifications on graph *G*, including its vertex states and edge labels, where *T* represents the number of time steps. The *t*th substring of *W*, $w_t^1 w_t^2 \ldots w_t^{j_t}$, denotes the updates on vertices and edges at time step *t*. The f_i functions are implicit in the schedule and they are applied at each time step — see Algorithm 1.

In an extended CGDDS representing epidemic dynamics in a social contact network, *G* is the contact network, *f* functions correspond to between-host disease progression, g^V functions correspond to within-host disease progression, pharmaceutical interventions (**PI**'s, e.g. antiviral, vaccination), and behavioral adaptations that directly change people's states (e.g., increase of fear level as the epidemic takes off, use of face masks), and g^E functions correspond to non-pharmaceutical interventions (**NPI**'s, e.g., school closure, quarantine and social distancing) that change the graph structure. Public policy is a combination of **PI**'s and **NPI**'s that simultaneously affects a group of individuals. Individual behavioral adaptations affect a single individual and his interactions with his neighbors in the social contact network. Public policy provides a guideline; individuals interpret and comply with the policy based on their individual attributes. For example, public policy might suggest that individuals use face masks. Individuals within a population comply with this policy decision based on their perception of disease, risk aversion, economic condition, etc.

Note that although theoretically an extended CGDDS model can be implemented in a simulation tool, it is difficult if not impossible to have a complete representation of the CGDDS in the software code. For example, in the current EpiFast implementation, $\mathbb{D}$is fixed; and expanding $\mathbb{D}$needs significant code changes. Also, the available g^V and g^E functions are very limited; a new modification function usually means nontrivial code changes. In Table I, we elaborate on the extended CGDDS using our EpiFast implementation [Bisset and Chen et al. 2009] as a concrete example.

Algorithm 1 shows the disease propagation loop with static interventions in the extended CGDDS. Note that the propagation step is computed in a synchronized way among vertices to make sure that the execution order does not matter. That is, the state updates are realized only when all vertices have finished executing the local state transitions (f_i). A new vertex state is computed for each vertex based on a snapshot of the current system; then each vertex realizes its new state.

Although the local transition in this algorithm has been precisely modeled, the intervention model is not close to reality: intervention scenarios are deterministic and given in the schedule string W before the start of the computation. The extended CGDDS does not seem adequate to model real-world systems such as an epidemic evolution. The interventions in an epidemic usually involve decision making processes, both at the public health level and at the individual level, based on the ongoing progress of the epidemic. The interventions (g^V and g^E functions) co-evolve with the propagation process. To this end, we introduce POMDP to capture the online decision making process for interventions.

Algorithm 1

Pseudo-code of computations in the extended CGDDS. Note that the functions that modify the vertex states or edge labels (policies or individual behavioral adaptations) are applied sequentially. In contrast, the disease states of vertices are updated in parallel.

```
for $t = 0$ to $T$ do
    let $w_t^1 w_t^2 \ldots w_t^{j_t}$ be the $t^{\text{th}}$ substring in $W$;            /* interventions for this time step */
    apply in increasing order of $j$, $1 \leq j \leq j_t$, function $w_t^j$ (function from sets $g^V$ or $g^E$) to modify vertex states
    or edge labels;
    for each vertex $v_i \in V$ in parallel apply function $f_i$ to change the health state;        /* propagation */
```

4.2. Extended POMDP

We model the decision making process for interventions in the schedule string *W* of the extended CGDDS as a POMDP, which has been widely employed to model various control and optimization problems. We make a necessary and reasonable modification on POMDP to model the intervention generation. We specify our extended POMDP using the terminology in [Mundhenk et al. 2000]. Recall that a POMDP *M* consists of: a finite set of states *S*, an initial state $s_0 \in S$, a finite set of actions *A*, a finite set of observations *O*, a probabilistic state transition function *ts*, an observation function $o : S \mapsto O$, and a reward function *r*, which gives the reward for taking action $a \in A$ while the system state is $s \in S$. Our extended POMDP is formally specified as follows.

1. States $S \subseteq (\mathbb{D}^V \times \mathbb{L}^{V\times V})$ are all possible vectors of vertex states and edge labels. Each system state is a vector of length $n+\left(\begin{array}{c} n \\ 2 \end{array}\right)$.
2. Actions *A* are interventions that modify vertex states and edge labels.
3. State transition *ts* computes the disease propagation with interventions.
4. Reward function *r* can be the number of infected cases, or it can be a combination of the economic and social costs of applying the interventions and the economic and social costs of the disease outbreak (mortality and morbidity), or it can consist of vertex level local functions. Maximization of the output of *r* maps observations to actions. The policy to determine actions based on observations can be history dependent $\pi_h : O^* \mapsto A$ [Mundhenk et al. 2000].

With the extended POMDP overlaid on the extended CGDDS, in each time step, CGDDS drives the system to a new state, POMDP derives intervention actions based on the observed system state, and the derived actions update CGDDS. This is described in Algorithm 2. Compared to Algorithm 1, the interventions in Algorithm 2 are dynamic and the actions are related to the run-time system states. The mapping from system states to intervention actions in Algorithm 2 is not decoupled in current modeling environments. This has the following weaknesses: (*i*) unless all possible intervention scenarios are known and can be enumerated in the system, re-engineering on the intervention mapping for new policy scenarios is inevitable; (*ii*) the data that can be included in the vertex state for computing interventions is limited to what is already implemented within the simulation tool; (*iii*) there is no control and optimization based on the reward function.

4.3. Interactive CGDDS

Let *DB* be a data management system and *Q* be a set of queries that map from *O* (observations) to *A* (actions). We can then formalize an interactive CGDDS as two algorithms given in Algorithms 3 and 4. Algorithm 3 corresponds to the propagation dynamics component and Algorithm 4 corresponds to the intervention computation component. The coordinator is embedded in Algorithm 4.

Algorithm 2

Pseudo-code of computations in the extended CGDDS and extended POMDP

```
for $t = 0$ to $T$ do
    compute interventions in $w_t = w_t^1 w_t^2 \dots w_t^{j_t}$ based on the observed system state $o(s_t),\ s_t \in S$;
    apply each $w_t^j$ function to modify vertex states or edge labels;            /* apply interventions */
    apply all $f_i$ functions to change the health states of all vertices simultaneously; /* disease spread */
```

Algorithm 3

SysDif: Propagation computation in the interactive CGDDS.

```
for $t = 0$ to $T$ do
    wait for $w_t$ from SysInt;                                       /* interventions for this time step */
    receive $w_t = w_t^1 w_t^2 \ldots w_t^{j_t}$;
    apply each $w_t^j$ function to modify vertex states or edge labels;
    apply all $f_i$ functions to change the health states of all vertices simultaneously; /* disease spread */
    send the changes in the observable system state $o(s_t)$ to SysInt;      /* newly diagnosed cases */
```

Algorithm 4

SysInt: Intervention computation in the interactive CGDDS.

```
for $t = 0$ to $T$ do
    read query $q_t \in Q$ from the user;
    compute $w_t$ by running query $q_t$ on $DB$;                    /* POMDP */
    send $w_t$ to SysDif;                     /* interventions for this time step */
    wait for updates on observable system state $o(s_t)$ SysDif;
    receive updates on $o(s_t)$ and store them in $DB$;          /* newly diagnosed cases */
```

A major difference between the interactive CGDDS and the extended CGDDS in Section 4.1 is the computation of the schedule string *W* and the scope of implementable modification functions g^V and g^E. This can be better explained by comparing our Indemics implementation and the EpiFast implementation. Specifically, Indemics can represent any intervention that can be described in EpiFast and can easily express many more interventions that are difficult to realize in EpiFast without significant code development. For example, the *household intervention*, where all vertices with one or more sick family members voluntarily take social distancing actions, is a g^E function that is difficult for EpiFast because the household data pertaining to each individual is not available within EpiFast. Even if we code this data in EpiFast, a slight change will move this intervention outside of EpiFast's g^E set, e.g. only households with a senior or a young child will take action, or only households living in a specific county will take action, or maybe household members have different thresholds — some wait until the second sick case in the family while others are more cautious — depending on age. The *targeted intervention*, where antiviral is applied to diagnosed vertices only, is a g^V function that is implemented in EpiFast—after substantial programming effort. The vertices are selected based on their current diagnosis states. If the selection is also based on another kind of state, e.g. current fear level, then EpiFast cannot handle it. All the above interventions, however, can be easily handled by Indemics implementation with simple scripting.

The interactive CGDDS decouples the primary components of an epidemic simulation described in Section 3.

Before proposing an architecture for Indemics and implementing it, we bound the communication complexity between the propagation dynamics and intervention computation components. The communication includes the following: (*i*) the extended CGDDS generates dynamic data about system state updates, which needs to be stored in the data management system; (*ii*) the extended POMDP needs to query the data management system and to obtain results; (*iii*) the extended POMDP generates interventions, which need to be sent to the extended CGDDS; and (*iv*) bookkeeping messages to keep the two modules synchronized. A proposition on the communication complexity is presented in Appendix A.5.

5. INTERVENTIONS AND SITUATION ASSESSMENT

In this section, we formalize the concept of an intervention, used for mitigating epidemic propagation. Informally, an intervention changes one or more attributes of a set of individuals. Some of the attributes correspond to behavioral changes, such as home isolation, use of a face mask, cutting down non-essential activities, etc. Other attributes correspond to disease specific changes such as immunity of an individual to a disease, level of infectiousness, infectious period duration, etc. The first type of interventions change the social contact network by adding or deleting edges, or modifying the edge labels (usually contact durations). The second type of interventions change the vertex states directly. Interventions are either a result of public policies, in which a group of individuals are simultaneously affected, or based on the perception of disease by individual members or by households. From an abstract standpoint, an intervention changes the label of a subset of vertices or edges of the interaction network. Furthermore, the functions used to compute this set depend on attributes associated with the vertices and edges of the network as well as environmental factors. The set of possible functions grows even faster and is at least double exponential in the size of the representation. As a result, it is not possible to develop a set of fixed templates that capture the range of interventions. Nevertheless, for a certain class of interventions, including many studied in practical settings, it is possible to specify the interventions using SQL. For the purposes of this paper, an intervention consists of three steps:

— Compute the set of individuals (vertices) S using a function I. I is a function of the relational data R, the dendogram until the present time t, given by D_t, and the social contact network $G(V,E)$. R represents demographic information about individuals and locations in a relational format.

— Apply an evaluation criterion that is a function F over the set S. F is usually an aggregation function. For example, F could evaluate "$|S| > 1\%$ of the population size".

— An action function H that computes the set $\mathscr{S} = \{(S_1,A_1),(S_2,A_2),\ldots\}$ where S_i is a set of individuals and A_i is an action that modifies the individuals' attributes (one or more labels of the vertices in S_i or the labels associated with edges which have one endpoint in set S_i).

Situation assessment consists of just the first two steps. It does not change the state of the system. An intervention that is applied as a result of a public policy implementation results in a set $\mathscr{S}$ where each element set S_i contains one or more elements. An individual-based intervention results in a set $\mathscr{S}$ where every element set S_i has exactly one element, corresponding to the individual, represented by the node. A household level intervention changes the labels of a set of nodes that constitute the household. Closing a school results in changing the labels of the edges that capture the interactions between students attending the school as well as the labels that correspond to their family interactions (assuming that the children stay at home). Table II lists a few examples of interventions studied in literature.

The class of functions I, F, H studied are precisely those that can be expressed using a SQL-like scripting language which we have specifically designed for representing intervention simulations. Examples of interventions written in this scripting language can be found in Algorithms 5 and 6. The computational complexity of computing these functions depends on the computational complexity of evaluating the corresponding SQL queries. This naturally places limitations on the kinds of interventions and situation assessment queries that can be answered by our method.

6. INDEMICS ARCHITECTURE AND IMPLEMENTATION

We now describe the software architecture of Indemics and its implementation. We also describe the data structures used to internally represent data within each module and the format for data interchange. The architecture builds on the interactive CGDDS framework described in Section 4.

6.1. System Architecture

As shown in Figure 1, Indemics consists of four loosely coupled modules:

- the Indemics Middleware Platform (IMP),
- the Indemics Epidemic Propagation Simulation Engine (IEPSE),
- the Indemics Intervention Simulation and Situation Assessment Engine (ISSAE) and
- the Indemics Client (IC).

These modules may be distributed across multiple heterogeneous computer systems. As an example, we may deploy the IEPSE on an HPC-cluster, the IC on a desktop, and the ISSAE on a high end database server platform. Each module can also have multiple concurrent instances.

To reduce the communication dependencies between these loosely coupled modules, distributed across the network, these modules are connected in a star-shaped architecture, where IMP is the central hub of the framework and coordinates the interaction between the remaining three components. The architecture provides better system modularity and portability. The IMP also provides mechanisms for data transformation, synchronization and concurrency control to support multiple concurrent instances of each of the modules. The following sections describe the modules of the Indemics framework in detail. The architecture described in Section 6.1 can be implemented using various approaches based on the type of simulation engine chosen to simulate the epidemic propagation process and the database management system chosen for intervention simulation and situation assessment.

6.1.1. Indemics Middleware Platform. (IMP)—The Indemics Middleware Platform is the central hub in the Indemics framework and is responsible for synchronizing and coordinating the interactions between the IC, the ISSAE and the IEPSE, in a distributed environment. All database accesses from the IEPSE or the IC go through the IMP.

To account for the differences in data formats across different modules, the IMP is responsible for appropriate data transformations to facilitate communication. Also, to make the Indemics framework independent of the specific implementations of its components and hide the implementation details of the message communication layer, as shown in Figure 2, Indemics abstracts the interactions between the IMP and other components and wraps them with a set of APIs (application programming interfaces), which are part of the IMP.

The implementation of the IMP has been designed to provide interfaces that hide low level socket communication and allow higher level abstractions for structured data to support communication with diverse modules with different data transfer and storage formats. The current implementation as shown in Figure 2 of the IMP has two sub-components: the Indemics adapter and the Indemics server.

Figure 3 shows the computation flow diagram of Indemics. As seen in the diagram, the IC, the ISSAE and the IEPSE wait for confirmation from the Indemics server before moving to the next execution step and notify the Indemics server when they finish the current processing.

This ensures that the computations are synchronized and the data remains consistent across all the modules.

The Indemics adapter has a collection of APIs that abstract the implementation details of the middleware from the other modules. The implementation of the Indemics adapter can be accomplished using any language that simplifies the interactions with the IEPSE as well as the IC. In the current Indemics system, the adapter that interfaces with EpiFast is implemented using C++, and the interface with the IC is a Java based implementation.

The Indemics server is capable of supporting multiple concurrent simulation sessions. As shown in Figure 2, each simulation session is registered with the simulation pool of the Indemics server and is managed by the session manager. The server prepares corresponding data tables for the session in the database using the database schema described above. The session manager manages appropriate data being passed from the database and the IC to the appropriate simulation session based on a session identifier. The message and data deliverer module of the Indemics server handles the data transformations for the data being passed to and from the database system.

6.1.2. The Indemics Epidemic Propagation Simulation Engine. (IEPSE)—The epidemic propagation simulation engine provides the core computations that simulate disease propagation over a dynamic social contact network. The simulation engine can be derived from existing non-interactive simulation engines, with modifications to support online interactions. The simulation engine can also be enhanced with desirable features such as rollback and check pointing, to allow users to reprocess disease propagation over a given time period, perhaps with a different set of interventions.

In our current Indemics implementation, we have chosen the extended version of EpiFast [Bisset and Chen et al. 2009] as the disease propagation simulation engine. However, as described before, any other CGDDS based simulation tool can fit in this framework. In a more complex setting, we can combine several simulation engines from multiple disciplines to study the co-evolution of multiple dynamical systems.

EpiFast is an agent-based simulation engine that simulates disease propagation in a region, represented as a social contact network. It is a concrete implementation of the CGDDS framework described in Section 4. EpiFast executes over multiple time steps using the SEIR model. Over these time steps, the agents of the simulation, which represent people in the simulated region, change their current disease state to a new disease state with probabilities defined in the EpiFast algorithm.

To support external interventions, EpiFast has been modified to include an extended CGDDS system as described in Section 4.1, with a mechanism to start and stop the disease transmission process. EpiFast stores different intervention types as vertex and edge modification functions on the social contact network, represented by g^V and g^E (defined in Section 4.1) respectively that lead to changes in the probability of infection transmission. In the Indemics system, EpiFast receives external interventions as input from the Indemics middleware in the form of the targeted subpopulation to be intervened (i.e., list of nodes to be intervened) and the intervention type to be applied (input value to the vertex or edge modification functions). For specific interventions such as vaccination, EpiFast receives intervention properties such as efficacy and compliance rates. After receiving and applying interventions, EpiFast proceeds with the propagation computation for the next time step. The output from EpiFast is a list of nodes that are infected in the current time step and this data is passed to the database through the middleware. The amount of data that is passed to and from EpiFast is very small compared to the scale of the entire simulation data. This ensures that the Indemics simulation system is scalable and efficient.

6.1.3. The Indemics Intervention Simulation and Situation Assessment Engine. (ISSAE)—As discussed earlier, we use RDBMS to support ISSAE. The current version of Indemics uses Oracle 11g relational database management system for this purpose. This database comes with built-in capabilities for error-handling, query optimization, synchronization and fault tolerance to recover in case of system failures. The data stored in the Indemics database is broadly classified into four main categories as follows: (*i*) social contact network data representing the set of proximity relationships for the given region, (*ii*) demographic data about individuals in the given region, (*iii*) temporal data about disease transmission at every time step, and (*iv*) intervention data about the intervention strategies applied.

6.1.4. The Indemics Client. (IC)—IC (also known as the user interface) provides an interface to the Indemics system. IC interfaces may have distinct implementations to match different application requirements specific to users (e.g., researchers or students in a classroom). Communication with the Indemics server is facilitated using the Indemics adapter.

Our current implementation of IC provides an interactive console interface to allow the users to input run-time instructions to query the system state, intervene the system dynamics, and control the simulation (e.g., rollback, pause, and resume). Indemics also supports a batch client, which uses a script file consisting of a set of interaction rules to automatically feed the instructions to the Indemics server. The batch script has embedded SQL statements. This provides a high level query language interface to the users for situation assessment and subpopulation selection for applying interventions. The interface also allows selecting types of intervention actions to be applied to the subpopulation such as vaccination, social distancing, antiviral prophylaxis and so on. The IMP translates the embedded SQL statements to actual SQL statements that can be applied to the RDBMS for retrieving data.

Our Indemics implementation has also been integrated with a web-based user interface called the Interface to Synthetic Information Systems (ISIS) developed by our group for setting up epidemic simulation experiments and for analyzing the simulation results graphically [Deodhar et al. 2012]. Using the web interface, the users (epidemiologists and public health policy makers) can easily set up experiments with complex interventions, without having to know anything about the HPC-based environment or the Indemics deployment. Figure 4 shows a screenshot of the ISIS interface for configuring a simulation experiment with complex interventions using the Indemics model.

6.2. Data Abstractions and Specification

In this section, we describe the internal data structures and formats for data interchange. The data representation and abstractions have been designed to ensure minimum amount of data transfer and optimal frequency of data exchange between the modules, in order to satisfy performance requirements. This also ensures conformance to Proposition A.1.

6.2.1. EpiFast—As described before, EpiFast is a concrete implementation of the extended CGDDS framework described in Section 4.1. The primary input to EpiFast is a social contact network that represents proximity relationships between individuals of the population. It is represented by the Graph $G(V,E)$, where V is a set of vertices representing the individuals of the population and E represents contacts between them. Each vertex in V has an associated vector (pid, h, t_1, t_2, l_1) where

— *pid* is the person identifier of the given vertex,

— *h* is the health state from the SEIR model,

— t_1 is the time at which the vertex is infected,

— t_2 is the time of recovery and

— l_1 is the list of interventions applied on the vertex.

Each edge in E has an associated vector (V_1,V_2,p) where

— V_1 and V_2 are the vertices on which the edge is incident and

— p is the probability of transmission between the vertices as defined in the EpiFast algorithm.

EpiFast reads the entire graph $G(V,E)$ into the main memory from a flat file at the beginning of the simulation. $G(V,E)$ remains unchanged throughout the simulation. Interventions change the edge attributes and can simulate edge deletion by using appropriate edge label.

The other input to EpiFast is a list of interventions selected by the user from the ISSAE. For implementation, interventions are represented as I = (pid,**A**), where

— *pid* represents the identifier of the person to be intervened and

— **A** represents the intervention action to be implemented. **A** is a vector given by (*type, del, eff, dur, compl*), where

  — *type* is the type of the intervention to be applied such as vaccination, social distancing, antiviral,

  — *del* represents the delay in implementing the intervention action in real world, and

  — *dur*, *eff* and *compl* represent the duration, efficacy and compliance rate respectively of the intervention action applied on the targeted population.

After the intervention *I* is obtained from Indemics at every time step, EpiFast applies the algorithm as described in [Bisset and Chen et al. 2009] and computes disease propagation to generate a list of individuals infected in the next time step.

The output from EpiFast is a vector **O** of the form (*infected, infector, infDur, diagnosed*) where

- *infected* represents the set of newly infected vertices in the current time step,
- *infector* represents the corresponding vertices that infected them,
- *infDur* is the duration for which the vertex would remain in the infectious health state, and
- *diagnosed* are the vertices that are diagnosed.

The output from EpiFast is passed to ISSAE through the IMP at every time step.

6.2.2. RDBMS-based Situation Assessment and Intervention Simulation engine—As described before, ISSAE is based on RDBMS; it stores and processes the four kinds of datasets: the social contact network data *N*, demographic data *R*, infection dendogram data *D* and intervention data *I*. Demographic data for each region is stored in a simple relational format in a table given by the tuple *R* = (*pid, age, gender, income*). *R* is static and remains unchanged for the duration of a simulation. New demographic value sets can be added to the tuple based on availability of information for the population.

The social contact network data *N* is stored as a tuple $N = (pid_1, pid_2)$, where pid_1 and pid_2 represent the end points of an edge in the social contact network. This is a copy of the data used by EpiFast to simulate epidemic propagation. It is stored in the RDBMS so that interventions based on social contact network structure can be formulated.

The temporal data *D* related to infections is stored in a separate table which can be directly updated based on the output received from EpiFast. This data can be represented by the tuple (*infected, infector, infDur, diagnosed*) described above.

ISSAE is used to support situation assessment and intervention simulation. Section 5 has already described this in detail. The output obtained from ISSAE is given by $\mathscr{S} = \{(S_1,A_1), (S_2,A_2),\ldots\}$ where S_i contains person identifiers on whom to apply interventions and A_i represents the intervention actions. IMP sends this to EpiFast. When EpiFast resumes computation, it uses the new vertex and edge labels to evaluate the epidemic propagation for the next time step. For instance, if it is observed that more than a pre-defined threshold of school age individuals are infected in the current time step, then it would be appropriate to apply an intervention action, such as vaccination, to the school age population in the next time step.

7. PERFORMANCE EVALUATION

In this section, we present the experimental results on the performance of our Indemics implementation. We designed and performed a series of experiments simulating disease propagation in several U.S. metropolitan areas with different complex intervention strategies to show the performance of Indemics in realistic applications.

In our experiments, the IEPSE in Indemics is EpiFast, a parallel code and it runs on a shared-nothing Linux cluster. The cluster consists of 96 compute nodes, each having two Intel Quad Core Xeon E5440 processors running at 2.83 GHz and 16 GB memory (2 GB/core). We note that the simulations do not use all the CPU's in the cluster. The RDBMS in the Indemics framework is an Oracle Database 11g running on a separate server with 16 CPU cores and 64 GB memory. The Linux cluster and the database server are located in a campus-wide computing center. The Indemics server runs on the head node of the cluster.

In our epidemic simulations, the epidemic evolution depends on the contact network including its structure, contact durations, and interventions that change the network, as well as parameters of the SEIR disease model. In Appendix we explain the SEIR model and disease propagation in a network with or without interventions. The disease model parameters include the transmission coefficient, the incubation period, and the infectious period. In Appendix A.4, we explain these parameters and how they are related to the basic reproductive number R_0 (see Appendix A.4 for the R_0 definition). We emphasize, however, **our work is not about epidemic modeling**; instead, **it provides a modeling environment to support epidemic simulations**. Therefore Indemics does not pertain to any specific disease model. Although we ensure that the disease parameters used in our case studies are consistent with published epidemiological parameters [Halloran et al. 2008; Germann et al. 2006; Wu and Cowling 2011; Fraser et al. 2009], our experimental results regarding the capability and efficiency of Indemics are not limited to the specific disease parameters chosen in our simulations.

7.1. The Performance Overhead of Indemics

To integrate it with the Indemics framework, we have extended EpiFast by adding the stop-and-resume feature to allow synchronized data and message communication between the simulation and the Indemics server. This introduces several additional I/O operations for EpiFast: reading the information of intervention actions from the RDBMS via Indemics server before each simulation step and writing current system states to the RDBMS via the Indemics server after each simulation step. These communications create performance overhead. In our Indemics implementation, we have taken care to minimize the performance overhead. For instance, only the ID's of the individuals whose health states change during the current time step are sent to the database and only the ID's of the intervened individuals are sent to the simulation engine.

We evaluate the performance overhead by running both EpiFast (standalone) and Indemics (with the extended version of EpiFast) to simulate the same intervention scenarios. The epidemic dynamics and the computations in the simulations are exactly the same between EpiFast and Indemics. Figure 5 shows the ratio of the execution time of each simulation time step (day) using these two simulation systems for the Miami population with the scenario in Algorithm 5.

Algorithm 5

Intervention scenario: vaccinate preschoolers when over 1% of them are sick.

```
CREATE TABLE Preschool (pid) AS (SELECT pid FROM Person WHERE $0 \leq$ age $\leq 4$);    /* based on
demographic data */
DEFINE nPreschool AS (SELECT COUNT(pid) FROM Preschool);
for day from 1 to 300 do
    /* based on demographic and disease dynamic data                                    */
    WITH Infected_Preschool (pid) AS (SELECT pid FROM Preschool AND Infected_Person WHERE
    Preschool.pid = Infected_Person.pid)
    DEFINE nInfectedPreschool AS (SELECT COUNT(pid) FROM Infected_Preschool);
    /* check triggering condition                                                       */
    if nInfectedPreschool $>$ $1\% \times$ nPreschool then
        apply vaccines to (SELECT pid FROM Preschool) ;    /* intervention subpopulation and
        action */
```

Figure 5 indicates that Indemics incurs little overhead in the early period (day 0 to 60) and died-out period (after day 220) but incurs large overhead during the outbreak period (day 61 to day 119). On day 120, when the intervention is triggered and applied, the overhead reaches the maximum. This observation is within our expectation: the overhead of Indemics is roughly proportional to the data size communicated between the simulation engine and the database. When more people are infected, we observe larger overhead as their ID's need to be sent to the database. On day 120 when the 1% threshold is exceeded, transferring the intervened preschool subpopulation data from the database to the simulation engine causes additional overhead. We note that the total execution time of Indemics is only 70% greater than that of EpiFast.

7.2. The Performance of Indemics in Complex Interventions

The strength of Indemics is its capability and flexibility in simulating complex adaptive intervention scenarios. We evaluated the Indemics performance for a series of complicated interventions that are not available and difficult to implement within EpiFast. These intervention scenarios reinforce the belief that Indemics should be deployed for studies with new complex interventions in a short turnaround time instead of using the conventional HPC-based simulation engine like EpiFast that may give slightly better performance but occur a long development time.

Interventions—The intervention strategies in this experiment are household intervention and targeted intervention. In the **household intervention** strategy, all members of households with one or more diagnosed household members take social distancing actions. The **targeted intervention** is to treat people with a certain health state immediately, e.g., administer antiviral drugs to sick school-age children when they are diagnosed.

We chose these two intervention strategies because they are very distinct in several characteristics.

1. The targeted intervention is a public health level policy intervention; the household intervention is an individual/household level behavioral self protection reaction.
2. From the perspective of the information scope used in the intervention simulation, both interventions need the health state information of each individual, which is already available in the simulation engine. In addition, the targeted intervention needs age of each individual; the household intervention needs the household membership data; both are stored in the database.
3. From the perspective of computational complexity for generating the intervention, the household intervention is more complex, because it requires two RDBMS table-join operations for choosing who will be intervened, while the targeted intervention needs only one.
4. Finally, from the perspective of intervention effectiveness, the household intervention in our experiments is more effective in containing the epidemic, because it is a proactive strategy while the targeted intervention is reactive.

In summary, these two interventions are representative ones in various aspects.

Populations—These two interventions are simulated in three U.S. urban areas: Miami, Chicago and Los Angeles (LA). They are chosen because of their diversity in geographic location, population size, age distribution, as well as household income and size distributions. Some statistics of these populations and the number of processors used for the disease propagation simulations (EpiFast) are given in Table III. Since the population of Los Angeles is eight times larger than Miami and twice the size of Chicago, we use twice the number of processors for the Los Angeles simulation for the speed-up.

Each intervention in these three regions has been simulated 20 times with identical disease configurations but different infection seeds, and the average running time is logged. The epidemic simulations start on day zero with initial health conditions for each person, and terminate on the 300th day. The total execution time for these 300 simulation days is given in Table IV.

From Table IV, we observe that the cost of database query (DB) is much more dominant in the household intervention scenario than in the targeted intervention scenario. The major reason is that the household intervention is computationally more complex than the targeted intervention. Although the database query looks expensive, it is still meaningful to utilize the RDBMS. First, using a RDBMS and its query language avoids re-engineering work on the simulation engine for new intervention scenarios, which usually takes several weeks of development efforts [Ma et al. 2011]. Second, the RDBMS in our experiments is just a commodity RDBMS; if we incorporate a cluster-based DBMS [Simmhan et al. 2009], then the performance of Indemics could be significantly improved.

In addition to the total execution time, we also present the detailed execution time for each simulation day. The daily execution time for the disease propagation simulation, the database query, and the communication are shown in Figures 6, 7, and 8 respectively. Because the quantity of intervened individuals generally grows with the infected cases, the communication time to exchange the daily health information and the intervention subpopulation changes in the same direction as the quantity of the daily new infection cases (Figure 8); the simulation time to implement the vertex state and edge label changes also changes in the same direction (Figure 6). The database query time in the intervention simulations is remarkably distinct, because the complexity of the intervention simulation depends on the daily infection cases, as well as many other factors, such as the query complexity and the size of the static data (e.g., size of the household table).

7.3. Scalability

We carried out an experiment to assess the scalability of Indemics as a function of the social contact network size. We selected eight U.S. areas with increasing population sizes: Miami, Seattle, Boston, Dallas, Indiana (state), Virginia (state), New Jersey (state), and Chicago. Their social contact networks have similar average degrees. The intervention applied in the experiment is the *household pharmaceutical intervention (PI)*, where all household members of any household are vaccinated immediately after one household member is diagnosed. It is similar to the household intervention described in earlier subsections which involves social distancing actions. In all simulations, we used 40 processors (five compute nodes and eight processors per node) of the aforementioned cluster and the same Oracle database. We ran 10 replicates for each area. The eight areas are summarized in Table V.

We plot the execution times against the population size in Figure 10. Each data point in the plot shows the mean value and one standard deviation below and above the mean. The running time seems to have small variances across replicates. The complexity of the database queries for computing the individuals to be intervened increases with population size in general, but also depends on the disease prevalence and the average household size in the population.

We find that the total execution time for a simulation with such an intervention varies from 2 minutes to about 10 minutes. This shows that Indemics scales well to large populations with millions of people. For a complete case study for a large urban region or a state with a factorial design of e.g. two diagnosis rates (probability of sick people being identified) by two intervention actions (vaccine or antiviral) by two compliance rates (fraction of people following the intervention policy), and 20 replicates per setting, it takes only about 6 hours to 1.5 days to run the whole study using a small amount of computing resources.

8. CASE STUDY

There are two important motivations to employ Indemics for complicated epidemic intervention case studies. First, Indemics manages supplemental data for the case studies separately from the simulation engine. Hence it is not required to modify the simulation engine code to accommodate new intervention scenarios. Second, the intervention scenarios are written in a SQL-like scripting language specifically designed for intervention simulations; see Algorithms 5 and 6 for two examples. Scripting in this language usually takes much less time than coding for the simulation engine.

We have used Indemics to assist public health decision makers and economists interested in assessing the efficacy of intervention strategies to control pandemics. In this section, we describe a case study where Indemics was used to compare complex intervention strategies for epidemic control and prevention.

8.1. Scenario of Case Study

The case study aimed at evaluating three intervention strategies for containing a potential influenza outbreak in a region. The case study has been published in [Marathe et al. 2011]. Here, we discuss how Indemics improved human productivity and led to the completion of the study in a timely manner.

1. **School Intervention**. In any school, if the fraction of students diagnosed with flu exceeds a given threshold, say 5%, then apply vaccination or antiviral to all students in this school.

2. **Block Intervention**. In any census block, if more than a given fraction of the people living in this block are diagnosed with flu, then apply vaccination or antiviral to all people living in this block.

3. **Distance-1 Intervention**. It is based on individual decision making. If a person observes that more than a threshold fraction of his/her social contacts are diagnosed with flu, then he/she will take vaccine or antiviral.

The purpose of this study is to examine the effectiveness and social cost of the three interventions as a function of influenza disease model parameters. In our experiment design, we vary the disease infectivity (R_0), diagnosis rate, and intervention compliance rate. Two types of pharmaceutical interventions, vaccination and antiviral, are considered. The trigger threshold of interventions are set as 1% and 5%. Totally, we have 160 different experiment cases.

8.2. Study Implementation and Observations

We use the school intervention to demonstrate the implementation of an intervention in Indemics. The similar approach applies to other interventions, with different Indemics scripts. There are two steps to implement an Indemics intervention: data preparation and script development. First we create a table Student School in the database for the student-school relation. Second we specify the school intervention using the intervention scripting language, as illustrated in Algorithm 6. Then we run it together with the propagation simulation.

By using Indemics, it only took us one day to implement the three intervention strategies in Indemics scripts. We ran 4000 simulation replicates (160 different cases and 25 iterations per case) in three days. Without Indemics, adding code to represent the three interventions in EpiFast would have taken us weeks. The main reasons for this are the following: (*i*) EpiFast needs access to the data necessary for mapping students to schools or individuals to census blocks; codes would be needed to read this data and create an appropriate data structure, (*ii*) codes would be needed to count the diagnosed people in each school, block, or in the first neighborhood (measured using the social contact network) of every individual using a parallel algorithm, since the individuals are distributed across processors in the current code; (*iii*) identify all individuals in the schools or blocks that need to be intervened and this needs to be done using a parallel code too. Implementing the distance-1 intervention is relatively easier than the other two interventions; but all of them need extra care for parallel computing and to ensure correctness. It would take substantial recoding should the policy analyst choose workplace instead of schools to vaccinate individuals. This change is trivial to implement when using SQL.

We summarize some of the case study results. Further results of the study are reported in [Marathe et al. 2011]. In Figures 11 and 12 we plot the daily infections count and number of people selected for interventions. Ideally one would like to see an intervention where both the disease prevalence (solid line) and the cost of intervention (dotted line) are low. In the vaccination case, none of the three interventions show this. The school intervention uses a small amount of vaccines but has a high peak in the epidemic curve. The block intervention has a small epidemic but uses too many vaccines. The distance-1 intervention has neither a low cost nor a low disease prevalence. In the antiviral case, the distance-1 intervention clearly is the optimal intervention strategy with the lowest epidemic and the lowest number of antiviral courses used.

Algorithm 6

School Intervention: vaccinate all students in any school where over 1% of the students in this school are sick.

```
for replicate from 1 to 25 do
    CREATE TABLE School_Diagnosed_Today (school, diagnosed, day) ;        /* number of students
    newly diagnosed */
    CREATE TABLE School_Diagnosed_Total (school, diagnosed, day) ;        /* number of students
    still sick */
    for day $\alpha$ from 1 to 300 do
        /* count number of newly diagnosed students in each school                  */
        INSERT (SELECT school, COUNT(pid), $\alpha$ FROM Student_School, Daily_Diagnosed WHERE
        Student_School.pid = Daily_Diagnosed.pid GROUP BY school) INTO School_Diagnosed_Today;
        /* count number of students still sick in each school                       */
        INSERT (SELECT school, SUM(diagnosed), $\alpha$ FROM School_Diagnosed_Today WHERE $\alpha - 7 <$ day
        $<= \alpha$) INTO School_Diagnosed_Total;
        /* find schools where over 1% students are sick to be intervened            */
        WITH Infected_School (school) AS (SELECT school FROM School_Diagnosed_Total,
        School_Intervened WHERE diagnosed / school-size $> 0.01$ and day = $\alpha$)
        SET intervened_day = $\alpha$ in School_Intervened WHERE intervened_day = NA and school $\in$
        Infected_School;
        /* find students in these schools and apply vaccines to them                */
        WITH Targeted_School (school) AS (SELECT school FROM School_Intervened WHERE intervened_day
        = $\alpha$)
        APPLY vaccination treatment on (SELECT pid FROM Targeted_School, Student_School WHERE
        Targeted_School.school = Student_School.school);
```

9. CONCLUSIONS AND FUTURE WORK

We presented a HPC-based service architecture to support epidemic modeling. A key feature of the architecture is to decouple the three components of the system: disease progression simulation, situation assessment, and intervention simulation. The disease simulation is a compute intensive component and is suitable for implementation on a traditional HPC-cluster. The latter two components are data intensive and we investigated the use of RDBMS to support them. The decoupling results in additional computational time due to the increased communication complexity and computing the interventions using RDBMS. But this is better than programming the interventions and results in a significant decrease in the overall time of completing a study. In addition, the decoupling allowed the system to be more easily accessible to public health analysts who were not computing experts. Indemics is a prototype implementation that realizes the architecture. A simple user interface allows the user to interact with Indemics.

Indemics can be improved in a number of ways. First, as discussed earlier, the class of interventions that are currently supported are based on the state of the system from the beginning to the current time. We plan to investigate extensions that will support interventions that are based on one or more possible future system states. Second, interventions that are based on complicated measures of the graphical structure around a given individual are extremely expensive since the network is currently stored in a RDBMS. We are currently investigating the use of heterogeneous database technologies to support the diverse forms of data. Finally, the user interface needs to be further extended and combined with simple programming language support to specify interventions. Our assessment is that a simple domain specific language might suffice for this purpose. A program written in this language can then be translated into a series of SQL statements to represent complex interventions.

Acknowledgments

We thank the reviewers and TOMACS editors for their extensive comments and suggestions that have greatly improved the manuscript. We also thank members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their helpful suggestions and comments. This work has been partially supported by NSF HSD Grant SES-0729441, NSF PetaApps Grant OCI-0904844, NSF NetSE Grant CNS-1011769, NSF SDCI Grant OCI-1032677, DTRA Grant HDTRA1-11-1-0016, DTRA CNIMS Contract HDTRA1-11-D-0016-0001 NIH MIDAS Grant 2U01GM070694-09.

REFERENCES

Ajelli M, Gonçalves B, Balcan D, Colizza V, Hu H, Ramasco J, Merler S, Vespignani A. Comparing Large-Scale Computational Approaches to Epidemic Modeling: Agent-Based Versus Structured Metapopulation Models. BMC Infectious Diseases. 2010; 10:190. (2010). [PubMed: 20587041]

Anderson, RM.; May, RM. Infectious Diseases of Humans. Oxford: Oxford University Press; 1991.

Bailey, N. The Mathematical Theory of Infectious Diseases and Its Applications. New York: Hafner Press; 1975.

Barrat, A.; Barthelemy, M.; Vespignani, A. Dynamical processes in complex networks. Cambridge University Press; 2008.

Barrett, CL.; Bisset, K.; Chen, J.; Eubank, S.; Lewis, B.; Kumar, VSA.; Marathe, MV.; Mortveit, HS. Interactions among human behavior, social networks, and societal infrastructures: A Case Study in Computational Epidemiology. In: Ravi, SS.; Shukla, Sandeep K., editors. Fundamental Problems in

Computing: Essays in Honor of Professor Daniel J. Rosenkrantz, Chapter 18. Springer Netherlands: 2009. p. 477-507.

Barrett, CL.; Bisset, KR.; Eubank, SG.; Feng, X.; Marathe, MV. EpiSimdemics: an Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks. Proc. ACM/IEEE conference on Supercomputing; Austin, Texas, USA. 2008. p. 290-294.

Bisset, K.; Chen, J.; Kuhlman, CJ.; Kumar, VSA.; Marathe, MV. Interaction-Based HPCModeling of Social, Biological, and Economic Contagions Over Large Networks. Proceedings of Winter Simulation Conference (WSC); 2011. p. 2933-2947.

Bisset, KR.; Chen, J.; Feng, X.; Kumar, VSA.; Marathe, MV. EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. Proceedings of the 23rd International Conference on Supercomputing; 2009. p. 430-439.

Bisset, KR.; Feng, X.; Marathe, MV.; Yardi, SM. Modeling Interaction between Individuals, Social Networks and Public Policy to Support Public Health Epidemiology. Winter Simulation Conference; 2009. p. 2020-2031.

CDC. [accessed on September 29, 2011] Updated CDC Estimates of 2009 H1N1 Influenza Cases, Hospitalizations and Deaths in the United States, April 2009 - April 10, 2010. 2010. http://www.cdc.gov/h1n1flu/estimates_2009_h1n1.htm (2010)

Chao DL, Halloran ME, Obenchain V, Longini IM Jr. FluTE, a publicly available stochastic influenza epidemic simulation model. PLoS Computational Biology. 2010; 6:1. (2010).

Colizza V, Barrat A, Barthelemy M, Valleron A, Vespignani A. Modeling the worldwide spread of pandemic influenza: baseline case and containment interventions. PLoS Medicine. 2007; 4:95. (2007).

Colizza V, Barrat A, Barthelemy M, Vespignani A. Prediction and predictability of global epidemics: the role of the airline transportation network. Arxiv preprint q-bio.OT/0507029. 2005 (2005).

Deodhar, S.; Bisset, K.; Chen, J.; Ma, Y.; Marathe, MV. Enhancing Software Capability through integration of distinct software in epidemiological systems. Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium; 2012. p. 171-180.

Dobbs, R.; Smit, S.; Remes, J.; Manyika, J.; Roxburgh, C.; Restrepo, A. Urban world: Mapping the economic power of cities, McKinsey Global Institute. 2011. http://www.mckinsey.com/insights/mgi/research/urbanization/urban_world. (2011)

Edlund, SB.; Davis, MA.; Kaufman, JH. The spatiotemporal epidemiological modeler. In. Proceedings of the 1st ACM International Health Informatics Symposium; 2010. p. 817-820.

Eubank, SG. ACM Symposium on Applied Computing. Spain: Madrid; 2002. Scalable, efficient epidemiological simulation; p. 139-145.

Ferguson NM, Cummings DAT, Cauchemez S, Fraser C, Riley S, Meeyai1 A, Iamsirithaworn S, Burke DS. Strategies for containing an emerging influenza pandemic in Southeast Asia. Nature. 2005; 437:209–214. (2005). [PubMed: 16079797]

Ferguson NM, Cummings DAT, Fraser C, Cajka JC, Cooley PC, Burke DS. Strategies for mitigating an influenza pandemic. Nature. 2006; 442:448–452. (2006). [PubMed: 16642006]

Ferguson NM, Keeling MJ, Edmunds WJ, Gani R, Grenfell BT, Anderson RM, Leach S. Planning for smallpox outbreaks. Nature. 2003; 425:681–685. (2003). [PubMed: 14562094]

Fineberg HV, Wilson ME. Epidemic Science in Real Time. Science. 2009 May.324:987. (2009). [PubMed: 19460968]

Fraser C, Donnelly CA, Cauchemez S, Hanage WP, et al. Pandemic Potential of a Strain of Influenza A (H1N1): Early Findings. Science. 2009; 324:1557–1561. 5934 (2009). [PubMed: 19433588]

Germann TC, Kadau K, Longini IM, Macken CA. Mitigation strategies for pandemic influenza in the United States. Proceedings of the National Academy of Sciences. 2006; 103:5935–5940. 15 (2006).

Goldstein E, Paur K, Fraser C, Kenah E, Wallinga J, Lipsitch M. Reproductive numbers, epidemic spread and control in a community of households. Mathematical Biosciences. 2009; 221:11–25. 1, (2009). [PubMed: 19559715]

Halloran E, Ferguson NM, Eubank S, Longini IM Jr, Cummings DAT, Lewis B, Xu S, Fraser C, Vullikanti A, Germann TC, Wagener D, Beckman R, Kadau K, Barrett C, Macken CA, Burke DS, Cooley P. Modeling targeted layered containment of an influenza pandemic in the United States. Proceedings of the National Academy of Sciences. 2008; 105:4639–4644. (2008).

Heber, G.; Gray, J. CoRR abs/cs/0701159. 2007a. Supporting Finite Element Analysis with a Relational Database Backend, Part I: There is Life beyond Files. (2007)

Heber, G.; Gray, J. CoRR abs/cs/0701160. 2007b. Supporting Finite Element Analysis with a Relational Database Backend, Part II: Database Design and Access. (2007)

Heesterbeek JAP. A Brief History of R0 and a Recipe for its Calculation. Acta Biotheoretica. 2002; 50:189–204. 3 (2002). [PubMed: 12211331]

Heffernan JM, Smith RJ, Wahl LM. Perspectives on the basic reproductive ratio. Journal of the Royal Society Interface. 2005; 2:281–293. 4 (2005).

Hethcote HW. The Mathematics of Infectious Diseases. SIAM Rev. 2000 Dec.42:599–653. 4 (2000).

Hufnagel L, Brockmann D, Geisel T. Forecast and control of epidemics in a globalized world. Proceedings of the National Academy of Sciences. 2004; 101:15124–15129. (2004).

Keeling MJ, Eames KTD. Networks and epidemic models. J. R. Soc. Interface. 2005; 2:295. (2005). [PubMed: 16849187]

Van Kerkhove M, Ferguson N. Epidemic and intervention modelling —a scientific rationale for policy decisions? Lessons from the 2009 influenza pandemic. Bull World Health Organ. 2012:306–310. (2012). [PubMed: 22511828]

Kermack WO, McKendrick AG. A Contribution to the Mathematical Theory of Epidemics. Proceedings of the Royal Society of London. Series A. 1927; 115:700–721. 772 (1927).

Liao, S.; Ma, Y.; Chen, J.; Marathe, A. Paid Sick-Leave: Is It a Good Way to Control Epidemics?. the 2nd International Conference on Complex Sciences: Theory and Applications (COMPLEX); 2012.

Lipsitch M, Finelli L, Heffernan RT, Leung GM, Redd SC. Improving the evidence base for decision making during a pandemic: The example of 2009 Influenza A H1N1. Biosecur Bioterror. 2011; 9:89–115. 2 (2011). [PubMed: 21612363]

Ma, Y.; Bisset, KR.; Chen, J.; Deodhar, S.; Marathe, MV. Efficient implmentation of complex interventions in large scale epidemic simulations. Proceedings of the Winter Simulation Conference; 2011.

Marathe A, Lewis B, Barrett C, Chen J, Marathe M, Eubank S, Ma Y. Comparing Effectiveness of Top-Down and Bottom-Up Strategies in Containing Influenza. PLoS ONE. 2011 Sep.6:e25149. 9 (2011). [PubMed: 21966439]

Merler S, Ajelli M. The role of population heterogeneity and human mobility in the spread of pandemic influenza. Processings of Royal Society. 2010; 277:557–565. 1681 (2010).

Meyers LA. Contact network epidemiology: Bond percolation applied to infectious disease prediction and control. Bulletin of The American Mathematical Society. 2007; 44:63–86. (2007).

Meyers, LA.; Dimitrov, N. INFORMS, Tutorials in Operations Research. 2010. Mathematical Approaches to Infectious Disease Prediction and Control. (2010)

Meyers LA, Newman MEJ, Pourbohloul B. Predicting epidemics on directed contact networks. Journal of Theoretical Biology. 2006; 240:400–418. 3 (2006). [PubMed: 16300796]

Mundhenk M, Goldsmith J, Lusena C, Allender E. Complexity of finite-horizon Markov decision process problems. J. ACM. 2000; 47:681–720. 4 (2000).

Newman M, Jensen I, Ziff RM. Percolation and epidemics in a two-dimensional small world. Physical Review E. 2002; 65:021904. (2002).

Parker J, Epstein JM. A Distributed Platform for Global-Scale Agent-Based Models of Disease Transmission. ACM Transactions on Modeling and Computer Simulation. 2012; 22 1 (2012).

Pastor-Satorras, R.; Vespignani, A. Epidemics and immunization in scale-free networks. In: Bornholdt, S.; Schuster, HG., editors. Handbook of Graphs and Networks. Berlin: Wiley-VCH; 2002.

Perumalla, KS.; Seal, SK. SIMULATION. 2011. Discrete event modeling and massively parallel execution of epidemic outbreak phenomena. (2011)

Rvachev LA, Longini IM. A mathematical model for the global spread of influenza. Mathematical Biosciences. 1985; 17:3–22. (1985).

Simmhan, Y.; Barga, R.; van Ingen, C.; Nieto-Santisteban, M.; Dobos, L.; Li, N.; Shipway, M.; Szalay, AS.; Werner, S.; Heasley, J. HICSS. 2009. GrayWulf: Scalable Software Architecture for Data Intensive Computing; p. 1-10.

Sirakoulis, G Ch; Karafyllidis, I.; Thanailakis, A. A cellular automaton model for the effects of population movement and vaccination on epidemic propagation. Ecological Modelling. 2000; 133:209–223. 3 (2000).

Sowell, B.; Demers, A.; Gehrke, J.; Gupta, N.; Li, H.; White, W. CIDR. 2009. From Declarative Languages to Declarative Processing in Computer Games.

Vynnycky, E.; White, RG. An Introduction to Infectious Disease Modelling. Oxford University Press; 2010.

Wang G, Salles M, Sowell B, Wang X, Cao T, Demers A, Gehrke J, White W. Behavioral Simulations in MapReduce. Proceedings of the VLDB Endowment. 2010; 3:952–963. 1 (2010).

White SH, del Rey AM, Sánchez GR. Modeling epidemics using cellular automata. Appl. Math. Comput. 2007a; 186:193–202. 1 (2007).

White, WM.; Demers, AJ.; Koch, C.; Gehrke, J.; Rajagopalan, R. Scaling games to epic proportion. SIGMOD Conference; 2007b. p. 31-42.

WHO. WHO consultation on priority public health interventions before and during an influenza pandemic. 2004. Available at http://www.who.int/csr/disease/avian_influenza/consultation/en/ (2004)

Wu JT, Cowling BJ. The use of mathematical models to inform influenza pandemic preparedness and response. Experimental Biology and Medicine. 2011; 236:955–961. 8 (2011). [PubMed: 21727183]

APPENDIX

In this section we describe details about the models used in epidemic simulations in this work. More details can be found in e.g. [Barrett et al. 2008; Bisset and Chen et al. 2009].

A.1. SEIR model for within host disease progression

In our epidemic simulations, transitions of the health state within each vertex follow the classic SEIR model. Each vertex is in one of the following four health states at any time: *susceptible*, *exposed*, *infectious*, and *removed*. A vertex starts in the susceptible state and remains so unless he has contacts with one or more infectious vertices, in which case he probabilistically becomes exposed (infected). If a vertex v becomes exposed, he remains so for $\Delta t_E(v)$ days (called latent phase or incubation period), during which he is not infectious. Then he becomes infectious and remains so for $\Delta t_I(v)$ days, during which he can spread the disease to his contacts probabilistically. Finally he becomes removed (or recovered) and remains so permanently. Figure 13 shows state transitions in the SEIR model.

A.2. Contact network

A social contact network $G(V,E)$ is a directed, edge-labeled network. Vertices correspond to individuals in a population, edges represent the contacts between pairs of vertices. Each edge has a weight label that is the duration of the contact and a type label that is the contact type (home, work, school, shopping, or others). Edge (u,v) with weight $w(u,v)$ represents that vertex u has a contact of duration (in units of time) $w(u,v)$ with vertex v each day, during which an influenza like illness may transmit from vertex u to vertex v with probability $p(u,v)$. Probability $p(u,v)$ depends on the disease infectivity as well as the states of two vertices and the contact between them. In Figure 14 we show three possible trajectories (out of many) of disease propagation in a small contact network. The figure shows the randomness of epidemic dynamics, as well as how interventions may change the network and affect the epidemic.

The contact network represents five co-workers in a workplace (office for instance) and they have contacts with each other during working hours every day. For simplicity we assume the disease has zero incubation days; so there are only three states: susceptible (green), infectious (red), and removed (white). Although generally different vertices have heterogeneous infectious durations, in this example we assume that the infectious period lasts exactly one day for each infected vertex. Suppose the epidemic starts with vertex A being infectious ($t = 1$).

In scenario 1 ⇒ 1.1, A infects B and D, so on the second day ($t = 2$) B and D are infectious. The edges along which disease transmits are marked with arrows. Both B and D infect vertex C so at $t = 3$ C becomes infectious. Finally C infects E so all five vertices are infected in this scenario.

In scenario 1 ⇒ 1.2, after C is infected he takes sick leave. This removes the contacts between C and other vertices in the network (edges removed in $t = 4$ network). In this scenario C's intervention action prevents disease transmission to vertex E.

Scenario 2 is another trajectory of the epidemic, where A infects only vertex B. B infects D and E. Observing his coworkers' symptoms, vertex C takes extra caution at work, e.g. washing hands often, opening windows for ventilation, and taking antiviral drugs. These measures do not avoid contacts but reduce the probability of disease transmission through the contacts—shown with edges being dotted lines. In this scenario vertex C is not infected.

We note that there are many more possible scenarios and outcomes of the epidemic dynamics in this network than can be shown in the figure. What really happens in an epidemic is one random instance whose distribution depends on interventions both applied at public health level and adopted at individual level.

A.3. SEIR model for between host disease propagation

With the SEIR model, the disease spreads in a population in the following way. On any day, if any vertex u is in the infectious state, and he has contact with vertex v, and v is in the susceptible state on that day, then the probability of the disease transmission from u to v on this day is:

$$p(u,v)=1-(1-r_u^o r_v^i)^{w(u,v)},$$

where $w(u,v)$ is the weight of edge (u,v); r_u^o is the probability of vertex u infecting any other vertex in one unit of time of contact; r_v^i is the probability of vertex v getting infected by any other vertex in one unit of time of contact; these two variables depend on the demographics of the two vertices, as well as their immunity. So the disease propagates probabilistically along the edges of the contact network. In Figure 14, we show three possible trajectories of disease propagation in a small contact network.

A crucial assumption made in almost all epidemic models is that of *independence*: we assume that the spread of infection from a vertex u to vertex v is completely independent of the infection from a vertex u' to vertex v. Similarly, an infected vertex u spreads the infection to each neighbor v, independent of the other neighbors of u. This is a central assumption in almost all the epidemic models and the analytical results based on percolation.

A.4. Parameters in our SEIR disease model

In epidemiology the *basic reproductive number* R_0 has been a key parameter and widely used in the study of infectious disease. In general R_0 is defined as the expected number of secondary cases one case generates over the course of its infectious period [Fraser et al. 2009]. In contact network epidemiology where complete mixing is not assumed, this parameter obviously depends on the network structure, as well as the durations that infected vertices remain infectious which may be heterogeneous among vertices. There are various ways of formulating and estimating R_0 from epidemic data, see [Heesterbeek 2002; Heffernan et al. 2005; Meyers 2007; Goldstein et al. 2009] for details.

In our epidemic simulations, instead of R_0, we specify the following parameters related to the disease: transmission coefficient β which is the probability of transmission per unit time of contact, distribution of incubation period, and distribution of infectious period. For each vertex we sample its incubation period from the incubation period distribution, and likewise for its infectious period. Let $\overline{\Delta t_I}$ be the average infectious period (in days) among vertices. Given these parameters and the contact network, the basic reproductive number R_0 can be approximated as $\overline{d}\beta\overline{w}\overline{\Delta t_I}$, where $\bar{d}$ is the average degree of the network and $\bar{w}$ is the average daily contact duration (in units of time) between two vertices.

A.5. Proposition on communication between propagation dynamics and interventions

Suppose that the query results for the interventions consist of either values of aggregate functions or subsets of vertices chosen for the interventions (e.g. school age individuals who are sick). This usually holds in practice. We have the following proposition.

Proposition A.1. *The communications between the propagation dynamics component and intervention computation component of the interactive CGDDS in each time step are at most* $O(k|V| + |V||g^V| + |E||g^E|)$.

Proof. Consider the communications: (i) in the worst case each vertex changes all its states and the data that needs to be communicated is $O(k|V|)$ where k is the number of states (see Section 4.1); (ii) the query results are $O(|V|)$; (iii) in the worst case different interventions are generated for each vertex and each edge and the data that needs to be communicated is $O(|V||$

$g^V| + |E||g^E|$); (4) the total amount of data in the control messages used for synchronizing the components is $O(1)$.

In fact, an epidemic simulation usually involves only health state updates in the CGDDS, and a constant number of possible vertex or edge modification functions. So in each time step, for (*i*) the data is only $O(|V|)$; for (*iii*) the data is only $O(|V| + |E|)$. Furthermore, in most common scenarios the total data communication is $O(|V|)$ in the whole simulation. For example, with the SEIR model (see Appendix A.1) each individual changes his/her health state only for $O(1)$ number of times during the whole simulation; epidemic interventions can usually be specified by listing the involved individuals although the interventions may change contact properties. In our performance tests presented in Section 7, we show that communications take only a small fraction of the total simulation running time.

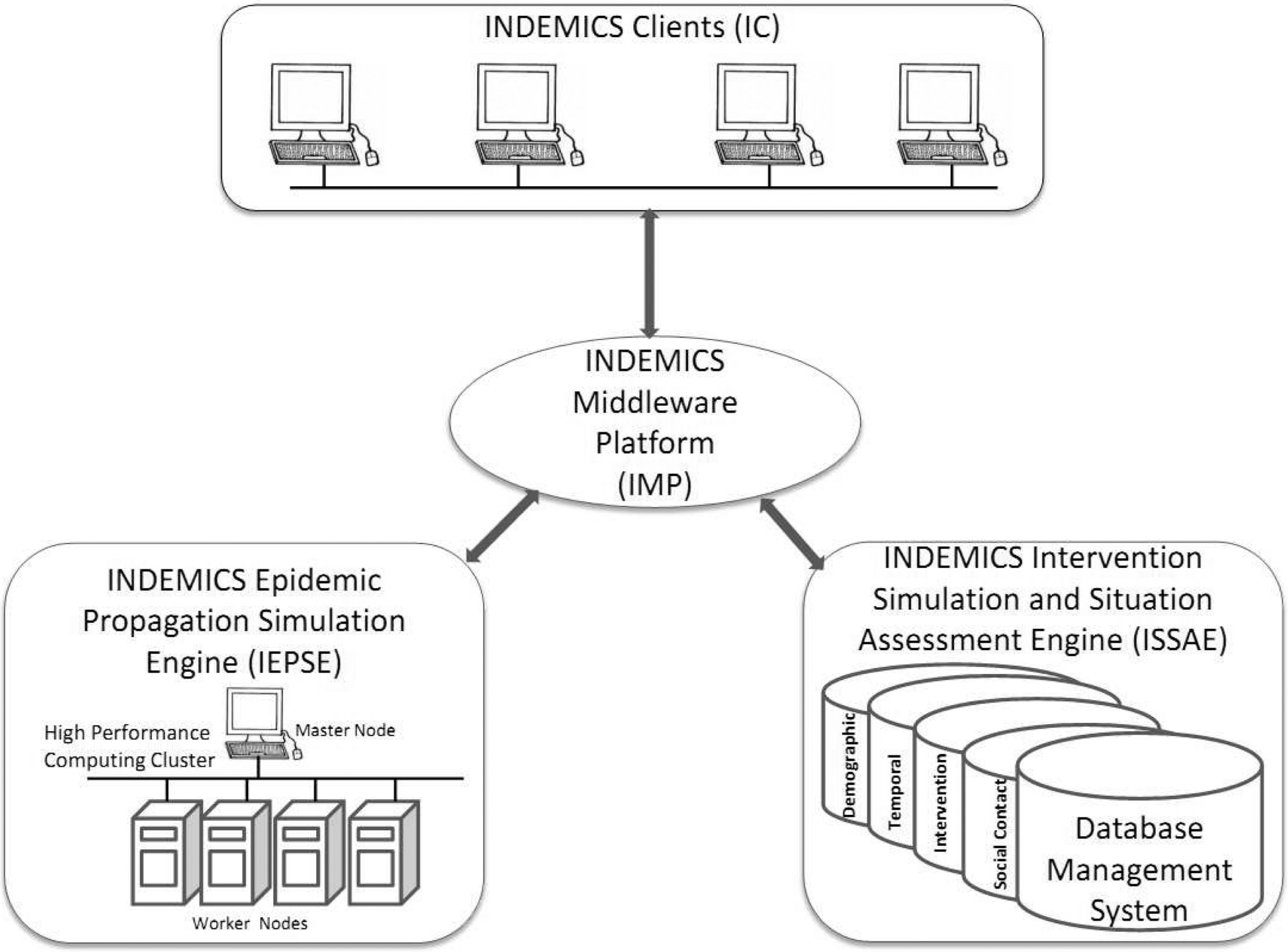

INDEMICS Clients (IC)
INDEMICS Middleware Platform (IMP)
INDEMICS Epidemic Propagation Simulation Engine (IEPSE)
High Performance Computing Cluster
Master Node
Worker Nodes
INDEMICS Intervention Simulation and Situation Assessment Engine (ISSAE)
Demographic
Temporal
Intervention
Social Contact
Database Management System

Fig. 1.

The high level architecture of Indemics. The Indemics middleware platform coordinates and synchronizes the communication between the Indemics Epidemic Propagation Simulation Engine, the Indemics Clients and the Indemics Intervention Simulation and Situation Assessment Engine.

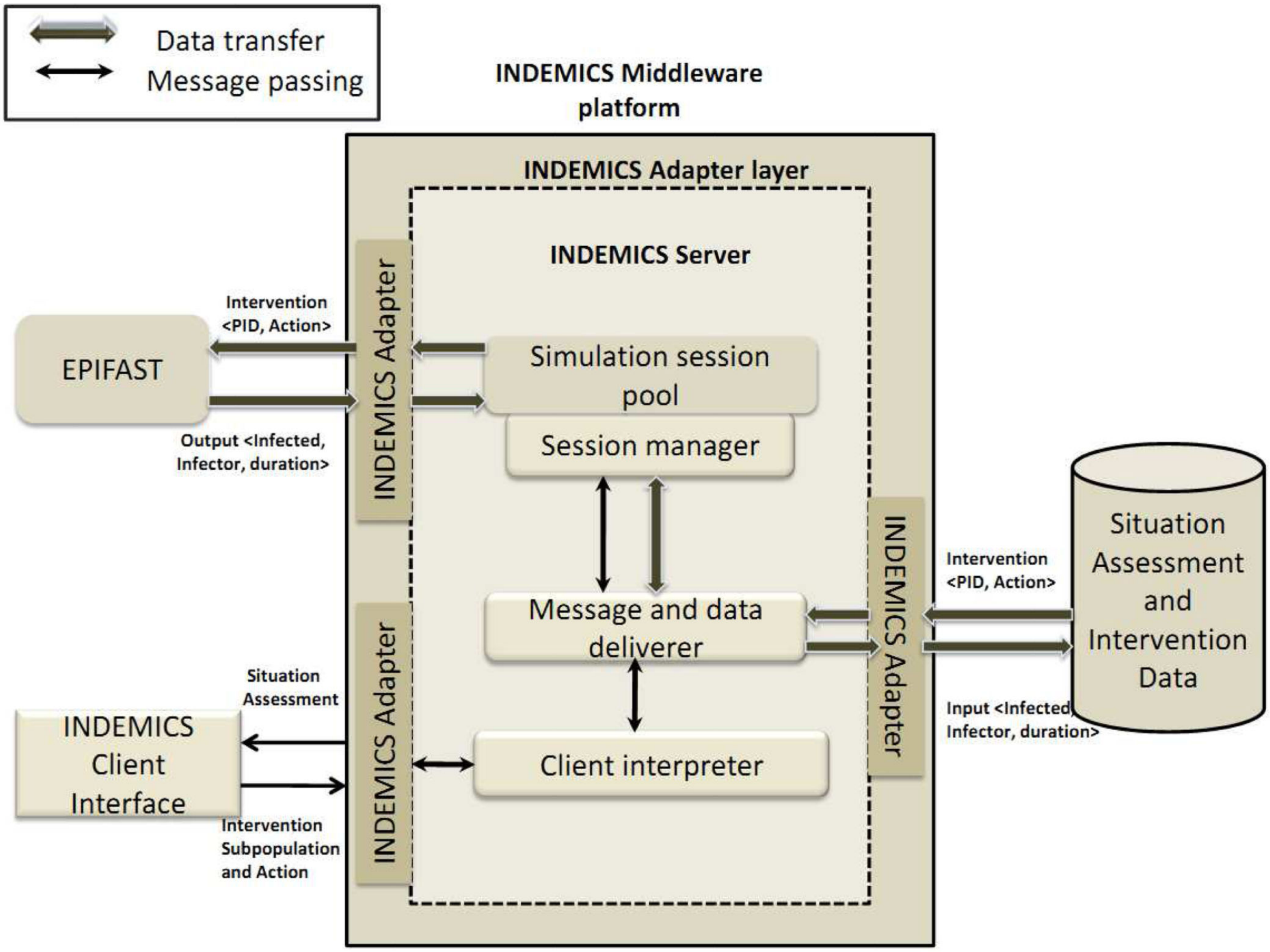

Data transfer
Message passing
INDEMICS Middleware platform
INDEMICS Adapter layer
INDEMICS Server
INDEMICS Adapter
Intervention <PID, Action>
EPIFAST
Output <Infected, Infector, duration>
Simulation session pool
Session manager
Message and data deliverer
Client interpreter
INDEMICS Adapter
Situation Assessment
INDEMICS Client Interface
Intervention Subpopulation and Action
INDEMICS Adapter
Intervention <PID, Action>
Input <Infected, Infector, duration>
Situation Assessment and Intervention Data

Fig. 2.

Details of INDEMICS middleware and data communication. The INDEMICS server plays the role of message/data delivering center. The INDEMICS adapter hides the details of data transformations and communication by providing a set of APIs to EPIFAST, INDEMICS clients and the database.

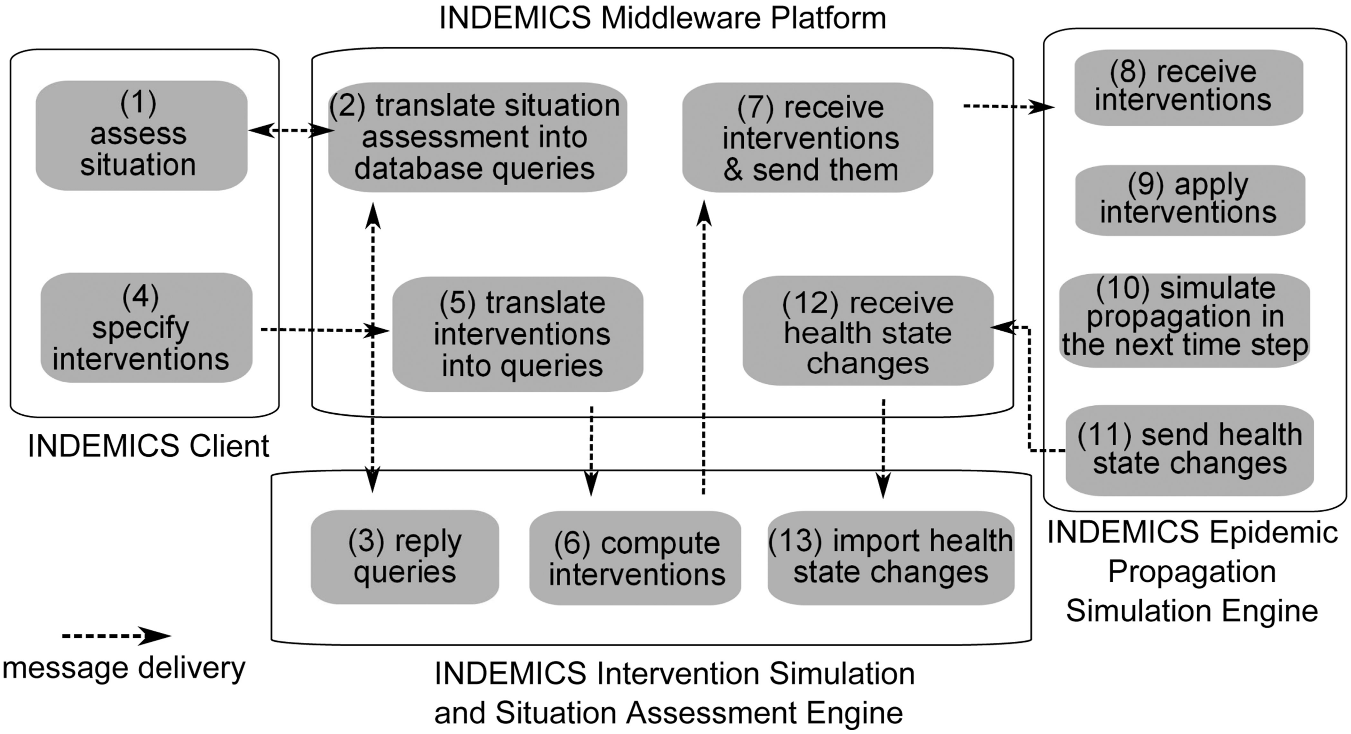
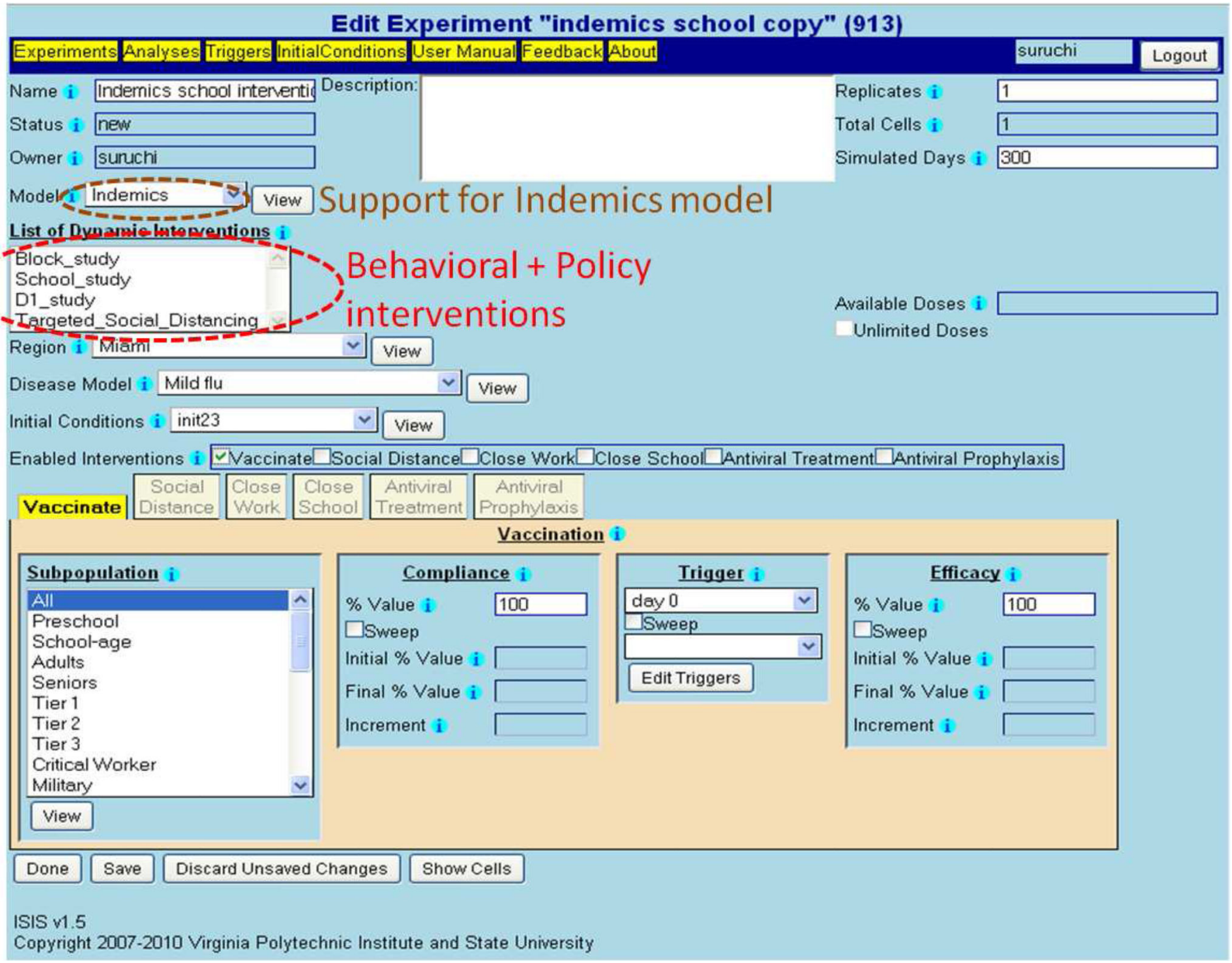

INDEMICS Middleware Platform
(1) assess situation
(2) translate situation assessment into database queries
(7) receive interventions & send them
(8) receive interventions
(9) apply interventions
(10) simulate propagation in the next time step
(11) send health state changes
(4) specify interventions
(5) translate interventions into queries
(12) receive health state changes
INDEMICS Client
(3) reply queries
(6) compute interventions
(13) import health state changes
INDEMICS Epidemic Propagation Simulation Engine
message delivery
INDEMICS Intervention Simulation and Situation Assessment Engine

Fig. 3.

The computation flow diagram of Indemics. The Indemics server coordinates the computations in the simulation engine and the client, and queries the database with control messages. The server also relays the data between the database and the simulation engine.



Edit Experiment "indemics school copy" (913)

Name: Indemics school interventic

Status: new

Owner: suruchi

Description:

Replicates: 1

Total Cells: 1

Simulated Days: 300

Model: Indemics

List of Dynamic Interventions

- Block_study
- School_study
- D1_study
- Targeted_Social_Distancing

Available Doses:

□ Unlimited Doses

Region: Miami

Disease Model: Mild flu

Initial Conditions: init23

Enabled Interventions: ☑ Vaccinate □ Social Distance □ Close Work □ Close School □ Antiviral Treatment □ Antiviral Prophylaxis

Vaccination

Subpopulation: All (selected); Preschool; School-age; Adults; Seniors; Tier 1; Tier 2; Tier 3; Critical Worker; Military

Compliance

% Value: 100

□ Sweep

Initial % Value:

Final % Value:

Increment:

Trigger

day 0

□ Sweep

Efficacy

% Value: 100

□ Sweep

Initial % Value:

Final % Value:

Increment:

ISIS v1.5

Copyright 2007-2010 Virginia Polytechnic Institute and State University

Fig. 4.

Screen shot of ISIS (Interface to Synthetic Information Systems) web-based interface for running simulations using Indemics. Using ISIS, users can set up and manage experiments for studying the effects of complex interventions to epidemic propagation simulation. Implementation details of ISIS interface for Indemics have been explained in [Deodhar et al. 2012]

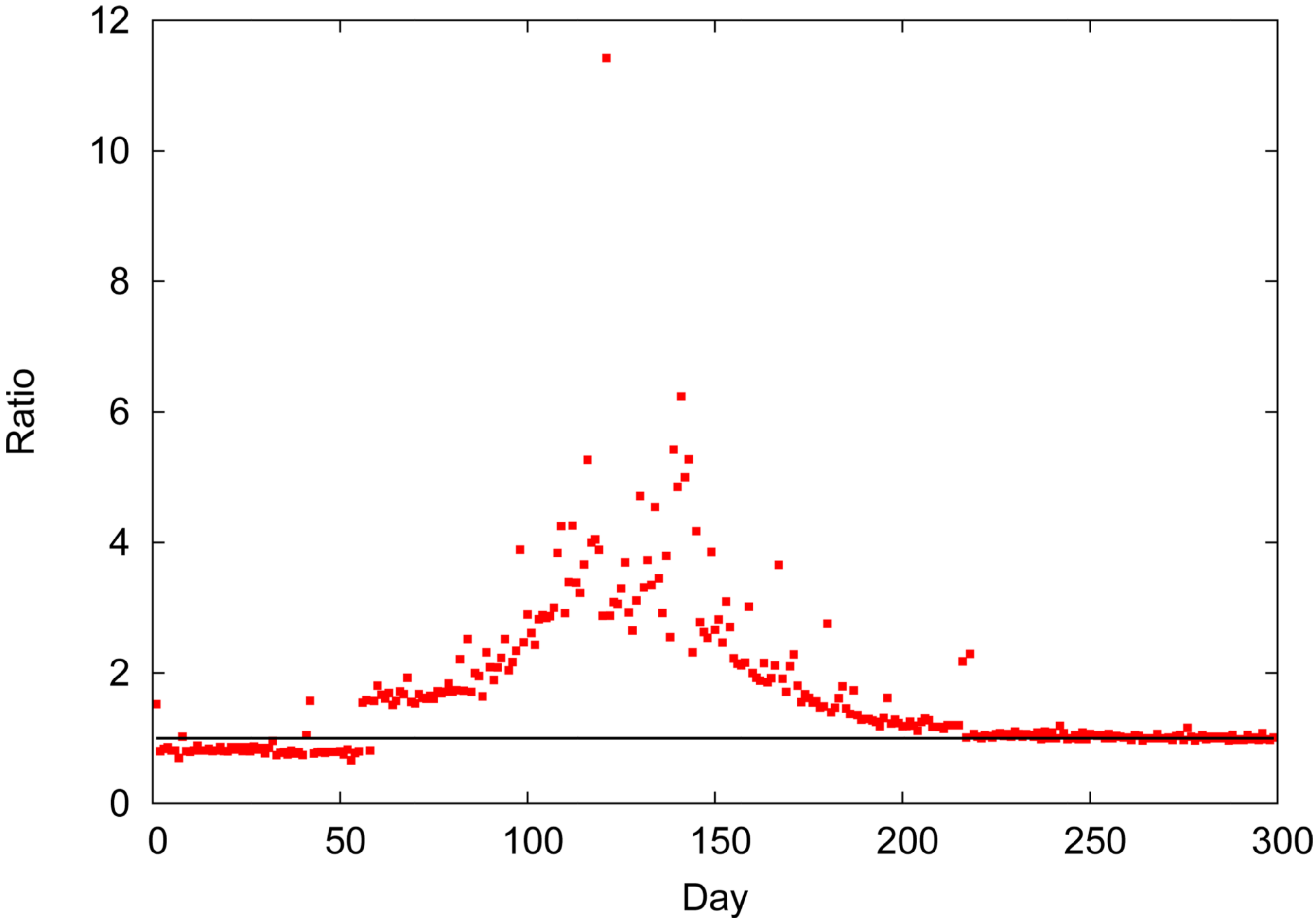


Fig. 5.

The performance overhead of Indemics. The plot shows the ratio of execution time with Indemics by that with EpiFast on each simulation day.

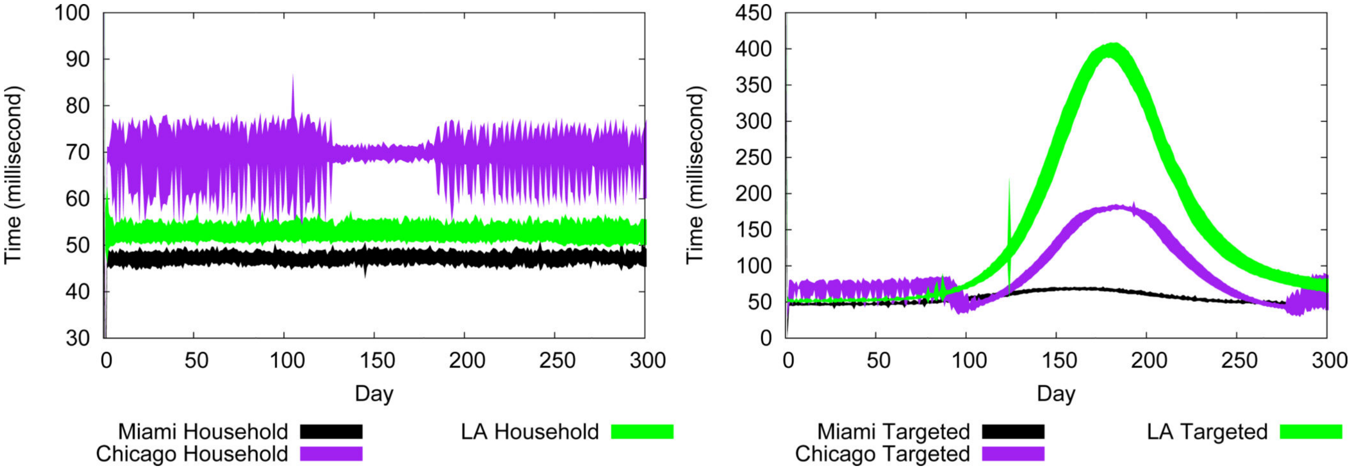


Fig. 6.

Cost of the disease propagation simulation. The average running time is shown for each simulation day. The width of each curve represents one standard deviation above and below the average. Note that in the right plot the Los Angeles curve has a variation spike on day 120 — it is an isolated case due to the fluctuation of the HPC resource at that moment.

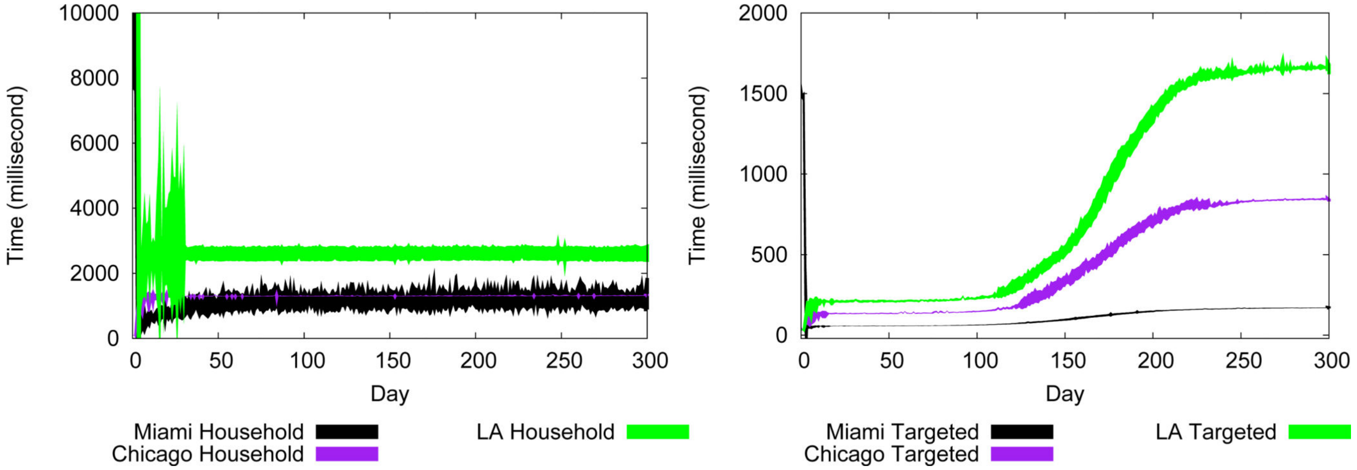


Fig. 7.

Cost of the database query in the intervention scenario simulation. The average query time is shown for each simulation day. The width of each curve represents one standard deviation above and below the average. Note that in the left plot, the Chicago and Miami curves overlap with each other.

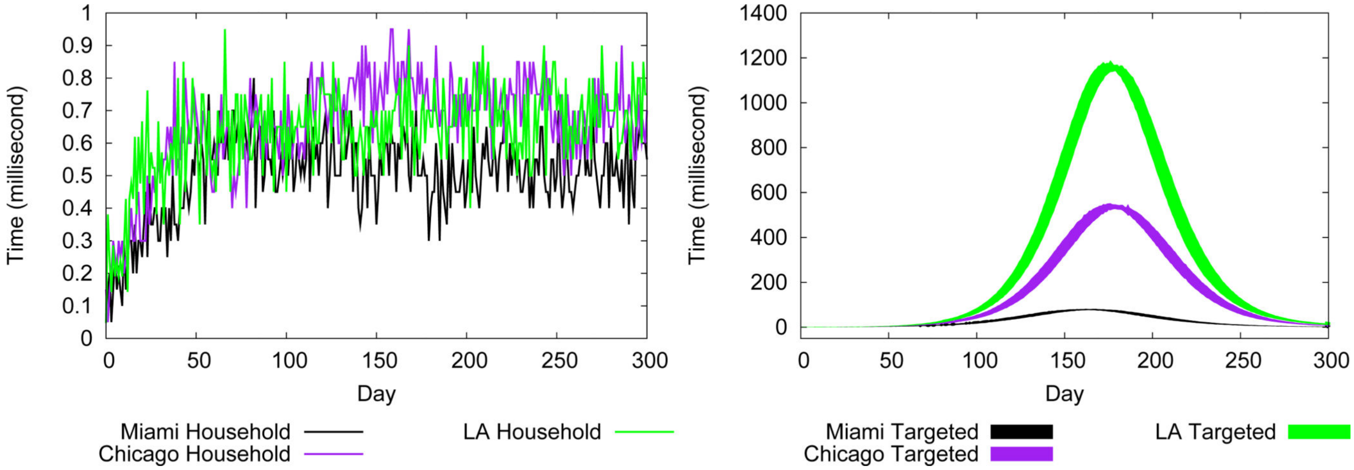


Fig. 8.

Cost of data communication between the epidemic propagation simulation and the database. The average communication time is shown for each simulation day. In the right plot (targeted intervention) the width of each curve represents one standard deviation above and below the average. In the household intervention case, both the average communication time and its standard deviation are less than one millisecond on any simulation day. For readability we do not show the standard deviation in the left plot.

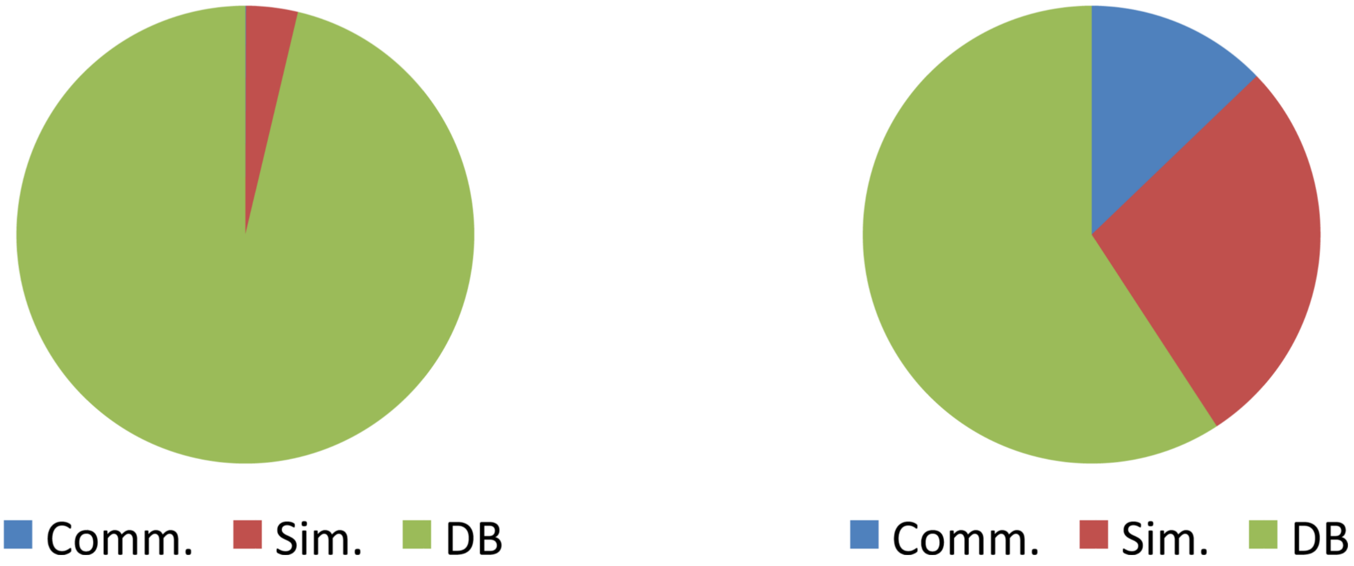


Fig. 9.

The performance comparisons between the disease propagation simulation (Sim.), the intervention simulation (DB), and the communication (Comm.) for Miami household intervention (left figure) and targeted intervention (right figure). The database query time in the household intervention simulation is dominant because of the query complexity.

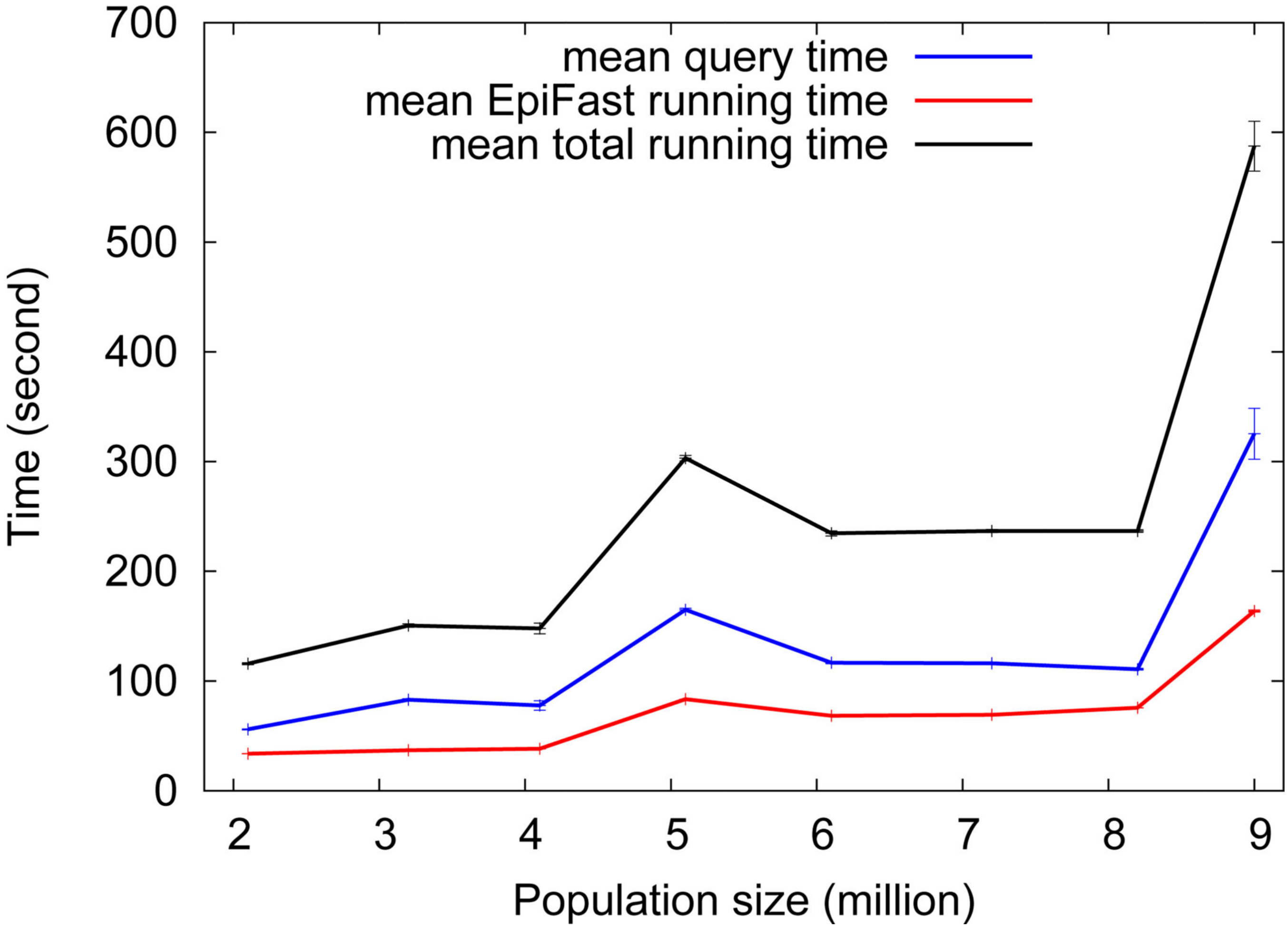


Fig. 10.

Plot summarizing how Indemics scales as a function of the size of the social contact network. The curves show the average running time of the epidemic propagation simulation (EpiFast), that of the intervention simulation (database query), and that of the whole simulation. Standard deviations are shown with error-bars and they are very small at most data points.

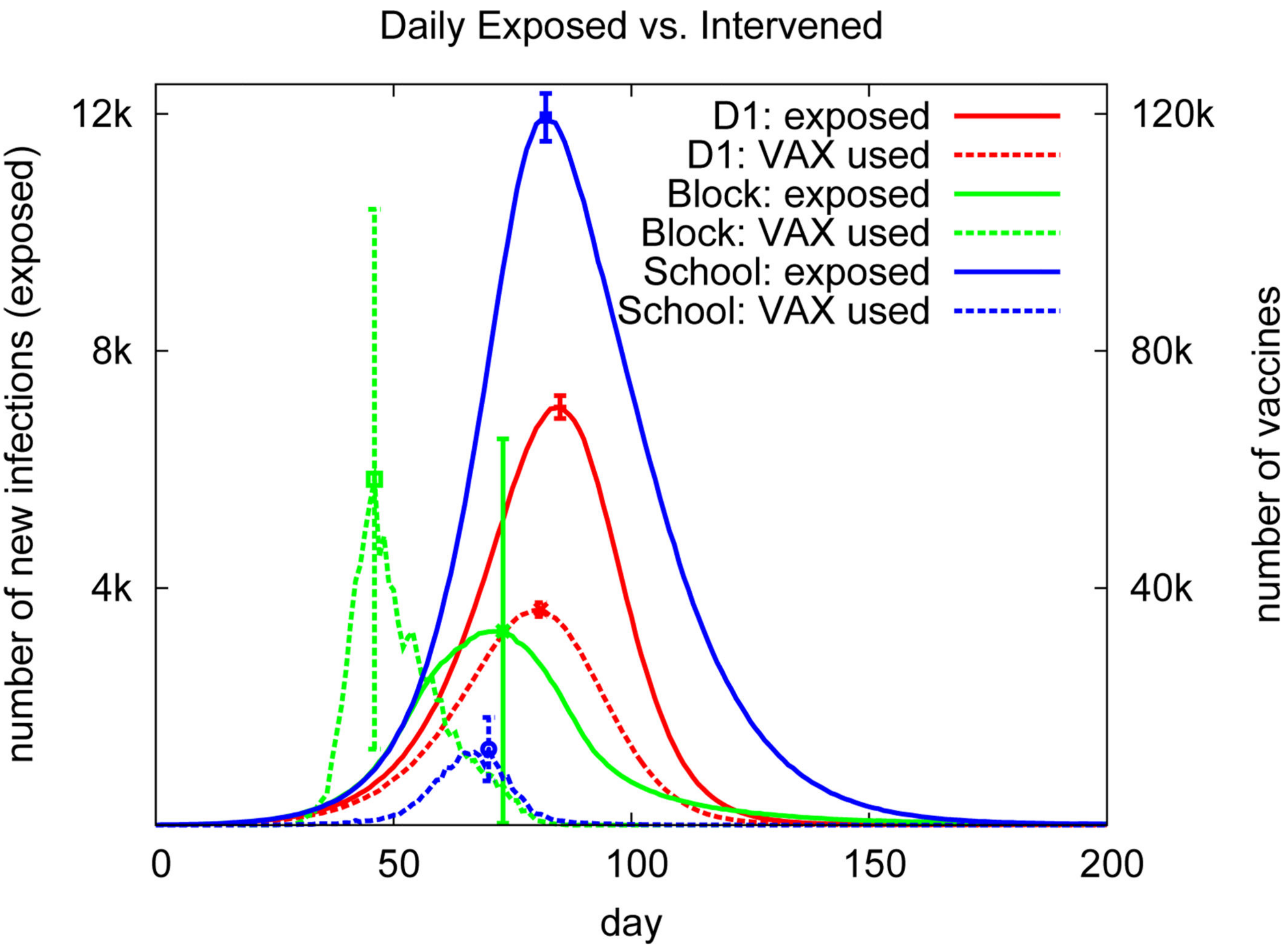


Fig. 11.

Number of people exposed versus the number of vaccines (VAX) used on a daily basis with each of the three interventions.

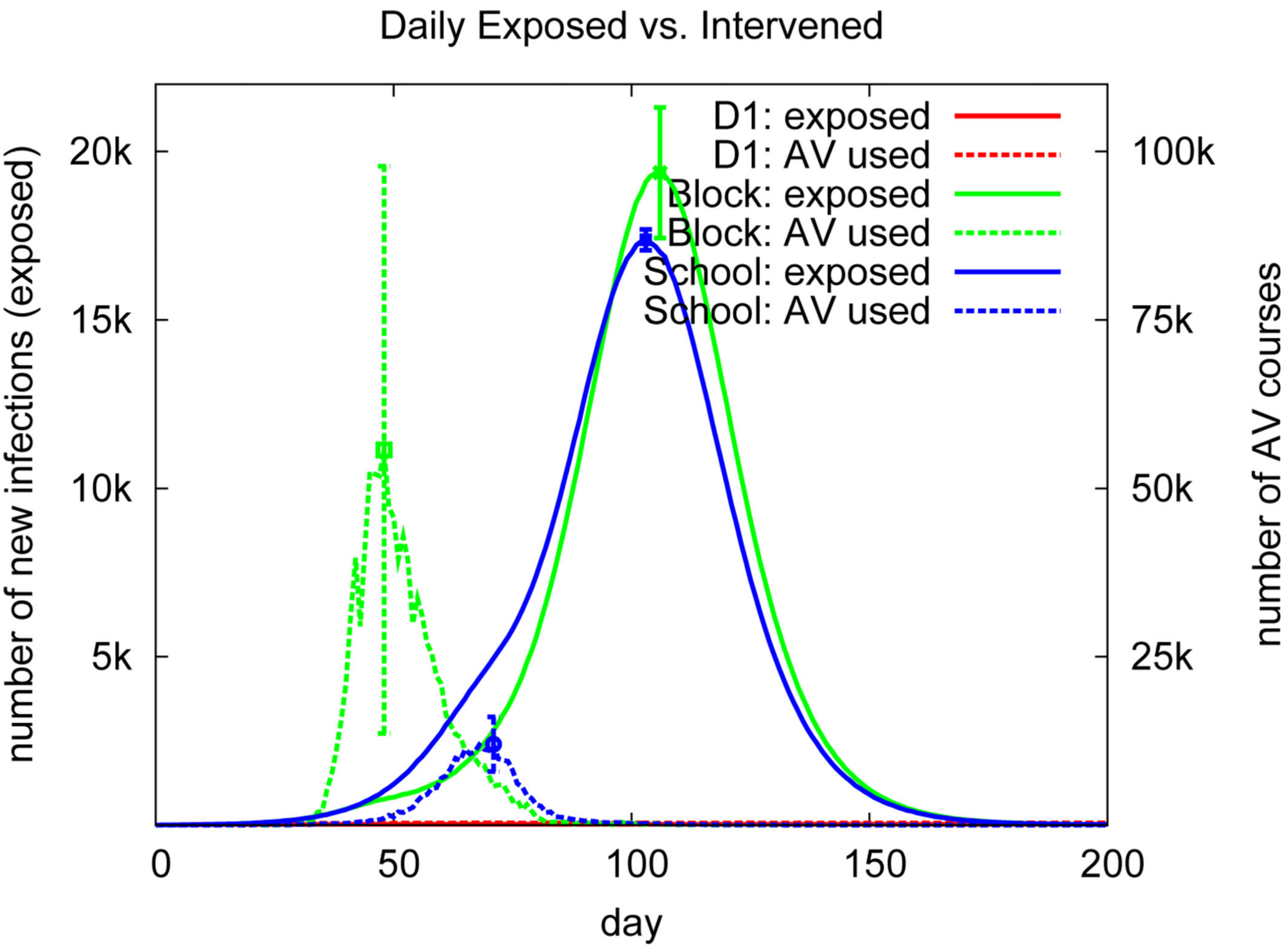


Fig. 12.

Number of people exposed versus the number of courses of antiviral (AV) used on a daily basis with each of the three interventions.

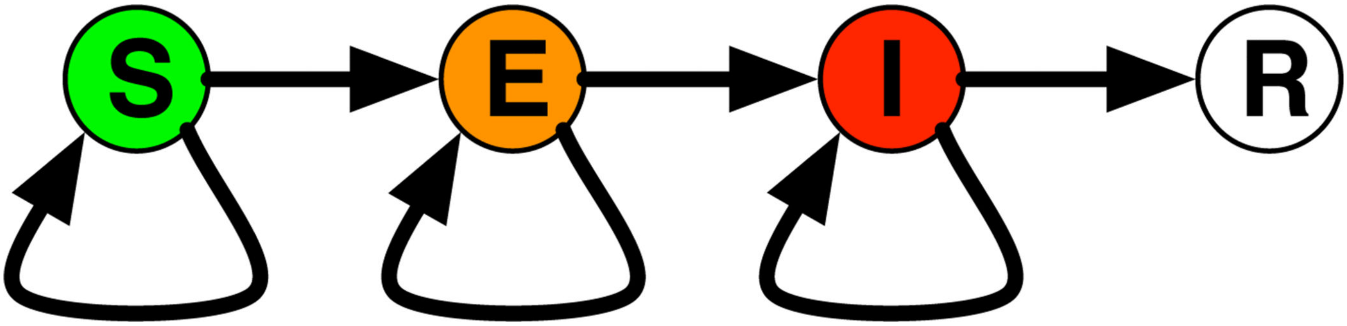


Fig. 13.
State transitions in the SEIR disease model.

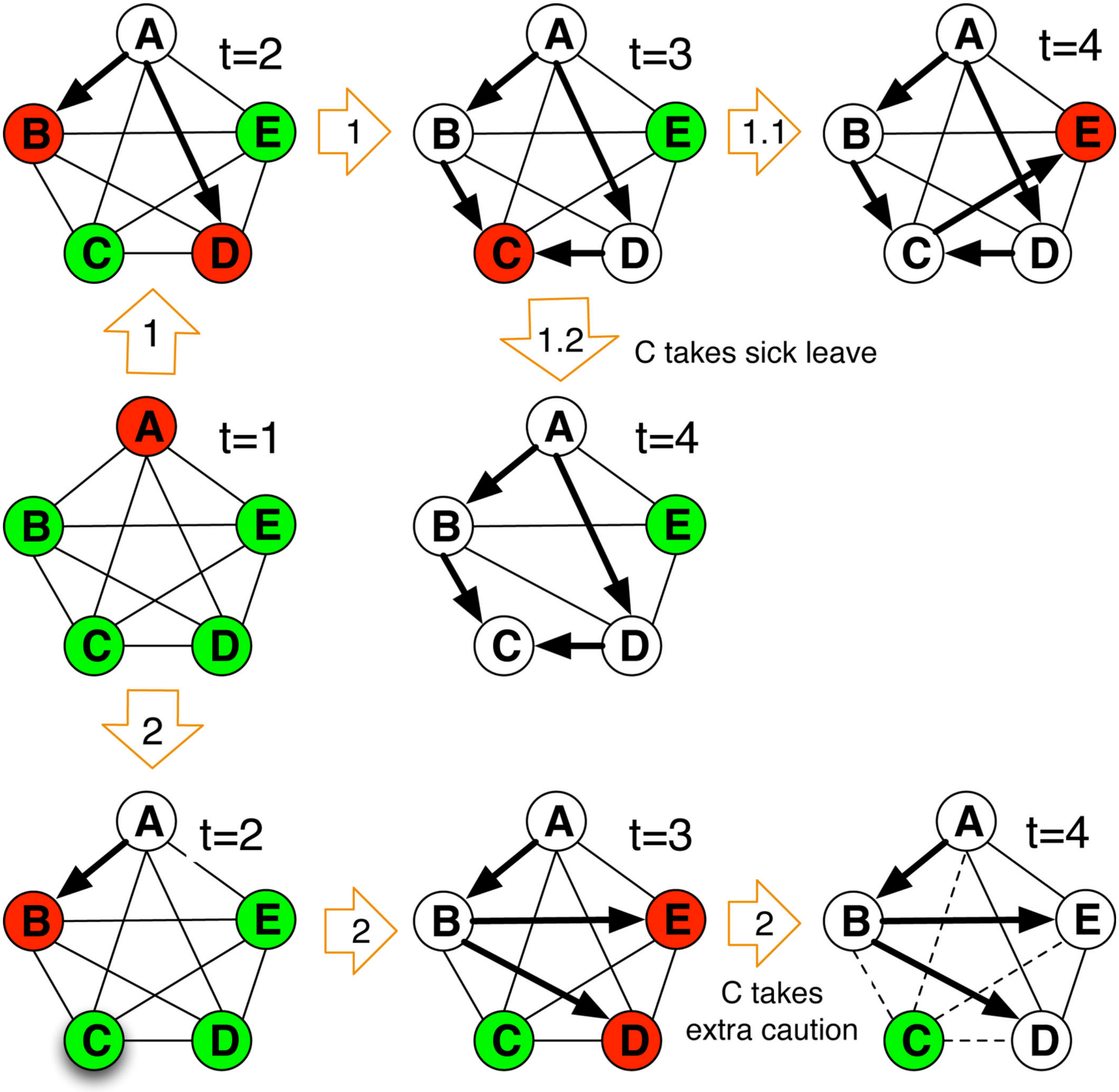


Fig. 14.

This example shows the stochastic disease propagation in a social contact network and the effect of interventions.

Table I

Extended CGDDS in EpiFast implementation: A concrete example.

CGDDS component	in EpiFast
$\mathbb{D}$	SEIR (Susceptible, Exposed, Infectious and Recovered, see Appendix A.1) state, infectivity, vulnerability, diagnosis, symptomatic state
$\mathbb{L}$	contact duration, type
f	stochastic transmission from infectious vertex to susceptible vertex with probability determined by the infectivity of infectious vertex, the vulnerability of susceptible vertex and the duration of contact between them
g^V	g_1^V : seeding (e.g. for 5 pre-selected vertices, set their states to *exposed* if they are still *susceptible*) g_2^V : within-host transition of SEIR state (e.g. if vertex *v* was exposed 2 days ago, then its state changes to *infectious*; if it was exposed 7 days ago, then its state changes to *removed*; otherwise its state does not change) g_3^V : diagnosis (vertices in infectious state show symptoms and get diagnosed randomly) PI's (e.g. g_4^V : apply antiviral to pre-selected vertices to reduce their infectivity and vulnerability)
g^E	NPI's (e.g. g_1^E : diagnosed vertices stay home, so they have more contacts with household members and no contacts with other people such as co-workers; g_2^E : vertices with a symptomatic co-worker reduce contacts in the workplace)
W	pre-determined schedule, e.g. g_1^V on day 1, g_2^V and g_3^V and g_2^E on each day, g_4^V on day 30, g_1^E on each day after day 60

Table II

Examples of interventions studied in literature.

Intervention	Subpopulation	Triggering condition	Action
Antiviral treatment of diagnosed cases [Halloran et al. 2008]	diagnosed people	over 1% of population are infected	reduce their infectivity
Home isolation of diagnosed cases [Halloran et al. 2008]	diagnosed people	over 1% of population are infected	remove their contacts with non-household members
School closure when disease prevalence is high [Halloran et al. 2008]	all school age children	over 1% of population are infected	remove in-school contacts between them
Deference of travel to unaffected areas [WHO 2004]	people who live in an affected area and plan to travel to unaffected areas	over 1% of population are infected	remove their contacts with people outside of the affected area
Avoidance of contact with high-risk environments [WHO 2004]	people going to high-risk environments	early phase: disease prevalence > 0.1%	remove their contacts with people in high-risk environments
Vaccination of people in any census block with an outbreak [Marathe et al. 2011]	all people living in affected census blocks	number of diagnosed cases in the block > 1% of block population	increase their immunity
Social targeting with antiviral prophylaxis [Ferguson et al. 2005]	individuals in the same household, school or workplace with diagnosed cases	cases are diagnosed in this household, school, or workplace	increase their immunity

Table III

Urban regions used in our experiments in Section 7.2.

Region	Population $|V|$	Contacts $|E|$	Avg age	Households	Median HH income	CPUs
Miami	2.1M	53M	36.1	0.7M	35K	16
Chicago	9.0M	262M	34.3	3.3M	50K	16
LA	16.2M	460M	32.9	5.4M	46K	32

Table IV

The running time for data communication, disease propagation simulation, intervention computation by database, and the total execution time (all in seconds).

Intervention	Communication (sec)	Propagation (sec)	DB query (sec)	Total (sec)
Miami Household	0.2	14	370	384
Miami Targeted	7.8	17	36	61
Chicago Household	0.2	21	387	408
Chicago Targeted	45	29	132	206
LA Household	0.2	16	1015	1031
LA Targeted	89	50	247	386

Table V

The regions in the scalability experiments.

Region	Population size (million)	Contacts (million)	Mean household size
Miami	2.1	53	2.87
Seattle	3.2	87	2.50
Boston	4.1	110	2.54
Dallas	5.1	141	2.71
Indiana	6.1	172	2.47
Virginia	7.2	204	2.47
New Jersey	8.2	212	2.60
Chicago	9.0	262	2.72